Arkady Zaslavsky
Zhaolong Ning
Vana Kalogeraki
Dimitrios Georgakopoulos
Panos K. Chrysanthis (Eds.)



593

Mobile and Ubiquitous Systems

Computing, Networking and Services

LNICST

20th EAI International Conference, MobiQuitous 2023
Melbourne, VIC, Australia, November 14–17, 2023
Proceedings, Part I

Part 1





Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 593

Editorial Board Members

Ozgur Akan, *Middle East Technical University, Ankara, Türkiye*
Paolo Bellavista, *University of Bologna, Bologna, Italy*
Jiannong Cao, *Hong Kong Polytechnic University, Hong Kong, China*
Geoffrey Coulson, *Lancaster University, Lancaster, UK*
Falko Dressler, *University of Erlangen, Erlangen, Germany*
Domenico Ferrari, *Università Cattolica Piacenza, Piacenza, Italy*
Mario Gerla, *UCLA, Los Angeles, USA*
Hisashi Kobayashi, *Princeton University, Princeton, USA*
Sergio Palazzo, *University of Catania, Catania, Italy*
Sartaj Sahni, *University of Florida, Gainesville, USA*
Xuemin Shen, *University of Waterloo, Waterloo, Canada*
Mircea Stan, *University of Virginia, Charlottesville, USA*
Xiaohua Jia, *City University of Hong Kong, Kowloon, Hong Kong*
Albert Y. Zomaya, *University of Sydney, Sydney, Australia*

The LNICST series publishes ICST's conferences, symposia and workshops.

LNICST reports state-of-the-art results in areas related to the scope of the Institute. The type of material published includes

- Proceedings (published in time for the respective event)
- Other edited monographs (such as project reports or invited volumes)

LNICST topics span the following areas:

- General Computer Science
- E-Economy
- E-Medicine
- Knowledge Management
- Multimedia
- Operations, Management and Policy
- Social Informatics
- Systems

Arkady Zaslavsky · Zhaolong Ning ·
Vana Kalogeraki · Dimitrios Georgakopoulos ·
Panos K. Chrysanthis
Editors

Mobile and Ubiquitous Systems

Computing, Networking and Services

20th EAI International Conference, MobiQuitous 2023
Melbourne, VIC, Australia, November 14–17, 2023
Proceedings, Part I

Springer

Editors
Arkady Zaslavsky
Deakin University
Melbourne, VIC, Australia

Vana Kalogeraki
Athens University of Economics
and Business
Athens, Greece

Panos K. Chrysanthis
University of Pittsburgh
Pittsburgh, PA, USA

Zhaolong Ning
Chongqing University of Posts
and Telecommunications
Chongqing, China

Dimitrios Georgakopoulos
Swinburne University of Technology
Melbourne, VIC, Australia

ISSN 1867-8211 ISSN 1867-822X (electronic)
Lecture Notes of the Institute for Computer Sciences, Social Informatics
and Telecommunications Engineering
ISBN 978-3-031-63988-3 ISBN 978-3-031-63989-0 (eBook)
https://doi.org/10.1007/978-3-031-63989-0

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

For the first time after the COVID-19 world pandemic, the 20th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (EAI MobiQuitous 2023) was held as an on-site conference with an option to present remotely. The conference took place on November 14–17, 2023, in Melbourne, Australia, at the RMIT University, which acted as a co-sponsor of the event. This year's MobiQuitous was also special as it celebrated its 20th edition as a premier mobile and ubiquitous computing forum. We are delighted to introduce the proceedings, which match MobiQuitous's longstanding tradition of providing unique opportunities for participants, researchers, industry representatives, and students from around the world to explore new ideas, techniques, and tools, and to exchange experiences in all aspects of mobile and ubiquitous computing including systems, applications, social networks, middleware, networking, sensing, IoT, data management, data processing, and services. This year, some emerging themes were federated machine learning, mobile edge and fog computing, mobile augmented reality, and interaction technologies.

This year, 161 research papers were submitted, which were subject to a rigorous review procedure; every paper had three to five independent reviews. 57 full-length papers (i.e., an acceptance rate of 35%) and 4 short papers (i.e., a total acceptance rate of 37%) were selected for publication in the proceedings. The accepted papers were organized into 15 research sessions based on research on tracking systems, Internet of Things (IoT), federated learning, networks, activity recognition, cyber and internet security, urban and mobile crowdsensing, edge computing, crowdsourcing platforms and localizations, activity recognition and prediction, machine learning, mobile edge and fog computing, mobile augmented reality, and applications for mobile computing and interaction technologies. In addition to the research sessions, the program included one workshop and a PhD track event on Day 1. The workshop discussed 5 papers under the theme of *AutoQuitous*.

MobiQuitous 2023 featured two keynote speeches, and three invited presentations. The first keynote speech is titled "*Emerging haptically-enabled systems for immersive simulation-based training*" by Saeid Nahavandi, Distinguished Professor and inaugural Associate Deputy Vice-Chancellor Research and Chief of Defence Innovation at Swinburne University of Technology, Australia. The second keynote speech is titled "*MinerFinder: A GAE-LSTM method for predicting spatio-temporal locations of miners in underground mines*" by Sanjay K. Madria, a Curators' Distinguished Professor in the Department of Computer Science at the Missouri University of Science and Technology, USA. The first invited talk was on *The Online Dynamic Path Planner for UAVs*, submitted by a group of researchers and officially presented by the keynote speaker Sanjay K. Madria. The second one was on *Securing Wireless Communication in Critical Infrastructure: Challenges and Opportunities*, officially submitted by Avinaba Mistry from the University of New Brunswick, Canada. The last one was on *Scalable Contact*

Tracing Over Uncertain Trajectories, submitted and presented by Martin Henze from RWTH Aachen University & Fraunhofer FKIE, Germany.

We owe our sincere gratitude to the members of all the Technical Program Committees who contributed in assembling such a high-quality program, working within a very tight schedule. Several additional people contributed to the successful organization of the MobiQuitous 2023 conference. Special thanks go to the following valued collaborators for their dedication to excellence: Workshop Chairs Gregor Shiele (Duisburg-Essen University, Germany) and Herodotos Herodotou (Cyprus University of Technology, Cyprus); Demonstration Chair Manos Papagelis (York University, Canada); Panel Chairs Michael Sheng (Macquarie University, Australia) and Kyriakos Moraitides (Singapore Management University, Singapore); Tutorials Chairs Xiaojie Wang (Chongqing University of Posts and Telecommunications, China), Dali Kaafar (Macquarie University / CSIRO Data61, Australia) and Demetris Zeinalipour (University of Cyprus, Cyprus); Sponsorship & Exhibits Chairs Wei Xiang (La Trobe University, Australia) and Xu Jianliang (Hong Kong Baptist University, Hong Kong); Posters and PhD Track Chairs Jinho Choi (Deakin University, Australia) and Lina Yao (UNSW, Australia); Publications Chair Saguna Saguna (Lulea University of Technology, Sweden); Web Chair Maria Spichkova (RMIT University, Australia); Publicity and Social Media Chair Azadeh Ghari Neiat (Deakin University, Australia); and Local Chairs Harindu Korala (Monash University, Australia) and Anas Dawod (Swinburne University of Technology, Australia).

We would also like to express our special appreciation to the steering committee: Tao Gu (Macquarie University, Australia, Chair), Takahiro Hara (Osaka University, Japan) and Max Mühlhäuser (Technische Universitat Darmstadt, Germany), and to the European Alliance for Innovation and its conference management team – Martin Hochel, Marica Scevlikova, and Radka Vasileiadis – for the overall support provided throughout the process. Finally, special thanks go to the authors who submitted papers and to the presenters; without their contribution, MobiQuitous 2023 would not be possible.

We hope you will find the papers included in the proceedings interesting and stimulating, and that they will lead to new discoveries, innovations, and collaborations.

Dimitrios Georgakopoulos
Panos Chrysanthis
Arkady Zaslavsky
Zhaolong Ning
Vana Kalogeraki

Organization

Steering Committee

Tao Gu	Macquarie University, Australia
Takahiro Hara	Osaka University, Japan
Max Mühlhäuser	Technische Universitat Darmstadt, Germany

Organizing Committee

General Chairs

Dimitrios Georgakopoulos	Swinburne University of Technology, Australia
Panos K. Chrysanthis	University of Pittsburgh, USA

TPC Chairs and Co-chairs

Arkady Zaslavsky	Deakin University, Australia
Zhaolong Ning	Chongqing University of Posts and Telecommunications, China
Vana Kalogeraki	Athens University of Economics and Business, Greece

Sponsorship and Exhibit Chairs

Wei Xiang	La Trobe University, Australia
Xu Jianliang	Hong Kong Baptist University, China

Local Chairs

Harindu Korala	Monash University, Australia
Anas Dawod	Swinburne University of Technology, Australia

Workshops Chairs

Gregor Shiele	Duisburg-Essen University, Germany
Herodotos Herodotou	Cyprus University of Technology, Cyprus

Publicity and Social Media Chair

Azadeh Ghari Neiat	Deakin University, Australia

Publications Chair

Saguna Saguna	Luleå University of Technology, Sweden

Web Chair

Maria Spichkova	RMIT University, Australia

Posters and PhD Track Chairs

Jinho Choi	Deakin University, Australia
Lina Yao	UNSW Sydney, Australia

Panels Chairs

Michael Sheng	Macquarie University, Australia
Kyriakos Mouratidis	Singapore Management University, Singapore

Demos Chair

Manos Papagelis	York University, Canada

Tutorials Chairs

Xiaojie Wang	Chongqing University of Posts and Telecommunications, China
Dali Kaafar	Macquarie University/CSIRO Data61, Australia
Demetris Zeinalipour	University of Cyprus, Cyprus

Technical Program Committee

Rakib Abdur	Coventry University, UK
Christian Becker	University of Stuttgart, Germany
Nikos Bikakis	HMU & Athena Research Center, Greece
Cristian Borcea	New Jersey Institute of Technology, USA
Kevin Matthe Caramancion	University of Wisconsin-Stout, USA

Zhichao Cao	Michigan State University, USA
Jagmohan Chauhan	University of Southampton, UK
Junxin Chen	Dalian University of Technology, China
Tao Chen	University of Pittsburgh, USA
Maria Luisa Damiani	University of Milan, Italy
Klaus David	University of Kassel, Germany
Pari Delir Haghighi	Monash University, Australia
Alexis Delis	University of Athens, Greece
Christos Efstratiou	University of Kent, UK
Xiaoran Fan	Google, USA
Stefan Fischer	University of Lübeck, Germany
Francesco Flammini	Mälardalen University, Sweden
Thippa Reddy Gadekallu	Vellore Institute of Technology, India
Honghao Gao	Gachon University, South Korea
Akshay Gadre	Carnegie Mellon University, USA
Jean Philippe Georges	Université de Lorraine, France
Giorgos Giannopoulos	Athena Research Center, Greece
Voncenzo Gulis	Chalmers University, Sweden
Xiuzhen Guo	Tsinghua University, China
Stathes Hadjiefthymiades	University of Athens, Greece
Armin Haller	Australian National University, Australia
Takahiro Hara	Osaka University, Japan
Martin Henze	RWTH Aachen University, Germany & Fraunhofer FKIE, Germany
Prem Prakash Jayaraman	Swinburne University, Australia
Beihong Jin	Institute of Software, Chinese Academy of Sciences, China
Salil Kanhere	UNSW, Australia
Sylvain Kubler	University of Luxembourg, Luxembourg
Satoshi Kurihara	Keio University, Japan
Brano Kusy	CSIRO, Australia
Brent Lagesse	University of Washington Bothell, USA
Guohao Lan	Delft University of Technology, Netherlands
Stephen Lee	University of Pittsburgh, USA
Kevin Lee	Deakin University, Australia
Wang-Chien Lee	Pennsylvania State University, USA
Chi Lin	Dalian University of Technology, China
Sicong Liu	Northwestern Polytechnical University, China
Sanjay Madria	Missouri University of Science and Technology, USA
Sajib Mistry	Curtin University, Australia
Karan Mitra	Luleå University of Technology, Sweden

Valeh Moghaddam	Deakin University, Australia
Mohamed Mokbel	University of Minnesota, USA
Manzur Murshed	Deakin University, Australia
Amiya Nayak	University of Ottawa, Canada
Anne Ngu	Texas State University, USA
VP Nguyen	University of Texas at Arlington, USA
Andrea Passarella	IIT-CNR, Italy
Tomasz Pieciak	Universidad de Valladolid, Spain
Evaggelia Pitoura	University of Ioannina, Greece
Suprio Ray	University of New Brunswick, USA
George Roussos	Birkbeck, University of London, UK
Stephan Sigg	Aalto University, Finland
Amit Kumar Singh	National Institute of Technology Patna, India
Dimitris Skoutas	Athena Research Center, Greece
Dimitris Spathis	Bell Labs, USA
Sanjib Sur	University of South Carolina, USA
Gautam Srivastava	Brandon University, Canada
Chiu Tan	Temple University, USA
Mohammad Tanveer	Indian Institute of Technology Indore, India
Zahir Tari	RMIT University, Australia
My Thai	University of Florida, USA
Arm Tolba	King Saud University, Saudi Arabia
Dimitrios Tomaras	Athens University of Economics and Business, Greece
Jeffrey Tweedale	Defence Science and Technology Group, Australia
Zhongnan Qu	ETH Zurich, Switzerland
Ju Wang	Samsung AI Center-Montreal, Canada
Minmei Wang	University of Connecticut, USA
Chenshu Wu	University of Hong Kong, China
Marcin Woźniak	Silesian University of Technology, Poland
Yaxiong Xie	Princeton University, USA
Zongxing Xie	Stony Brook University, USA
Jingao Xu	Tsinghua University, China
George Xylomenos	Athens University of Economics and Business, Greece
Lei Yang	Hong Kong Polytechnic University, China
Vladimir Zadorozhny	University of Pittsburgh, USA
Fusang Zhang	Institute of Software, Chinese Academy of Sciences, China
Leo Zhang	Griffith University, Australia

Xiaolong Zheng	Beijing University of Posts and Telecommunications, China
Yuanqinq Zheng	Hong Kong Polytechnic University, China
Yanmin Zhu	Shanghai Jiao Tong University, China

Contents – Part I

Tracking and Detection

NOD: Lightweight Continuous Neighbor Discovery on Everyday Devices 3
Hsiao-Yuan Chen, Evan King, and Christine Julien

Decentralized Collaborative Inertial Tracking 26
Alpha Diallo and Benoît Garbinato

High-Performance Features in Generalizable Fingerprint-Based Indoor Positioning 46
Andrea Brunello, Angelo Montanari, Nicola Saccomanno, and Joaquín Torres-Sospedra

SCORE: Scalable Contact Tracing over Uncertain Trajectories 68
Avinaba Mistry, Xichen Zhang, Suprio Ray, and Sanjeev Seahra

IoT

FaultBit: Generic and Efficient Wireless Fault Detection Using the Internet of Things 97
Koustabh Dolui, Ashok Samraj Thangarajan, Sergii Morshchavka, Zhaoyi Liu, Sam Michiels, and Danny Hughes

DeepHeteroIoT: Deep Local and Global Learning over Heterogeneous IoT Sensor Data 119
Muhammad Sakib Khan Inan, Kewen Liao, Haifeng Shen, Prem Prakash Jayaraman, Dimitrios Georgakopoulos, and Ming Jian Tang

Federated Reinforcement Learning for Automated LoRaWAN Management in Industrial IoT 136
Ameer Ivoghlian, Zoran Salcic, and Kevin I-Kai Wang

A Hybrid Approach to Monitor Context Parameters for Optimising Caching for Context-Aware IoT Applications 156
Ashish Manchanda, Prem Prakash Jayaraman, Abhik Banerjee, Arkady Zaslavsky, Shakthi Weerasinghe, and Guang-Li Huang

LOADHoC: Towards the Automatic Local Distribution of Computation Using Existing IoT Devices 175
Shaine Christmas, Kevin Lee, and Jean-Guy Schneider

E-Go Bicycle Intelligent Speed Adaptation System for Catching the Green Light 201
Khalil Ben Fredj, Akhil Reddy Pallamreddy, Geert Heijenk, Paul Havinga, and Yanqiu Huang

Federated Learning

FedGCS: Addressing Class Imbalance in Long-Tail Federated Learning 223
Guozheng Liu, Wei Zhang, Huiling Shi, Lizhuang Tan, Chang Tang, and Meihong Yang

FedRC: Representational Consistency Guided Model Uploading Mechanism for Asynchronous Federated Learning 239
Sheng Liu, Linlin You, and Yuren Zhou

RADEAN: A Resource Allocation Model Based on Deep Reinforcement Learning and Generative Adversarial Networks in Edge Computing 257
Zhaoyang Yu, Sinong Zhao, Tongtong Su, Wenwen Liu, Xiaoguang Liu, Gang Wang, Zehua Wang, and Victor C. M. Leung

Networks

A Stream Data Service Framework for Real-Time Vehicle Companion Discovery 281
Zhongmei Zhang and Shuai Zhang

KS-Autoformer: An Autoformer-Based SOC Prediction Framework for Electric Vehicles 297
Yaoyidi Wang, Niansheng Chen, Lei Rao, Dingyu Yang, Guangyu Fan, Songlin Cheng, and Xiaoyong Song

Deep Reinforcement Learning-Based Multi-node Collaborative Task Offloading Optimization in 6G Space-Air-Ground Integrated Networks 313
Wang Li, Xin Chen, Libo Jiao, Wangzhong Ning, and Wenwu Zhu

Securing Wireless Communication in Critical Infrastructure: Challenges and Opportunities 333
Jörn Bodenhausen, Christian Sorgatz, Thomas Vogt, Kolja Grafflage, Sebastian Rötzel, Michael Rademacher, and Martin Henze

Activity Recognition

Cross-User Activity Recognition via Temporal Relation Optimal Transport 355
Xiaozhou Ye and Kevin I-Kai Wang

SelfAct: Personalized Activity Recognition Based on Self-Supervised and Active Learning 375
Luca Arrotta, Gabriele Civitarese, and Claudio Bettini

A Novel Method for Wearable Activity Recognition with Feature Evolvable Streams 392
Yixiao Wang, Chunyu Hu, Hong Liu, Lei Lyu, and Lin Yuan

Let's Vibrate with Vibration: Augmenting Structural Engineering with Low-Cost Vibration Sensing 408
Masfiqur Rahaman, Md. Nazmul Hasan Sakib, Nafisa Islam, Saiful Islam Salim, Uday Kamal, Raihan Rasheed, and A. B. M. Alim Al Islam

Security Management

Research on Data Drift and Class Imbalance in Android Malware Detection ... 429
Zhen Liu, Ruoyu Wang, Bitao Peng, Changji Wang, and Qingqing Gan

Reputation-Based Dissemination of Trustworthy Information in VANETs 445
Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Antonio Virga

Reputation Systems for Supply Chains: The Challenge of Achieving Privacy Preservation 464
Lennart Bader, Jan Pennekamp, Emildeon Thevaraj, Maria Spiß, Salil S. Kanhere, and Klaus Wehrle

Data Management in Appendable-Block Blockchains: A Case Study for IT Life-Cycle Management 476
Rodrigo W. Silveira, Roben C. Lunardi, and Avelino F. Zorzo

Exploiting the Potential Anomaly Detection in Automobile Safety Data with Multi-type Neural Network 489
Quanlong Guan, Tian Zhang, Xiujie Huang, Yuansheng Zhong, Cuifeng Du, Changjiang Liu, Zhefu Li, Guanghui Zhang, Xiaofeng Wu, and Zhifei Duan

Urban/Mobile Crowdsensing

HAUM^3: A Height Aware Urban Map Matching Mechanism 505
Jie Tang, Sunjian Zheng, Bo Yu, and Shaoshan Liu

A Resource-Efficient Approach of GNSS Activation for Pedestrian Monitoring 526
Asiimwe Paddy Junior, Luis Enrique Díez, Alfonso Bahillo, and Odongo Steven Eyobu

Author Index 547

Contents – Part II

Urban/Mobile Crowdsensing

A Data-Driven Study on Pedestrian Walking Behaviour as Transitioning Different Spaces 3
Mengnan Cai and Xinling Shen

Standardizing Your Training Process for Human Activity Recognition Models – A Comprehensive Review in the Tunable Factors 15
Yiran Huang, Haibin Zhao, Yexu Zhou, Till Riedel, and Michael Beigl

MyPath: Accessible Route Generation Using Crowd-Sensed Surface Information 28
Thomas Nguyen, Md Fourkanul Islam, Rochishnu Banerjee, Hanna M. Noyce, Emily M. Olejniczak, Roger O. Smith, Md Osman Gani, and Vaskar Raychoudhury

Edge Computing

`Shepard`: Dynamic Placement of Microservices in the Edge-Cloud Continuum 43
Farhan Asghar, Tehreem Fatima, Junaid Haroon Siddiqui, Naveed Anwar Bhatti, and Muhammad Hamad Alizai

Minimizing Data Retrieval Delay in Edge Computing 63
Kolichala Rajashekar, Souradyuti Paul, Sushanta Karmakar, and Subhajit Sidhanta

Bringing the Edge Home: Edge Computing in the Era of Emerging WLANs . . . 86
Sampath Edirisinghe and Chathurika Ranaweera

Crowdsourcing, Platforms and Localization

A Novel Task Assignment Adjustment Method in Spatial-Temporal Crowdsourcing 105
Bingyi Sun, Jiaxu Cui, Hongtao Bai, and Yonggang Zhang

PPS: A Publish-Process-Subscribe Middleware for Predictive Supply Chains 125
Amir Jabbari, Gowri Ramachandran, Sidra Malik, and Raja Jurdak

Gestalt Perceptual Calibration for Multi-agent Collaborative Localization 145
Yan Zhang and Rong Xie

Activity Recognition and Prediction

Traffic Flow Prediction Using Uber Movement Data 167
Daniele Cenni and Qi Han

Fingertip Air-Writing with Ambient Light 179
Hao Liu, Hanting Ye, Xiangxie Zhang, Jie Yang, and Qing Wang

Global Optimal PnP Algorithm with Reverse Projection Error 192
Xuhao Jin

Proposal of Word Prediction Method for Gaze Swipe Text Entry 204
Hisanori Fujii and Ryo Katsuma

AI and Machine Learning

Integrating Social Environment in Machine Learning Model for Debiased Recommendation 219
Yihong Zhang, Lina Yao, and Takahiro Hara

A Novel Deep Learning Model for Smartphone-Based Human Activity Recognition 231
Nadia Agti, Lyazid Sabri, Okba Kazar, and Abdelghani Chibani

AIMSafe: EEG-Based Driver Behavior Understanding via Attention and Incremental Learning Mechanisms 244
Landu Jiang, Cheng Luo, Tao Gu, Kezhong Lu, and Dian Zhang

A Multi-task Learning Framework with Features Based on Behavioral Pattern Conversation 256
Bo Tang, Nan Wang, Jinbao Li, and Zhonghui Shen

Efficient Parcel Damage Detection via Faster R-CNN: A Deep Learning Approach for Logistical Parcels' Automated Inspection 268
Zhi Chen, Cuifeng Du, Quanlong Guan, Yuyu Zhou, Vichen Hoo, Xiujie Huang, Zhefu Li, Shuanghuan Lv, Xiaofeng Wu, and Xiaotian Zhuang

Fall Detection and Assessment Using Multitask Learning and Micro-sized LiDAR in Elderly Care 280
Shota Yamada, Hamada Rizk, Tatsuya Amano, and Hirozumi Yamaguchi

Online Dynamic Path Planner for UAVs 294
Manisha Wadhwa, Neelima Gupta, and Sanjay Madria

Mobile Edge and Fog Computing

Achieving Observability on Fog Computing with the Use of Open-Source Tools 319
Breno Costa, Abhik Banerjee, Prem Prakash Jayaraman, Leonardo R. Carvalho, João Bachiega Jr., and Aleteia Araujo

Adaptive QoS-Aware Task Offloading in Dynamic Mobile Edge Computing Environment 341
Jacob Don, Sajib Mistry, Redowan Mahmud, and Aneesh Krishna

Discretization-Based Ensemble Model for Robust Learning in IoT 353
Anahita Namvar, Chandra Thapa, and Salil S. Kanhere

Flute: Enabling a Battery-Free and Energy Harvesting Ecosystem for the Internet of Things 368
Van Vu Bui, Shuaibu Musa Adam, Brendan J. Mackenzie, Ashok Samraj Thangarajan, Mengyao Liu, Sam Michiels, Nelson Souto Rosa, Huynh Nguyen Bao Phuong, and Danny Hughes

Exploring Optimal Placement of Head-Based Hierarchical Marking Menus on Smartphones 381
Sabiha Tahsin Soha, Aritra Mazumder, Forhad Rabbi, Farida Chowdhury, and Khalad Hasan

SDN-DMQTT: SDN-Based Platform for Re-configurable MQTT Distributed Brokers Architecture 393
Fatma Hmissi and Sofiane Ouni

Mobile Augmented Reality and Applications for Mobile Computing

Quality Evaluation of Image Segmentation in Mobile Augmented Reality 415
Shneka Muthu Kumara Swamy and Qi Han

Reading the Mind's Eye: Detecting Trauma-Vulnerability in Individuals by Analyzing Attention Through Eye-Tracking 428
Taseen Mubassira, Mehedi Hasan, and Sadia Sharmin

GUI2DSVec: Detecting Visual Design Smells Based on Semantic Embedding of GUI Images and Components 445
Bo Yang and Shanping Li

Interaction Technologies

Mobile User Experience from the Lens of Project-Based Learning 461
Maria Spichkova

Usability Evaluation of NOW Interactions 473
Rosaline Barendregt, Barbara Wasson, and Martin Heitmann

QA Reasoning Enhancement Model Based on the Fusion of Dictionary and Hierarchical Directed Graph 485
Yuhang Bie, Meiling Liu, and Jiyun Zhou

AutoQuitous Workshop Session 1

Towards Distributed Control Under Deficient Communication with Multi-agent Reinforcement Learning 503
Arne Kummerow and Torben Weis

Deep Learning Based Radio Frequency Fingerprint Identification by Exploiting Spatial Stereoscopic Features 511
Shunliang Zhang, Jing Li, and Xiaolei Guo

Modeling Emergent Behaviour for Enhanced Autonomy in Cyber-Physical Systems 520
Matthias Schaffeld and Torben Weis

On-Device Soft Sensors: Real-Time Fluid Flow Estimation from Level Sensor Data 529
Tianheng Ling, Chao Qian, and Gregor Schiele

Cultural Influence on Autonomous Vehicles Acceptance 538
Chowdhury Shahriar Muzammel, Maria Spichkova, and James Harland

Author Index 549

Tracking and Detection

NOD: Lightweight Continuous Neighbor Discovery on Everyday Devices

Hsiao-Yuan Chen(✉), Evan King, and Christine Julien

The University of Texas at Austin, Austin, TX 78712, USA
{littlecircle0730,e.king,c.julien}@utexas.edu

Abstract. Continuous neighbor discovery (CND) allows wirelessly enabled devices to build a continuously updated picture of nearby devices. Such situational awareness is useful in many circumstances. While feasible on commodity devices like smartphones and wearables devices, CND is often not enabled due to practical limitations and concerns. Therefore, protocol implementations for CND have been mostly relegated to purpose-built sensor network deployments. However, the emergence of lightweight IoT devices and the need to rapidly discover what capabilities are around has renewed interest in CND on commodity devices. An important principle of CND protocols is their ability to be parameterized to achieve probabilistic guarantees regarding timeliness and assuredness of discovery events. However, transitioning from controlled deployments to commodity devices raises issues in providing guarantees related to CND. We explore two such concerns: (1) packet loss due to communication competition and (2) the time required to transition between sending and scanning for beacons. We propose a new CND protocol, NOD, that accounts for the impacts of these practical issues. We validate NOD through extensive simulations framed by real-world deployment settings and a proof-of-concept implementation for Android. NOD considers real-world application environments, allowing applications to identify optimal settings for guaranteeing probabilistic discovery in practical scenarios.

Keywords: Continuous Neighbor Discovery · Bluetooth Low Energy

1 Introduction

Continuous neighbor discovery (CND) entails continuously using on-device communication capabilities to detect ambient devices and collect information about what other devices and capabilities are in the surroundings. Protocols for energy-efficient CND are useful in many circumstances. For example, CND was fundamental to digital contact tracing [6] as demanded by the COVID-19 pandemic. CND has also been suggested in IoT applications, including those for smart cities [16]. Historically, CND is primarily implemented in wireless sensor networks [1,2,4,12,14], but with emerging lightweight and energy-efficient

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 3–25, 2024.
https://doi.org/10.1007/978-3-031-63989-0_1

technologies on mobile devices, CND has expanded to include even ordinary smartphones, broadening the reach of these protocols. In particular, Bluetooth Low Energy has become a technology of choice for implementing CND [8,10].

Generically, CND protocols consist of a schedule that alternates between sending beacons to announce presence and listening for others' beacons. When a CND protocol is designed on a purpose-built sensor network, many practical concerns can be assumed or designed away. However, making CND work on commodity sensing devices and smartphones requires considering practical constraints that arise when low-level control of the radio states is not possible and the communication medium is shared. Protocols have started to consider these constraints, for instance BLEnd [8] considers the significant impact of packet collisions on the probability of discovering other nearby devices. However, BLEnd only considers collisions that occur with other devices as they also perform continuous neighbor discovery. Kindt et al. consider some properties of real radios that impact the correctness of discovery [11] but fail to consider the performance of neighbor discovery in very dense deployments where collisions are problematic.

As a result, diverse practical considerations remain unaddressed in the application of CND on everyday multi-purpose networks like those that connect smartphones with each other and with wirelessly available resources in their environments. We focus on two such practical considerations, which we refer to as *packet loss* and *warmup interval*. The first concern is caused largely by collisions that occur between wirelessly transmitting devices, whether the collisions happen because two neighbor discovery beacons collide or because a neighbor discovery beacon is impacted by other wireless communication in a busy wireless environment. In noisy environments like cities, homes, and offices, packets may be lost seemingly randomly. The second concern, which we term *warmup*, arises because CND protocols that rely on strict timing requirements of listening and beaconing neglect the fact that the wireless radio cannot immediately switch modes. Instead, there is a period of time that precedes the start of both listening and beaconing before the device can successfully receive or send data. This phenomenon has been recognized by others as well; it has also been referred to as *turnaround time* [11].

As CND shifts from MCU-class devices in sensor networks to smartphones, new protocols are necessary. We leverage existing CND design work, identify limitations when applied to smartphones, and develop new models and protocols for this context. Our research reveals significant impacts of packet loss and radio command timing on traditional CND protocols. Despite these protocols not being tailored for smartphones, we quantify these impacts on their performance. To address these issues, we introduce **Nod**[1], a CND protocol considering packet loss and warmup intervals. We construct an analytical model to evaluate **Nod** in dense, varied networks. Compared to applying traditional CND protocols on smartphones, **Nod** notably enhances CND performance, especially in discovery guarantees. Concretely, the contributions of this paper are:

[1] The name **Nod** is an homage to a subtle head-tilt upon meeting, a greeting that can be lightweight and ephemeral.

- We quantify the impact that practical considerations of packet loss and warmup have on discovery performance in existing continuous neighbor discovery protocols.
- We propose a new protocol and associated analytical model for neighbor discovery that addresses the real-world issues of packet loss and warm-up interval. The new model builds on existing work to provide a guaranteed discovery probability within a specified discovery latency.
- We demonstrate the improvements possible with a protocol that implements our updated model and quantify tradeoffs relative to energy consumption.

The remainder of this paper is organized as follows. Section 2 provides motivation and related work. In Sect. 3, we empirically demonstrate the impact of packet loss and warmup interval in existing CND protocols. Section 4 presents **Nod**, a new protocol and associated analytical model of CND that accounts for packet loss and warmup intervals, and Sect. 5 shows the performance of **Nod**, given a model that considers real-world behavior.

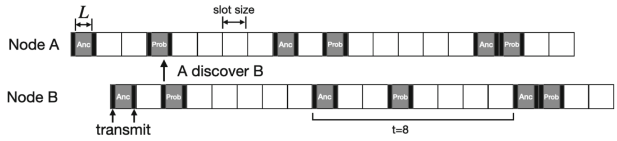
2 Background and Related Work

Given the interest in CND in wireless sensor networks, diverse CND protocols have been proposed. Protocols can be generally separated into two groups: slotted protocols and slotless ones. Both aim to optimize neighbor discovery performance against energy consumption by defining a *schedule* that defines when a participating node should listen and when it should beacon. Slotted protocols break time into equal sized chunks called *slots* and schedule periods of advertising and scanning based on these slots. Slotless protocols, while maintaining a repeating fixed-length schedule, offer more flexibility. In this section, we describe a few protocols in each class; we use three of these protocols in our motivating empirical study in Sect. 3.

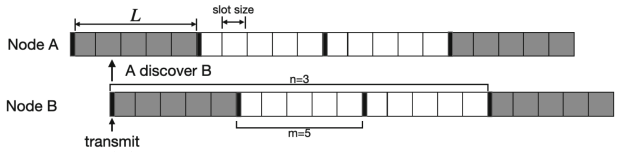
Generically, CND protocols try to balance three concerns:

λ: a discovery latency, i.e., the time between when a neighboring node appears and when it is discovered;
ρ_Λ: a discovery probability, i.e., the probability of discovering a neighbor within some target discovery latency Λ; and
η: the duty cycle of neighbor discovery, i.e., the fraction of time a device's radio actively participates in CND.

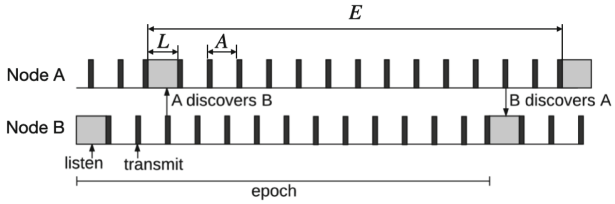
Slotted Neighbor Discovery Protocols. In a slotted protocol, the length of a slot is often determined by practical aspects of the radio. The protocols differ most substantially on how they choose to use slots (i.e., during which slots the protocol is *active* and which slots it is *idle*) and how they schedule beaconing and listening activities within active slots. Some of the earliest protocols achieved good performance in choosing active slots probabilistically [13,17]. The analytical models of these protocols could then assure probabilistic discovery by examining the probability of overlap in two devices' active slots.



(a) A Searchlight schedule with $t = 8$. Anchor slots are labeled as 'Anc' and Probe slots are labeled as 'Prob'.



(b) A Nihao schedule with $m = 5$ and $n = 3$



(c) An F-BLEnd schedule

Fig. 1. Example schedules for existing protocols

These protocols often abstract the details of activities *internal* to an active slot, assuming that overlapping active slots between two nodes lead to discovery. Many protocols overlook the impact of collisions: if three nodes have overlapping active slots, they are assumed to discover each other, despite practical radio limitations. The Birthday protocol [13] is an exception; its model does consider potential discovery event losses due to multiple aligned active beaconing slots. In our analysis in Sect. 3, we use two slotted protocols: Searchlight [1] and Nihao [14], selected for their recognition and distinct approaches to allocating time for listening versus beaconing.

Figure 1(a) illustrates a Searchlight [1] schedule. It features two active slots per period (with period length t slots), placing beacons at each active slot's start and end, with a listening interval in between. This structure ensures a discovery chance when active slots overlap. Searchlight uses anchor slots at each period's start and probe slots in a random order within the first $t/2$ non-anchor slots, ensuring eventual overlap between one node's anchor slot and another's probe slot. For instance, in Fig. 1(a), Node A follows the pattern {3, 2, 1, ...}, and Node B {2, 3, 1, ...}.

Figure 1(b) shows the Nihao schedule [14]. Nihao is characterized by two parameters, m and n and follows a repeating pattern every $m * n$ slots, which we refer to as an *epoch*. During each epoch, each node schedules a beacon at the beginning of every m slots, and then each node listens for a duration of m slots after their first beacon in the epoch. This pattern repeats for the duration of the

Nihao schedule. This schedule ensures nodes discover each other as long as the start times of two nodes do not align with each other. Specifically, if the start time of one node is at least b (the length of a beacon) before or after another node's start time, then the two nodes have a chance to discover each other.

Slotless Neighbor Discovery Protocols. In contrast to slotted protocols, slotless protocols are not driven by slots of fixed length. Instead, they define a repeating schedule of beaconing and listening [8,10]. Some protocols schedule beacons independently of listening, so a beacon may be scheduled in the middle of a listen, in which case the node stops listening, sends the beacon, then starts listening again [10]. In general, the key parameters of a slotless protocol are the frequencies of listen and beacon activities and the length of a schedule epoch.

Because slotless protocols have similar timing constraints relative to our practical considerations, we choose a single such protocol for study: BLEnd [8]. As shown in Fig. 1(c), an *epoch* in BLEnd starts with a period of listening, immediately followed by a period of advertising. The full BLEnd protocol advertises for the remainder of the epoch, as depicted in Fig. 1(c). The lengths of the listen period (L) and the advertising interval, (A) (i.e., the length of time between the start time of two successive beacons) are designed to ensure the protocol's discovery latency and probability targets. BLEnd can be configured by defining the length of an epoch, E (which can be of an arbitrary duration because BLEnd is slotless), and the length of the advertising interval, A. The length of a listen period is derived from the advertising interval. Adjusting E and A in BLEnd can respond to different application contexts, including the expected number of nearby nodes or to achieve different goals in terms of ρ_Λ.

Practical Considerations for CND. Several aspects of wireless communication affect CND performance; we consider two facets, both of which are important in making CND compatible with real-world devices. The first challenge of these, *packet loss*, refers to the fact that beacons may be not received, even after accounting for collisions with other neighbor discovery beacons. These packet losses may be due to interference with other communication in the environment, because of idiosyncrasies of the mobile operating systems, or several other causes. For instance, in [15], the authors demonstrate that, even without additional BLE devices competing for the wireless medium, there is still a packet loss rate of more than 13% for off-the-shelf smartphones. Additional experiments demonstrate packet loss rates that are between 4.55% to 39.47% [5].

The second challenge, *warmup interval*, relates to the fact that just because an application instructs a radio to change from one modality to another does not mean that the change happens instantaneously. On one hand, this is physically impossible, even when the application executes very close to the radio layer [11]. On the other hand, CND protocols ask radios to be in one of three states at any given time: beaconing, listening, or idle. The transition from any one of these states to any of the other three takes a non-negligible amount of time as a result of physical, application, and operating system constraints, during which the radio is not able to do anything (it can neither receive beacons nor send them). We capture two distinct *warmup* intervals: the time it takes a device

to transition into listening (ψ_L) and the time it takes a device to transition into beaconing (ψ_A). We conducted measurements to empirically determine the warmup intervals of various devices, and the average warmup of 5 trials are presented in Table 1. The results demonstrate that warmup intervals range up to more than one hundred milliseconds. Compared to a common CND epoch length of a few seconds, the warmup interval is non-negligible and can dramatically impact discovery probability, as we see in the next section.

Table 1. Measured warmup period on different devices. Each value is averaged over 5 trials and shown with 95% confidence intervals.

	Google Pixel 3	Samsung S7	Huawei N29
Advertising warmup (ψ_A) (ms)	257.3 ± 23.5	4.23 ± 23.5	264.7 ± 10.6
Scanning warmup (ψ_L) (ms)	42.8 ± 5.4	149 ± 4.9	49 ± 1.5

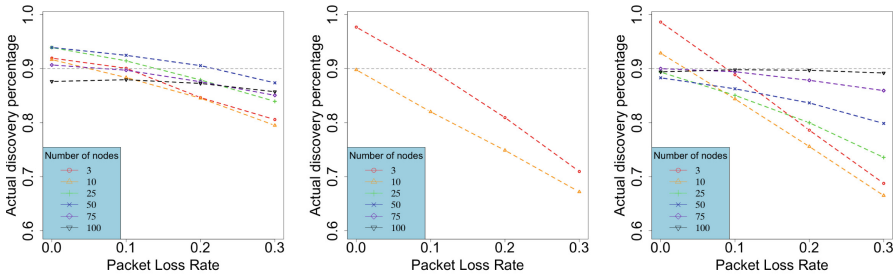
3 Neighbor Discovery in the Wild

While existing neighbor discovery protocols consider some realistic implications on performance, key challenges remain unaddressed. In particular, while the impact of collisions on discovery has been noted [8,9], neighbor discovery protocols have failed to address the potential impacts of random *packet loss* and communication *warmup intervals*. We empirically demonstrate the impact of these two concerns on discovery probability (ρ_Λ) and discovery latency (λ)—the two hard constraints of many CND protocols—on three neighbor discovery protocols.

To support our empirical analysis and evaluations, we developed a Java-based discrete event simulator for CND. It allows specifying various neighbor discovery schedules such as Searchlight [1], Nihao [14], or BLEnd [8], and simulates random schedules for nodes within range. The simulator[2] emulates BLE radio behavior typical in Android devices, assuming a 1.1 ms beacon length based on empirical data and the BLE standard [18]. While numerous network simulators exist, we opted to implement this simple event simulator because we are not interested in node mobility or the behavior of the network stack, which are the purposes of existing simulators. Instead, we are interested in probabilistic overlap of discovery events. Specifically, our simulator simply analyzes generic neighbor discovery schedules for a set of nodes that are all assumed to be within range of each other simultaneously. It accounts for collisions among neighbor discovery beacons, and it can also be adjusted to inject both random packet losses and arbitrary warmup intervals.

To assess the impact of packet loss, we studied CND protocol behavior with packet loss rates from 0% (no loss) to 30%, in 10% increments, based on the

[2] https://github.com/UT-MPC/CND_simulator.



(a) Searchlight (b) Nihao (c) BLEnd

Fig. 2. Impacts of packet loss on discovery probability. All protocols are configured to attempt to achieve 90% probability of discovery within 10 s. Random packet losses range from 0% of packets to 30% of packets (x-axis).

prevalence of such rates in smartphone communications [5]. For communication warmup intervals, including radio and OS/application delays, we used intervals of 0 (no warmup), 50 ms, 100 ms, 150 ms, and 200 ms, reflecting observed ranges (50–150 ms) with slight adjustments to 0–200 ms 1. We analyzed these factors across different node counts (3, 10, 50, 100, 150, 200) to represent various social scenarios, from home settings (3 nodes) to larger gatherings like concerts or parties (up to 200 nodes).

We set a target discovery latency $\Lambda = 10s$ and a discovery probability $\rho_\Lambda = 0.9$ to ensure high confidence in data retrieval while minimizing discovery time. This is particularly relevant in applications like opportunistic recommendation exchanges in smart cities [7], where quick device discovery (under 10 s) on transient interactions, like on a subway or street corner, is crucial for completing further steps like authentication and data exchange. Because we assume implementations that will function on commodity devices, we seek compatibility with Bluetooth Low Energy (BLE). Because the Android operating system (similar to other mobile operating systems) limits on advertising intervals for BLE, we adopt Android's *balanced* mode interval of 250 ms, balancing energy cost and delay. Consequently, we set Searchlight's slot size to 251.1 ms, accommodating two full beacons per slot. For Nihao, with beacons only at slot starts, the slot length is 250 ms. In BLEnd, we choose $A = 250$ ms.

Impacts of Packet Loss. Network packets can be lost for various reasons. While some losses, like collisions with other neighbor discovery beacons, can be modeled, others are random. Thus, a listener might not receive every beacon, even if it is actively scanning and within range. To evaluate the effect of packet loss on neighbor discovery in our protocols, we introduced random beacon loss into the simulation, observing protocol behavior amidst other devices also engaged in discovery. These losses are additional to those from collisions, which the simulator already models.

Figure 2 shows how CND protocols are impacted by random packet losses. First, on average, discovery probability decreases as packet loss rate (the x-axis)

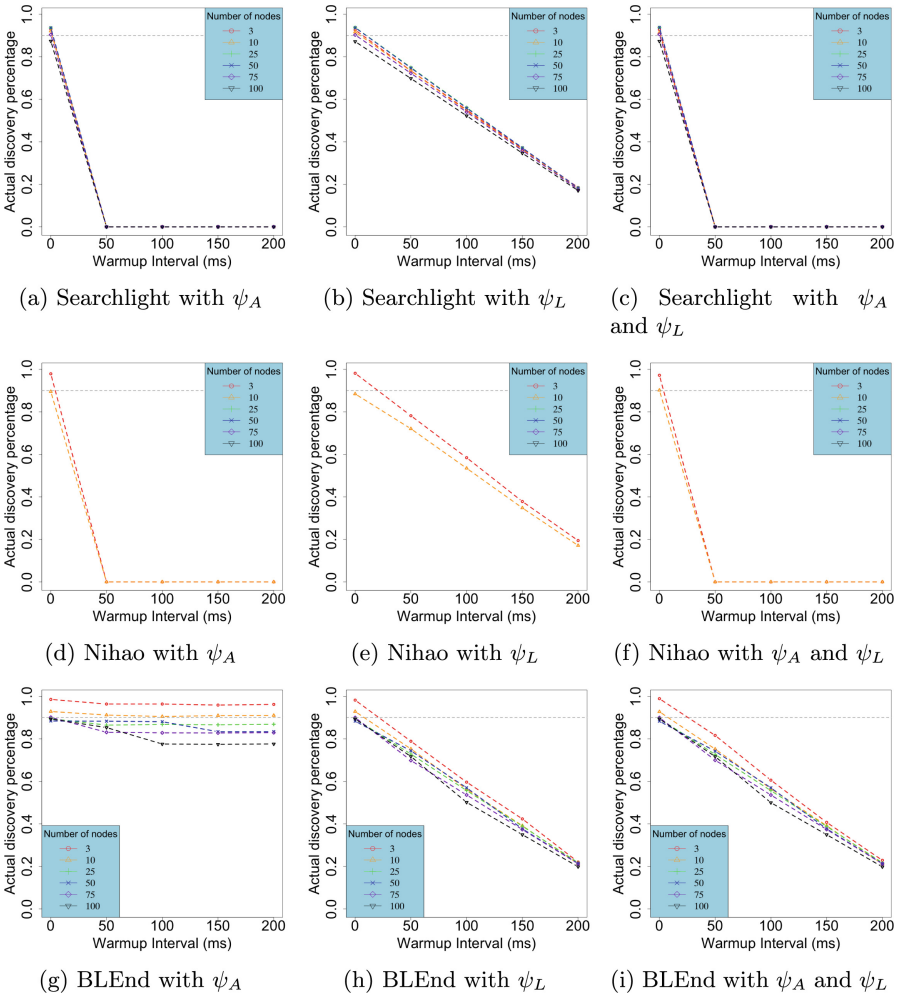
increases for all three protocols. Searchlight's performance is relatively consistent across all densities of nodes; BLEnd performs *better* as the number of surrounding nodes increases. This is due to the fact that BLEnd are tuned to *talk more* so they are more sensitive to collisions between neighbor discovery beacons, which is even more impactful with larger numbers of nodes. As beacons are randomly lost, collisions go down, which actually improves performance. As Fig. 2 shows, not explicitly considering packet loss in tuning a protocol for CND misses the mark for the target performance measures.

One observation is the lack of data for Nihao with more than 10 nodes. This is due to Nihao's inability to reach the desired 0.9 discovery probability, considering the detrimental effects of beacon collisions. Nihao and Searchlight's fixed beacon positions at each slot's start and end lead to similar scheduling across epochs. While Searchlight uses a probe slot to mitigate this, Nihao's beacons remain consistently positioned, heightening collision risks. With more nodes, collisions intensify, hindering Nihao's ability to meet the target discovery probability.

Impacts of Warmup Interval. The second real-world device characteristic we examine is the *warmup interval*, the time required for a device to transition to a transmitting or listening state, encompassing delays from the physical radio, operating system, and application-level. Typically, CND protocols like BLEnd [8], Searchlight [1], and Nihao [14] assume zero transition time, with precise scheduling of beacons. Kindt et al. [11] have recognized the impact of *turnaround time* on continuous neighbor discovery. The Searchlight authors also noted this and introduced a pre-slot to start service earlier as a workaround.

Figure 3 illustrates the impact of increasing warmup intervals on the protocols in three scenarios. First, we explore an advertising warmup interval (ψ_A) with values $0, 50, 100, 150, 200$. Second, we consider a scanning warmup interval (ψ_L) with the same values. Lastly, we assess the combined effects of ψ_A and ψ_L, using five pairs: (0, 0), (50, 50), (100, 100), (150, 150), (200, 200). Nodes cannot scan or advertise during warm-up intervals, so discovery probability inversely relates to warm-up length; listening capability diminishes as the warm-up duration increases relative to the original listening period. Searchlight and Nihao, scheduling beacons at slot edges, fail to send beacons with any advertising warmup, leading to a drop to 0 discovery probability when $\psi_A \neq 0$ in Fig. 3(a), (c), (d), (f). BLEnd, advertising continuously post-scanning, circumvents this issue.

Figure 4 shows the impact of scanning warm-up and packet loss on discovery probability and lifetime for 50 nodes using BLEnd. We estimated energy consumption assuming CND as the sole activity, based on current draw (including warm-up) and power data from [3]. While power profiles vary by device, relative magnitudes are expected to follow similar trends. As Fig. 4(a) demonstrates, increasing packet loss or warm-up severity decreases discovery probability. The figure illustrates how these two factors can result in a notable reduction in the overall discovery probability, emphasizing the importance of considering both practical issues when designing and analyzing beacon scheduling protocols. Under the same conditions, a node's lifetime actually *increases*, indicating that



(a) Searchlight with ψ_A (b) Searchlight with ψ_L (c) Searchlight with ψ_A and ψ_L

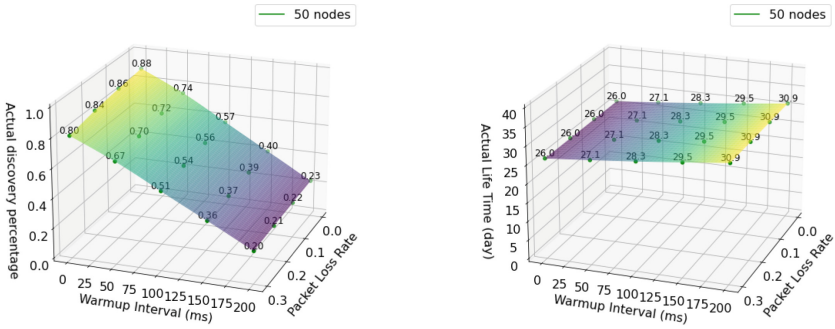
(d) Nihao with ψ_A (e) Nihao with ψ_L (f) Nihao with ψ_A and ψ_L

(g) BLEnd with ψ_A (h) BLEnd with ψ_L (i) BLEnd with ψ_A and ψ_L

Fig. 3. Impacts of different warm-up intervals on discovery probability. All protocols are configured to attempt to achieve 90% probability of discovery within 10 s. Warmup intervals range from 0 to 200 ms (x-axis).

the protocol is doing less work than the model anticipates (i.e., because of the warmup intervals, the radio is spending less time actively scanning or receiving).

Continuous neighbor discovery protocols designed for commodity smartphones must consider these practical impacts, whereas protocols tuned for MCU-class devices with tight timing constraints can get away with ignoring them. We next introduce a revised CND model and protocol that generates schedules in a way that is sensitive to random packet loss and arbitrary warm-up intervals.



(a) Discovery Probability (b) Lifetime

Fig. 4. Impact on discovery probability and lifetime of BLEnd with consideration in the presence of both packet loss and scanning warmup for 50 nodes. All protocols are configured to attempt to achieve 90% probability of discovery. Warm-up intervals range from 0 ms to 200 ms for both ψ_A and ψ_L. Packet loss rates range from 0 to 0.3.

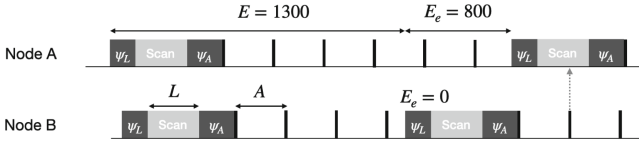


Fig. 5. Two nodes with different **Nod** schedules.

4 Solving Discovery with Real-World Practicalities

We next present a new model for continuous neighbor discovery that addresses the packet loss and warmup interval issues encountered in practical scenarios. Based on the results in Sect. 3, we adopt a slotless approach, as its less rigid and more flexible scheduling is better able to overcome these practicalities. We first introduce the basics of probabilistic slotless discovery and then use this as a basis for our new model that accounts for packet losses and warmup intervals.

4.1 Nod: CND with Constrained Schedules

To address practical issues such as packet loss and warmup intervals, we propose a new slotless protocol called **Nod**, as depicted in Fig. 5.

In **Nod**, a node follows a repeating pattern of epochs, each varying in duration, expressed as $E_r + E_e$. Here, E_r is a constant (*root* epoch), and E_e is an *extra* part, randomly chosen from $[0, E_r]$. This design makes each **Nod** epoch's length randomly and independently fall in $[E_r, 2E_r]$. An epoch begins with a listening interval L and continues with advertising at a fixed interval A.

Notice that the schedules depicted in Fig. 5 account for ψ_A and ψ_L so that the design can account for their impacts on discovery probability. We discuss three of these design decisions in detail in the following. In particular, we first

examine how ψ_L impacts the necessary settings for L. We then examine the effect of decoupling the length of an epoch and A, the advertising interval. Finally, we discuss the rationale of choosing E_e to be in the range $[0, E_r]$ to probabilistically ensure that two nodes whose schedules are synchronized in one epoch are unlikely to be synchronized in the next epoch, boosting the probability of discovery.

Ensuring a Sufficient Functional Listening Interval. Figure 6(a) shows a schedule from a naïve slotless protocol with a fixed epoch length and no consideration of warmup intervals. To ensure that two nodes, in isolation, are guaranteed to discover one another if one's listen interval overlaps the other's advertising interval, the schedule needs to ensure that L is sufficiently long to capture an entire beacon regardless of the shift between the two schedules. To ensure this, one must make $L \geq A + b$ [8] (as shown in Fig. 6(a))[3]. However, existing CND protocols assume a zero time warmup interval to transition from an idle state to a scanning one. Table 1 shows this is not the case; indeed the transition time is in the same order as commonly chosen listening intervals. Therefore, as Fig. 6(b) shows, a significant fraction of the listen interval can actually be *non-functional*, which is the direct cause of the steep declines we see in Figs. 3(b), (e), and (h). Fortunately, mitigating this impact is quite simple—we simply extend the size of the listening interval to include ψ_L as shown in Fig. 6(c), i.e., $L = A + b + \psi_L$, ensuring that the functional listening period remains at $A + b$. Effectively, we extend the listen interval (L) to ensure that an epoch's active scanning time is guaranteed to capture at least one of any pair of successive beacons of any neighboring node, reconfirming a design tenet of the original schedule.

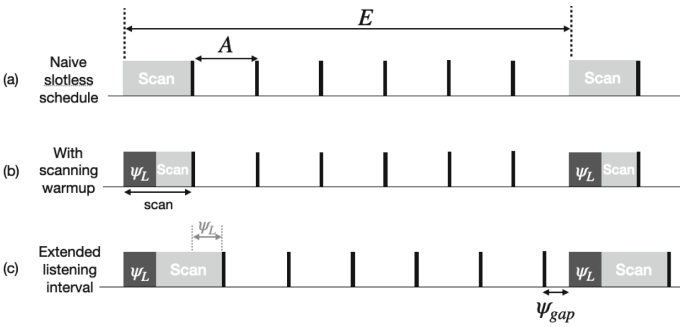


Fig. 6. Illustration of schedules in different scenarios: (a) original schedule (without warmup); (b) original schedule (with a scanning warmup); (c) **NOD** extends the listen interval (L) to restore the full functional scanning period

[3] In fact, when employing BLE radios, $L \geq A + b + s$, where s is some random slack added to the advertising interval and outside of the control of the application [18]. We account for s in our models and implementation but omit it here for simplicity.

That Last Beacon. In Figs. 5 and 6, the independence of epoch length and advertising interval leads to a fractional space, ψ_{gap}, at each epoch's end. One proposed solution is to position a beacon just before the listen interval [8], but warmup intervals hinder this approach. Firstly, the scanning can't immediately succeed the last beacon due to ψ_L. Secondly, the irregular timing between the hypothetical 'last' beacon (dashed in Fig. 7) and its predecessor necessitates an additional ψ_A, but ψ_{gap} might be shorter than ψ_A itself. Hence, rather than trying to 'fix' the schedule, we acknowledge the diminished discovery likelihood due to ψ_{gap}, which, along with ψ_A and ψ_L, constitutes another non-functional part of the discovery schedule. In particular, given that the end of the epoch is equally likely to fall anywhere within some advertising interval, the resulting expected value of ψ_{gap} is $\frac{A}{2}$.



Fig. 7. Illustration of ψ_{gap} and its potential incompatibility with ψ_A

Overlapping Epoch Start Times. With fixed epoch lengths, if one node chooses a start time that is nearly the same as another, the two nodes will never discover one another, effectively because they will always be aligned to be scanning at the same time. However, when we consider how the warmup intervals extend the fraction of an epoch that is non-functional, this potential alignment problem can be devastating to any effort to achieve high discovery probabilities.

Even with the extension of the listening interval, a node still has a lower chance of being discovered compared to when no warmup interval is present. As illustrated in Fig. 8, with the presence of a warmup interval, an advertiser must initiate its schedule either $\psi_L + \psi_{gap}$ after the scanner's start time or ψ_A before the scanner's start time, in order for one of its beacons to fall into the scanner's listening interval. That is, if an advertising neighbor's schedule start time is within the red block of the scanner's schedule in Fig. 8, discovery will not happen.

To tackle the consistent shift issue between nodes, we introduce irregular epoch lengths in **Nod**. We construct an epoch from two parts: E_r, a shared *root* interval, and E_e, a random addition chosen from $[0, E_r]$. Thus, an epoch's length, $E = E_r + E_e$, falls between $[E_r, 2E_r]$. We cap E_e at E_r to optimize the chance for a node to move out of the non-functional area in subsequent epochs. Selecting a smaller maximum for E_e might allow nodes to shift out of the non-functional area, but with low probability and potential for quick re-entry. A larger maximum causes epochs to wrap around, proving ineffective. Thus, we set E_e's maximum equal to E_r. This gives nodes an equal opportunity to shift within

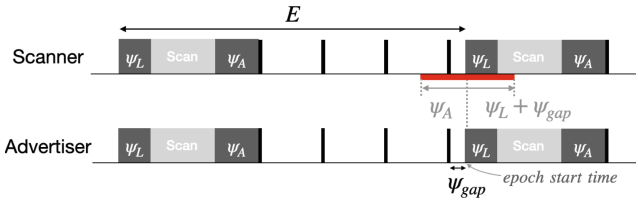


Fig. 8. Illustration of the non-functional area caused by the warmup interval and the removal of the last beacon right before next epoch.

the E_r range. The random epoch extension greatly increases discovery rates, key to **Nod**'s success in high discovery probabilities and handling warm-up intervals and packet loss. Table 2 details these parameters.

Table 2. Parameters used to model **Nod**.

Foundational Schedule Parameters	
N	number of nodes in the collision domain
Λ	target discovery latency (ms)
b	length (in time) of a beacon (ms)
E	epoch length (ms)
A	advertising interval (ms)
L	listening interval length (ms) ($L = A + s + b$)
γ	number of potentially colliding beacons
Parameters to Capture Real World Practicalities	
ℓ	packet loss rate
ψ_L	scanning warmup interval length (ms)
ψ_A	advertising warmup interval length (ms)
ψ_{gap}	expected value of gap between the last beacon and the next epoch (ms)
ψ	non-functional time in an epoch (ms), $\psi = \psi_L + \psi_A + \psi_{gap}$
E_{avg}	expected value of the length of a **Nod** epoch (ms)

4.2 Analytical Model for Nod

In this section, we propose a model to analyze the expected discovery probability using the **Nod** schedule described in the previous section, given the presence of packet loss and warm-up intervals. With the model, we can identify the optimal configuration, i.e., the schedule parameter E_r, that achieves the desired discovery probability while minimizing energy consumption. We base our model on the first few steps of the BLEnd model [8] which captures the impact of beacon collisions on discovery probability. From this basic model, we continue to consider the impacts of warmup intervals and random packet losses.

The Basic Model of Probabilistic CND. The model's goal is to define P_d (a "probability of discovery"), expressed in terms of N, the number of expected neighbors, Λ, the target discovery latency, and b, the length of a beacon (1.1 ms for a 31 byte BLE beacon [18]). The optimal scheduling, aimed at achieving Λ with minimal energy, is determined by two parameters: E_r, the root epoch length, and A, the advertising interval. A is set to 250 ms due to constraints from the Android operating system; the main task is to find the optimal E_r.

The probability of discovery, P_d, is also dependent on a value γ. γ represents the number of potential colliding beacons from neighboring nodes. While a simple assumption might suggest $\gamma = N - 2$ (excluding the sender and receiver), in reality, γ tends to be slightly higher. This is due to the positioning of beacons within nodes' schedules, especially when a target beacon occurs at the start or end of a listening interval, potentially overlapping with other beacons.

Given the above, we can use a relatively straightforward application of the birthday paradox to capture collisions and express the probability that a given node discovers a neighboring node as [8]:

$$P_d = \left(1 - \frac{2b}{A}\right)^{\gamma} \tag{1}$$

We next provide some brief intuition for Eq. 1. The design of $L = A + b$ ensures that when a neighboring node advertises during an epoch, at least one of its beacons will fall within the listener's functional interval. This basic model assumes that this beacon is guaranteed to be received by the listener unless it collides with another node's beacon. For modeling purposes, we treat these neighbor discovery collisions separately from more random packet losses, dealt with below. A collision occurs when beacons overlap. As nodes independently choose their schedule start times, there are $2b$ potential times leading to a collision with a neighbor's beacon, as illustrated in Fig. 9. Considering γ possible colliding beacons with independent start times, Eq. 1 gives the likelihood that a beacon is received without collision.

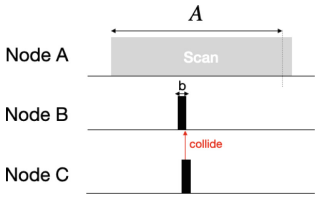


Fig. 9. The effective starting range of a beacon.

To ensure the possibility of discovery from schedule overlap, we must ensure that $E \leq \Lambda$. Otherwise it would be possible that a receiver's listen interval simply does not occur with Λ. Therefore, to express the probability that a node is discovered, not within a single epoch, but within the target discovery latency,

Λ, we need to consider the probability that the discovery event occurs in one of a set of contiguous epochs, plus a fraction of an epoch. Given that $\Lambda = kE + f$, where $f < E$, the probability of discovery within Λ, i.e., $P_{d,\Lambda}$ is [8]:

$$P_{d,\Lambda} = P_{d,kE} + P_{d,f} \tag{2}$$

where

$$P_{d,kE} = 1 - (1 - P_d)^k \tag{3}$$

and

$$P_{d,f} = (1 - P_d)^k \left(\frac{\Lambda - kE}{E} \right) P_d \tag{4}$$

Ultimately, the goal is to find a solution to Eq. 2 (i.e., values of E and A) that achieves the target discovery probability Λ with minimal energy consumption [8]. We use the basic model in Eq. 2 as a jumping off point for constructing schedules that account for both random packet losses and warmup intervals and modeling the probability of discovery of these new schedules.

Discovery Probability with Warmup Intervals. The model above suffices for schedules like those in Fig. 6(a). We next extend this model to consider the impacts of the schedule designs that include warmup intervals (ψ_L and ψ_A), the incomplete advertising interval at the end of an epoch (ψ_{gap}), and the random extension to the length of an epoch (E_e). These changes have two significant impacts. Both rely on the observation that ψ_L, ψ_A, and ψ_{gap} compound to create the non-functional block of time depicted in red in Fig. 8. Any neighbor node that chooses a schedule start time within this block of time will not have a beacon scheduled to overlap a receiver's functional listening interval. As a result, the number of potentially colliding beacons is reduced by a factor of $\frac{\psi}{E_{avg}}$ relative to the basic model, where E_{avg} refers to the expected value of the length of an epoch in **Nod**. Given our choice of E_e's range to be $[E_r, 2E_r]$, $E_{avg} = 1.5E_r$. Reflecting this in our model of discovery, we define γ_w, the number of potentially colliding beacons when one considers warmup interval impacts as:

$$\gamma_\psi = \left(1 - \frac{\psi}{E_{avg}} \right) \gamma \tag{5}$$

where γ is as defined above and in [8] and $\psi = \psi_L + \psi_A + \psi_{gap}$.

Second, and conversely, the probability of a node being discovered by a listener is *reduced* because beacons are not sent in the non-functional window represented by ψ. To account for these lost discoveries, we decrease the probability of discovering a node in a listening interval by the same factor, $\frac{\psi}{E_{avg}}$, effectively replacing Eq. 1 with:

$$P_{d,\psi} = \left(1 - \frac{\psi}{E_{avg}} \right) \left(1 - \frac{2b}{A} \right)^{\gamma_\psi} \tag{6}$$

We can use Eq. 6 to compute $P_{d,kE,\psi}$ and $P_{d,f,\psi}$ in a manner similar to Eqs. 3 and 4, which we combine as in Eq. 2 to express $P_{d,\Lambda,\psi}$, the probability any given

node discovers any other given node within the window Λ, when one assumes warmup period impacts whose non-functional periods sum to ψ. Because in **Nod** E varies between E_r and $2E_r$, we must refine the constraint on the relationship between E and Λ and require that $2E_r \leq \Lambda$ for the same reasons as argued above (i.e., we must ensure that at least one listen interval falls within Λ.

Discovery Probability with Packet Loss. With the model of warmup intervals demonstrated, we next turn our attention to modeling the impact of random packet losses on the effectiveness of neighbor discovery. We model random packet losses uniformly: any beacon may experience a loss with a rate denoted as $0 \leq \ell \leq 1$. These losses have the same two notable impacts as above: the number of potentially colliding beacons diminishes by a factor of ℓ and the probability of discovering a node is reduced by a factor of ℓ due to the possibility of the target beacon itself being lost. Building on our work in the previous section, we account for the former by defining $\gamma_{\psi,\ell}$:

$$\gamma_{\psi,\ell} = (1-\ell)\left(1 - \frac{\psi}{E_{avg}}\right)\gamma \tag{7}$$

Since the impact of packet loss and warmup interval are independent of each other, we can aggregate their effects to derive the combined model. As a result, our final model expressing $P_{d,\Lambda,\psi,\ell}$, the probability of discovery in the presence of warmup intervals that aggregate to ψ and a packet loss rate of ℓ, is:

$$P_{d,\Lambda,\psi,\ell} = 1 - (1 - P_{d,\psi,\ell})^k + P_{d,f,\psi,\ell} \tag{8}$$

where

$$P_{d,f,\psi,\ell} = (1 - P_{d,\psi,\ell})^k \left(\frac{\Lambda - kE_{avg}}{E_{avg}}\right) P_{d,\psi,\ell} \tag{9}$$

and for the latter by defining $P_{d,\psi,\ell}$:

$$P_{d,\psi,\ell} = (1-\ell)\left(1 - \frac{\psi}{E_{avg}}\right)\left(1 - \frac{2b}{A}\right)^{\gamma_{\psi,\ell}} \tag{10}$$

5 Evaluation

To validate our approach, we used both simulations and a small scale real-world deployment as a proof-of-concept.

5.1 Comparative Simulations

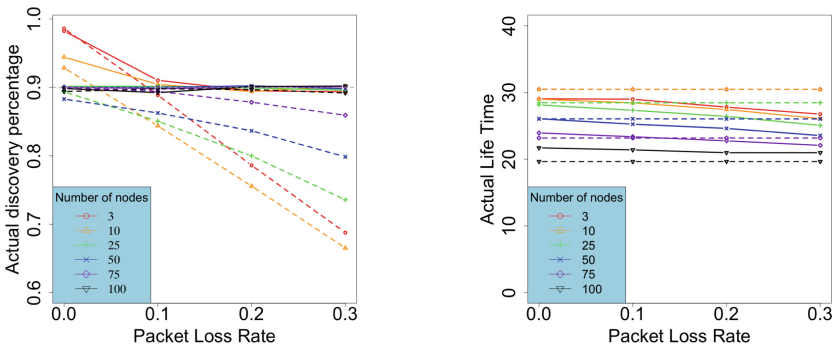
In our simulations, we compare the model of **Nod** with the BLEnd model. We compare against BLEnd for two reasons: (1) it was the best performing of the existing models out of the box (Sect. 3) and (2) our resulting schedule is also slotless and similar in structure to the BLEnd schedule. We used the same

discrete event simulator here as we used in Sect. 3. We also use the same settings for packet loss and warmup intervals, with the same application-motivated justifications as presented in Sect. 3.

For BLEnd, the published model [8] determined optimal parameters (E and A) for each setup. For **NOD**, we utilized the model from Sect. 4, running each configuration 500 times. Consistent with Sect. 3, the target discovery latency (Λ) was set to 10 s and the target discovery probability (ρ_Λ) to 0.9. Lifetime was computed using power profiles from [3]. Models were evaluated based on average discovery probability and lifetime (inverse of energy consumption).

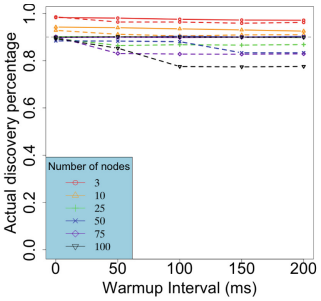
Random Packet Loss. Figure 10(a) shows how **NOD** performs in the presence of increasing rates of random packet loss. The figure compares **NOD** (solid lines) to BLEnd (dashed lines, the same as depicted in Fig. 2(c)). Notably, **NOD** shows a significant improvement in the discovery probability, meeting the target (0.9) discovery rate almost uniformly. In particular, the discovery rate is extremely stable across increasing numbers of nodes and increasing packet loss rate.

Of course, this resilience to packet loss comes at a cost, particularly in terms of energy consumption. As depicted in Figs. 10(b), it is evident that the system's lifetime decreases as the packet loss rate increases. This correlation arises due to the fact that higher packet loss rates result in lower discovery probabilities within a single listening interval. Consequently, to achieve the desired 0.9 discovery probability within the specified discovery latency, it becomes necessary to increase the total number of listening intervals (i.e., reduce epoch size and increase the number of epochs). Listening intervals generally consume more energy. As a result, the extended listening intervals lead to increased overall energy consumption of the system, resulting in a reduction in the system's lifetime.

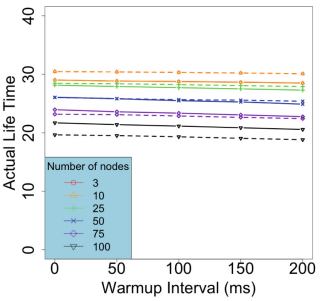


(a) Discovery Probability (b) Lifetime

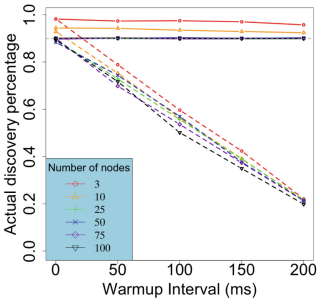
Fig. 10. Results of discovery probability and lifetime for the new model with packet loss rate. All protocols are configured to attempt to achieve 90% probability of discovery. Random packet losses range from 0% of packets to 30% of packets (x-axis).



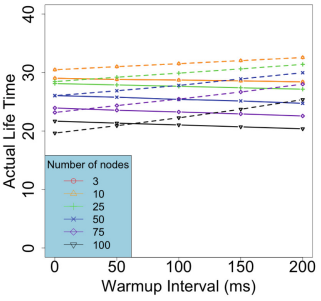
(a) Discovery probability with ψ_A



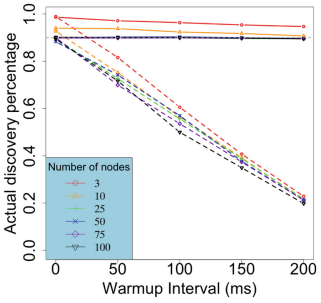
(d) Lifetime with ψ_A



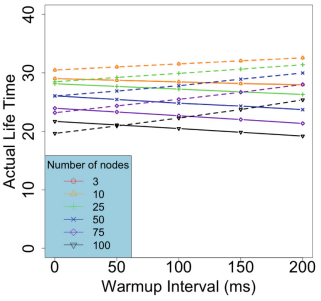
(b) Discovery probability with ψ_L



(e) Lifetime with ψ_L



(c) Discovery probability with $\psi_A + \psi_L$



(f) Lifetime with $\psi_A + \psi_L$

Fig. 11. Results of discovery probability and lifetime for **Nod** in the presence of warmup interval. All protocols are configured to attempt to achieve 90% probability of discovery. Warm-up intervals range from 0 ms to 200 ms (x-axis).

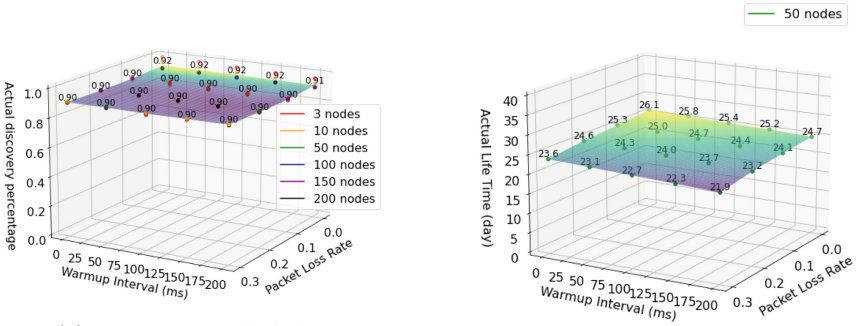
Warm-Up Interval. Figures 11(a)–(c) show how the performance of **Nod** (solid lines) compares with the performance of BLEnd (dashed lines; replicated from Figs. 3(g)–(i)). Notably, even with a large warm-up interval, **Nod** consistently achieves the target discovery probability, in contrast to BLEnd, which is dramatically impacted, especially by ψ_L.

In Figs. 11(d)–(f), we observe a slight decrease in the lifetime of nodes executing **Nod** schedules as the warm-up interval increases. This outcome can be attributed to a similar reason as discussed in Sect. 4.2. As the warm-up interval increases, the discovery probability within each epoch decreases, prompting the model to opt for a smaller E_τ. Consequently, a larger portion of the node's time is spent in listening intervals, resulting in increased overall energy consumption and reducing the system's lifetime.

Combined Packet Loss and Warm-Up Interval. Figure 12(a) presents the results of **Nod** for different packet loss rates and scanning warmup intervals. The surface plot illustrates the discovery probability, with each data point corresponding to the average discovery for different numbers of nodes. Interestingly, we observe that the discovery probability is higher when the packet loss rate is zero. This outcome can be attributed to the configurations selected by 3 nodes under those conditions, which are inherently expected to yield higher discovery probabilities. These results highlight the effective performance of **Nod**, even in scenarios involving both scanning and warm-up intervals – in all cases, the discovery rates remain consistently above the target discovery probability of 90%.

As above, the improvement in discovery probability comes at a cost. In Fig. 12(b), we present the lifetime results for 50 nodes. The impact of packet loss and warmup on the lifetime is cumulative. If either of these issues becomes severe, the lifetime further decreases, necessitating a more aggressive scheduling approach to compensate for the lower discovery probability in each epoch.

Changing Application Constraints. Different applications have unique ρ_Λ and Λ requirements. Contact tracing often prefers a 30-second discovery latency with a high 0.9 discovery probability. In contrast, opportunistic data sharing apps vary these parameters based on data urgency. Critical real-time information demands low latency and high probability, while casual sharing, like "song of the day", can accept lower probability and longer latency. To showcase our models' adaptability, we simulated various scenarios with ρ_Λ values (0.7, 0.8, 0.9) and Λ durations (10 s, 30 s) for 50 nodes. Results are in Fig. 13. As the results show, **Nod** is resilient to diverse application conditions – in all cases the protocol achieves the target discovery latency. The tradeoff is, as always, energy consumption – as examples, the expected lifetime for a node with the worst case packet loss and warmup conditions when $\rho_\Lambda = 0.7$ and $\Lambda = 30$ (Fig. 13(d)) is 31.6, while the expected lifetime for a node in the same conditions with $\rho_\Lambda = 0.9$ and $\Lambda = 10$ is 21.9.



(a) Discovery Probability

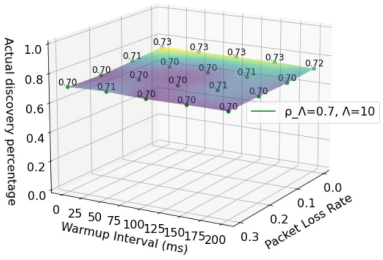
(b) Lifetime

Fig. 12. Discovery probability and lifetime results of the new model in the presence of packet loss rate and scanning warmup interval. All protocols are configured to attempt to achieve 90% probability of discovery. Random packet losses range from 0% of packets to 30% of packets. Scanning warm-up intervals range from 0 ms to 200 ms.

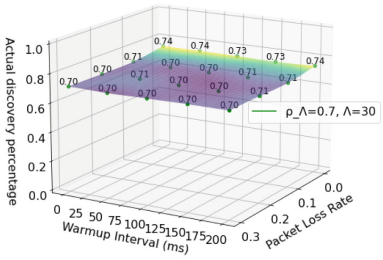
5.2 Nod in the Real World

Primarily as a proof-of-concept, we implemented **Nod** on three Android devices (also available in our repository): Google Pixel 3, Samsung S7, and Huawei N29. In our experiments, we designated the Samsung S7 device as the listener and observed its probability of detecting the other two devices. All three devices were randomly started asynchronously. We set the desired discovery probability to 0.9 and the discovery latency to 10 s, consistent with the settings used in our simulations. Based on empirical measurements with these devices (Sect. 2), we used an expected packet loss of 8%, advertising warmup as $\psi_A = 122$ and scanning warmup as $\psi_L = 147$.

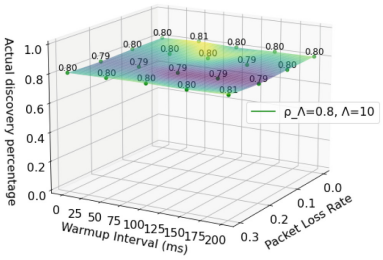
Table 1 shows varied scanning warmup interval across devices. For our tests, we used uniform parameters for three devices, considering the worst-case scenario. Alternatively, in real world deployments, **Nod** schedules can be customized to each device's scanning warmup. With inputs ($N = 3$, $A = 250$ ms, $\psi_L = 149$, $\psi_A = 135$, $\ell = 0.08$, $\rho_A = 0.9$, $\Lambda = 10000$ ms), the **Nod** model from Eq. 8 sets root epoch size E_r at 4951. The expected discovery probability based on the model is 0.903. Figure 14 shows the cumulative distribution of discovery latency. Points on the graph indicate the proportion of discoveries (y-axis) by a given time (x-axis). The pair of dashed lines indicate the **Nod** contract – the schedules are created to ensure that 90% of discoveries happen at or before 10,000 ms.



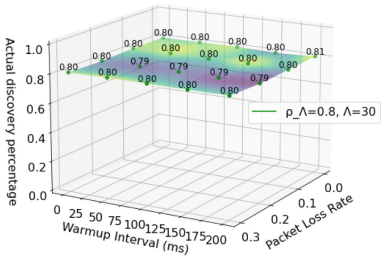
(a) $\rho_\Lambda = 0.7$ and $\Lambda = 10$



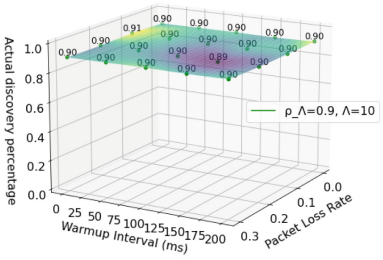
(d) $\rho_\Lambda = 0.7$ and $\Lambda = 30$



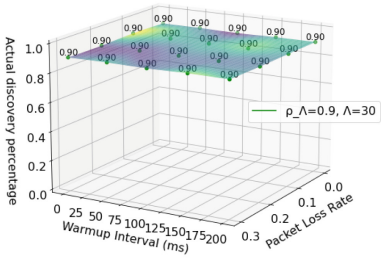
(b) $\rho_\Lambda = 0.8$ and $\Lambda = 10$



(e) $\rho_\Lambda = 0.8$ and $\Lambda = 30$



(c) $\rho_\Lambda = 0.9$ and $\Lambda = 10$



(f) $\rho_\Lambda = 0.9$ and $\Lambda = 30$

Fig. 13. Discovery probability for **Nod** in the presence of both packet loss and scanning warm-up intervals. **Nod** is configured to achieve target discovery probabilities of $\rho_\Lambda = 0.7$, 0.8, or 0.9, while Λ is set to either 10 s or 30 s. Random packet losses vary from 0% to 30%, and warm-up intervals from 0 ms to 200 ms.

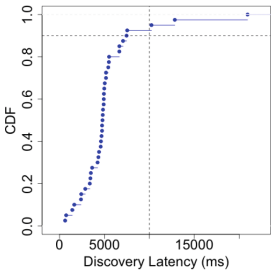


Fig. 14. Cumulative distribution of discovery times for real devices.

6 Conclusions

In this paper, we tackled two practical issues –packet loss and warmup intervals – in the context of continuous neighbor discovery protocols for real-world deployments. Our study quantified and revealed the significant impacts of these factors on the discovery performance, which are almost entirely overlooked by existing models. To overcome these challenges, we proposed a new protocol called **Nod**. This innovative protocol introduces the concept of a random extra epoch, enabling nodes initially trapped in non-functional areas to be discovered in subsequent epochs. As a result, **Nod** substantially improves the overall discovery probability. To determine optimal schedule parameters of **Nod** considering packet loss and warmup intervals, we developed analytical models that account for these practical limitations. In our evaluation, we assessed the discovery probability and energy cost associated with implementing **Nod**. While the protocol demonstrated improved resiliency, it does come at the cost of a slightly reduced node lifetime. Nonetheless, **Nod** presents a promising solution for continuous neighbor discovery, offering reliable performance on mobile devices and making it a feasible choice for real-world deployments.

References

1. Bakht, M., et al.: Searchlight: won't you be my neighbor? In: Proceedings of MobiCom, pp. 185–196 (2012)
2. Cohen, R., Kapchits, B.: Continuous neighbor discovery in asynchronous sensor networks. IEEE/ACM Trans. Netw. **19**(1), 69–79 (2010)
3. Dodangeh, M., Oude Alink, M.S., Nauta, B.: Quantifying the trade-off between latency and power consumption in bluetooth low energy and its mitigation by using a wake-up receiver. In: 2022 IEEE 8th World Forum on Internet of Things (WF-IoT), pp. 1–8 (2022). https://doi.org/10.1109/WF-IoT54382.2022.10152115
4. Dutta, P., Culler, D.: Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In: Proceedings of SenSys, pp. 71–84 (2008)
5. Girolami, M., et al.: Sensing social interactions through BLE beacons and commercial mobile devices. Pervasive Mob. Comput. **67** (2020)
6. Hatke, G.F., et al.: Using bluetooth low energy (BLE) signal strength estimation to facilitate contact tracing for COVID-19. arXiv preprint arXiv:2006.15711 (2020)

7. Herrera, J.L., Chen, H.Y., Berrocal, J., Murillo, J.M., Julien, C.: Context-aware privacy-preserving access control for mobile computing. Pervasive Mob. Comput. **87**, 101725 (2022). https://doi.org/10.1016/j.pmcj.2022.101725, https://www.sciencedirect.com/science/article/pii/S1574119222001389
8. Julien, C., Liu, C., Murphy, A.L., Picco, G.P.: Blend: practical continuous neighbor discovery for bluetooth low energy. In: Proceedings of IPSN, pp. 105–116 (2017)
9. Kindt, P.H., Chakraborty, S.: Performance limits of neighbor discovery in wireless networks. arXiv preprint arXiv:2103.05026 (2021)
10. Kindt, P.H., et al.: Griassdi: mutually assisted slotless neighbor discovery. In: Proceedings of IPSN, pp. 93–104 (2017)
11. Kindt, P.H., et al.: Optimizing BLE-like neighbor discovery. IEEE Trans. Mob. Comput. (2020)
12. Kohvakka, M., Suhonen, J., Kuorilehto, M., Kaseva, V., Hännikäinen, M., Hämäläinen, T.D.: Energy-efficient neighbor discovery protocol for mobile wireless sensor networks. Ad Hoc Netw. **7**(1), 24–41 (2009)
13. McGlynn, M.J., Borbash, S.A.: Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In: Proceedings of MobiCom, pp. 137–145 (2001)
14. Qiu, Y., Li, S., Xu, X., Li, Z.: Talk more listen less: energy-efficient neighbor discovery in wireless sensor networks. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)
15. Radhakrishnan, M., et al.: Smartphones and BLE services: empirical insights. In: Proceedings of MASS, pp. 226–234 (2015)
16. Saloni, M., Julien, C., Murphy, A.L., Picco, G.P.: LASSO: a device-to-device group monitoring service for smart cities. In: Proceedings of the International Smart Cities Conference, pp. 1–6 (2017)
17. Tseng, Y.C., et al.: Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. Comput. Netw. **43**(3), 317–337 (2003)
18. Woolley, M.: Bluetooth core specification version 5.2. Bluetooth SIG (2020)



Decentralized Collaborative Inertial Tracking

Alpha Diallo(✉) and Benoît Garbinato

University of Lausanne, Lausanne, Switzerland
{alpha.diallo,benoit.garbinato}@unil.ch

Abstract. Although people spend most of their time indoors, outdoor tracking systems, such as the Global Positioning System (GPS), are predominantly used for location-based services. These systems are accurate outdoors, easy to use, and operate autonomously on each mobile device. In contrast, Indoor Tracking Systems (ITS) lack standardization and are often difficult to operate because they require costly infrastructure. In this paper, we propose an indoor tracking algorithm that uses collected data from inertial sensors embedded in most mobile devices. In this setting, mobile devices autonomously estimate their location, hence removing the burden of deploying and maintaining complex and scattered hardware infrastructure. In addition, these devices collaborate by anonymously exchanging data with other nearby devices, using wireless communication, such as Bluetooth, to correct errors in their location estimates. Our collaborative algorithm relies on low-complexity geometry operations and can be deployed on any recent mobile device with commercial-grade sensors. We evaluate our solution on real-life data collected by different devices. Experimentation with 16 simultaneously moving and collaborating devices shows an average accuracy improvement of 44% compared to the standalone Pedestrian Dead Reckoning algorithm.

Keywords: Collaborative Indoor Tracking · Inertial Systems · Collaborative Algorithms · Indoor Localization

1 Introduction

Positioning systems are increasingly popular due to their ease of use and the growing number of digital consumer electronics such as smartphones, smartwatches, microcontrollers, etc. However, we observe that most of these systems rely on Global Navigation Satellite Systems (GNSS) such as the Global Positioning System (GPS), characterized by their availability, accuracy, and reliability [20]. Since these systems rely on satellites, they require a direct line of sight with mobile devices. Meaning that they cannot operate in indoor environments. This leaves a gap in potential use cases for location services, knowing that we spend 90% of our time indoors [11].

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 26–45, 2024.
https://doi.org/10.1007/978-3-031-63989-0_2

1.1 Indoor Tracking Systems

Many Indoor Tracking Systems (ITS) have been proposed in the literature and by the industry to overcome the limitations of outdoor positioning systems. These systems can be classified into two categories depending on whether they require fixed infrastructure. Infrastructure-based ITSs mostly rely on wireless signals, such as WiFi and Bluetooth Low Energy (BLE), to exchange data or estimate locations. Wireless signals are prone to environment perturbation and thus require some aggregation and processing steps, performed mainly by some central server. In such ITS, the central server acts as a single point of failure, a bottleneck for communications and a potential target for malicious adversaries because it stores sensitive data. On the other hand, infrastructure-less ITSs mostly rely on inertial sensors embedded in mobile devices and use their processing power to run the tracking algorithms. Inertial ITSs deployed in infrastructure-less environments suffer from accumulating errors due to environmental perturbations and noisy sensors.

In the literature, some authors combine inertial and wireless ITS to overcome the above limitations in a so-called sensor-fusion approach [8,33]. This approach offers better tracking accuracy but does not remove the need to deploy infrastructure, as the devices must calibrate their tracking algorithms using fixed access points. Another drawback of this approach is the significant computing power required to process a large amount of data several sensors collect. As a result, this approach is often evaluated in simulated environments or deployed on centralised systems. Collaborative ITS have been proposed in the literature to overcome the need for infrastructure without losing tracking accuracy.

1.2 Collaborative Indoor Tracking Systems

Collaborative Indoor Tracking Systems (CITS) tend to pursue at least one of two goals: (1) to reduce tracking errors and (2) to remove the need for a central server. Using the data collected from nearby devices, these systems effectively track devices while reducing the cost of deploying infrastructures. In the literature, CITSs can be classified into two categories: centralized and decentralized.

Centralized CITS follow the same architecture as traditional centralized ITS and aim at the first goal, i.e., reducing tracking errors. For this, each device is aware of its surroundings and collects data from nearby devices, such as Received Signal Strength Indication (RSSI). The collected data is integrated into the tracking algorithms run by the server to increase the overall accuracy of the tracking solution. This approach is popular because it gathers tracking data on a central server, allowing for better control. Yet it bears the same drawbacks as any centralized solution we already discussed.

In addition to the first goal, decentralized CITSs aim to remove the dependency on a central server by leveraging the computation capabilities of mobile devices to determine their location. In this approach, devices exchange data directly with their peers to improve their tracking estimates. Decentralized CITS fit well with inertial ITS, as they only need sensors found on most commercial handheld devices, such as smartphones or tablets.

1.3 Limitations of Existing CITS

As suggested, most decentralized CITSs in the literature focus on a tradeoff between accuracy and ease of deployment. Thus, the problems they address are centred around improving the accuracy of traditional ITS by using collaboration or reducing the hassle of deploying infrastructure and calibrating tracking algorithms. This is often achieved by relying on wireless technologies, such as BLE beacons with known locations, to track mobile devices [10]. However, this approach requires the deployment of numerous BLE beacons, impacting the deployment and maintenance cost of the solution.

Alternatively, some solutions estimate device locations by exchanging data with nearby devices, using peer-to-peer WiFi beaconing [19]. This approach offers the advantage of not relying on any infrastructure. However, it is impractical with power-constrained devices as WiFi beaconing has a significant impact on power consumption [25].

In the literature, collaboration is also used by inertial ITS to correct the accumulating errors caused by noisy sensors. Qui et al. propose a collaborative approach with a triangulation calculation model to improve the accuracy of Pedestrian Dead Reckoning (PDR) [26]. Their solution effectively reduces the average deviation from PDR to 0.5 m. However, they combine Bluetooth, WiFi and Zigbee technologies, but the latter is absent from most mobile devices. In addition, their triangulation calculation model requires collaboration between at least three devices. Furthermore, several inertial CITS still require a central server for aggregating data and running the tracking algorithms [4,15,26].

1.4 Contribution and Roadmap

In this paper, we propose a solution that relies on peer-to-peer collaboration to reduce the accumulating errors of inertial tracking. At the core of our solution lies a typical PDR algorithm, which uses data from inertial sensors embedded in each mobile device to estimate its location. This algorithm requires an initial position, heading estimates, and average step length. The initial position is typically the last known position the GPS provides before entering a building. The heading estimates are collected from the gyroscope embedded in most mobile devices today. As for the average step length, it is typically derived from the outdoor displacements of the user. The PDR algorithm is then complemented by a collaborative algorithm, which corrects tracking errors thanks to data exchanged anonymously with nearby devices.

Overall, our proposed collaborative algorithm increases the tracking accuracy of typical inertial solutions by 44%, while at the same time overcoming the limitations of centralized solutions.

The rest of the paper is structured as follows. Section 2 describes our system model and introduces more precisely the problem solved by our decentralized collaborative inertial tracking solution. Section 3 describes our solution in detail. Section 4 then presents the experimental evaluation. Section 5 highlights some observations we made after evaluating our proposed collaborative algorithm.

Section 6 discusses solutions comparable to ours. Finally, Sect. 7 summarizes our contributions and sketches possible extensions of this research.

2 System Model and Problem

We consider a system composed of an unbounded number of mobile devices, each embedding an Inertial Measurement Unit (IMU) and a wireless communication interface. The IMU consists of an accelerometer for measuring the linear acceleration of a device in three axes and a gyroscope for measuring the angular velocity. The wireless communication interface is assumed to be capable of advertising data at regular intervals, with limited impacts on the battery, and detecting nearby devices at a given distance, e.g., by measuring the received signal strength (RSS).[1] Furthermore, each mobile device is assumed to have sufficient memory and computing power to execute the algorithms required by the solution described in Sect. 3. Finally, we assume that each mobile has a means to accurately locate itself when outdoors, e.g., via GPS, but not when indoors.

2.1 Abstract Roles

In order to specify the problem addressed in this paper, we define four abstract roles that are endorsed by the physical components participating in an ITS. First, we have the ***trackees***, which are being tracked. Second, we have the ***trackers***, which are responsible for detecting the presence of the trackees. Third, we have the ***reference points***, which are responsible for providing reference coordinates within the indoor space. Finally, we have the ***aggregators***, which are responsible for computing the position of the trackees, based on information provided by the trackers and the reference points.

2.2 Problem Specification

Given the above system model and abstract roles, we can now specify the problem addressed in this paper: *we aim to build a fully decentralized ITS, where mobile devices are positioning themselves by playing all four roles and by communicating in a peer-to-peer manner to exchange and thus improve their local estimates.*

More precisely, each mobile device is a *trackee* but also the *tracker* of itself, thanks to its inertial sensors. Furthermore, each device acts as its own *reference point* when moving from outdoors to indoors and as *reference point* for other devices, when it gets close to them. Finally, each device acts as *aggregator*, since it computes its own location, using the data obtained through its *tracker* role, and the data received from other devices acting as *reference points*.

[1] This is typically the case of a BLE interface, which is what we assume for our evaluation experiments in Sect. 4.

3 A Decentralized Inertial CITS

Our solution is based on two parts. Presented in Sect. 3.1, the first part consists of a local PDR algorithm that uses the inertial sensors to estimate the position of a device. The second part, discussed in Sect. 3.2, describes the proposed collaborative algorithm that overcomes the limitations of the local PDR algorithm.

3.1 Local PDR Algorithm

The first step for tracking users in inertial-based CITS is to gather data from the devices that are being tracked. These non-processed data, defined as raw data, are extracted from the inertial sensors. They comprise a timestamp t, the acceleration on three-axis (x, y, z) obtained from the accelerometer at time t, and the angle of gyration obtained from the gyroscope at time t. To be exploitable, the raw data need to be processed. The processing step starts by computing the magnitude of acceleration using the accelerometer data to detect steps.

The magnitude of the acceleration is defined with the following equation:

$$mag = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

where x, y and z represent the data collected on the three-axis.

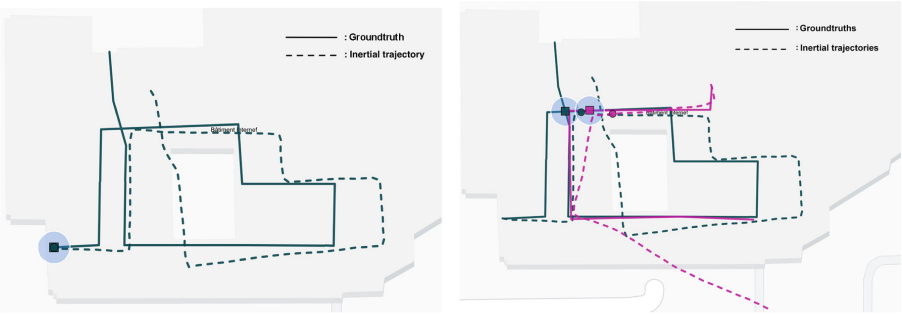
To estimate the distance travelled by the user, we use a peak detection algorithm that detects local maxima in the magnitude of acceleration. Peak detection algorithms have shown effectiveness in counting steps, thus estimating displacement. Brajdic et Harle evaluated this approach for counting steps on unconstrained smartphones and demonstrated an error rate of less than 3% [3].

After collecting and processing the inertial data, what follows is to run the PDR algorithm. The PDR algorithm is expressed with the following equation:

$$\begin{aligned} X_k &= X_{k-1} + S * \cos(\theta) \\ Y_k &= Y_{k-1} + S * \sin(\theta) \end{aligned} \tag{2}$$

In this equation, (X_k, Y_k) represents the coordinates of the new location. This new location is computed relative to the previous location (X_{k-1}, Y_{k-1}), with S representing the step length and θ being the rotation angle.

The PDR algorithm is known to generate drift because of the accumulation of errors from the inertial sensors. For example, the accelerometer can record noisy data that would be interpreted as a step, thus distorting the calculations. The same applies to the gyroscope prone to environmental perturbations. Figure 1 illustrates this phenomenon. In this figure, the solid lines represent the ground truths, and the dashed lines represent the corresponding trajectories generated with the local PDR algorithm. We can observe how the trajectories generated by PDR diverge over time from their respective groundtruths. In the following, we explain how we reduce the accumulated errors using a collaborative algorithm.



(a) Inertial and groundtruth trajectories (b) Devices within collaboration distance

Fig. 1. Drift generated by the accumulation of errors of inertial sensors.

3.2 Collaborative Part of Our Solution

The collaborative approach uses a path-loss model to estimate the distance between nearby devices. This distance is then used as a filter to consider devices only within close proximity.

A path-loss model estimates these distances using the signal strength between two devices, one acting as an emitter and the second as a receiver. Theoretically, the signal strength between an emitter and a receiver is correlated with the distance: the shorter the distance, the strongest the signal (and vice-versa). The log-distance path-loss establishes this relation with the following equation:

$$PL = P_0 + 10\lambda \log(d) + n \tag{3}$$

where P_0 is the signal strength at a reference distance, λ represents a path loss parameter related to the environment, d is the distance between the trackee and the tracker, and n is a zero-mean white Gaussian random variable.

We estimate the distance by transposing Eq. 3 such as:

$$d = 10^{\frac{P_0(d_0) - PL}{10n}} \tag{4}$$

For our configuration, we define close proximity as a distance below 4 m. This results from an observation that RSSI signals decrease substantially after a certain distance and thus become unreliable. To verify this, we ran an experiment in which we collected measurements from a BLE beacon at a given distance. Figure 2 illustrates our finding and justifies the reason for selecting an RSSI reflecting a distance below 4 m. Indeed, signals collected at a distance above 4 m fluctuate considerably and, thus, become less reliable.

In addition, as collaboration must occur seamlessly between people with no acquaintance, the distance of 4 m respects commonly applied social norms defined by the *theory of proxemics*. This theory, developed by Hall et al. in 1966, studies the human use of space within the context of culture. Hall et al. define a public space beyond which people will perceive interactions as impersonal and

relatively anonymous [9]. According to this theory, a public space around an individual ranges between 3.6 to 7.6 m. In Fig. 1, the distance of 4 m is represented as blue circles around each device. Figure 1b illustrates two devices with overlapping circles, i.e., located within collaboration distance.

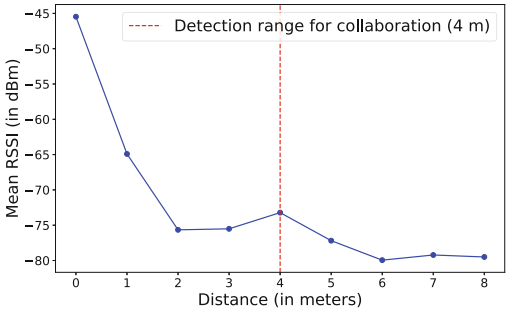


Fig. 2. Mean RSSI by distance (in meters).

As devices move indoors, they broadcast their estimated position and the accumulated errors. The estimated position is a tuple of two elements composed of their coordinates in a geographic coordinate system. The accumulated errors represent a value that grows linearly over time. To process the exchanged data, we introduce a novel collaborative algorithm that takes as input the values described hereafter.

- *Devices A and B* are the two devices whose detection ranges overlap. Each device constantly maintains its estimated location and a number representing its accumulated errors.
- The *lower-threshold* is the level of accumulated errors above which the device starts correcting its estimated location. That is, the collaborative algorithm is executed only when the device errors exceed this predefined threshold.
- The *upper-threshold* is the maximum error that is tolerated. When a device has an accumulated error beyond this threshold, it might negatively influence other devices by worsening their estimates. For this reason, its estimated location is no longer used by the devices it encounters until the error starts decreasing at the end of the walking path.

When detection ranges overlap, each of the devices run their collaborative algorithm by taking as input data provided by the other device. Thereby, the device running the algorithm is defined as device A and the other device is defined as device B. After running the collaborative algorithm, each of the devices updates its own location.

Our proposed algorithm called *Accumulation of Errors (AOE)*, uses the devices' errors accumulated over time to determine how much of the estimations need to be improved. It uses a low-complexity geographic computation technique

to draw a line between two locations estimated by overlapping devices' detection ranges. To assign new locations for both devices, the algorithm computes the ratio of accumulated errors for both devices as described in Algorithm 1. The ratio determines how far the new location would be assigned on the line. In Algorithm 1, the *intermediatePoint*() function returns a point whose distance from the starting point on a line is determined by a fraction of the line's length.

With this algorithm, we use stationary devices as reset points by decreasing their accumulated errors whenever they encounter other devices (lines 10–12). With this setting, such devices end up having a very low error over time and, thus, significantly impacting other devices.

Algorithm 1. Accumulation of errors (AOE)

1: **Input:** devices: A and B, lower-threshold l, upper-threshold u
2: **Output:** A
3: $sumErrors \leftarrow A.errors + B.errors$
4: **if** $sumErrors \neq 0$ **then**
5: $\quad ratio \leftarrow A.errors/sumErrors$
6: $\quad intermediatePoint \leftarrow intermediatePoint(A.location, B.location, ratio)$
7: $\quad$ **if** $l < A.errors$ and $B.errors < u$ **then**
8: $\quad\quad A.location \leftarrow intermediatePoint$
9: $\quad$ **end if**
10: $\quad$ **if** $A.location_{t-1} = A.location_t$ and $A.errors > 0$ **then**
11: $\quad\quad A.errors \leftarrow A.errors - 1$
12: $\quad$ **end if**
13: **end if**
14: **return** A

Data Exchange Protocol. A challenge in CITS is how to exchange data collaboratively on smartphones and tablets. A practical and viable solution consists of using BLE, as it offers a mechanism for bidirectional communications, has a low battery consumption, and is widely available. Another advantage of BLE is its low latency while allowing multiple devices to communicate simultaneously. Wu et al. experimented with using BLE for exchanging GPS data in a bidirectional manner. They demonstrate that their solution can support at least five devices to communicate simultaneously with each other. In doing so, they measure a communication latency of 56 ms [31].

In our proposed collaborative algorithm, data that is being exchanged is the estimated locations of the devices $p = (\phi, \lambda)$ and their accumulated errors e. Both ϕ and λ are defined as 64 bits floating-point numbers, while e is an integer that could be stored on 32 bits of memory. Without any encoding, we can fit the 160 bits (or 20 bytes) in the 255-byte payload allowed by the extended advertising functionality of BLE 5 [2].

4 Experiments and Evaluation

In this section, we start by describing the experiments, the deployment environment and the devices used for the evaluation. We then evaluate the proposed collaborative algorithm by comparing its results with the local PDR algorithm.

4.1 Experimental Setting and Procedure

The experiments were conducted in three phases: data collection, data aggregation, and execution replay.

Data Collection. For the first phase, five participants collected data on a single floor of a university building covering an area of up to 8400 m^2. The collection process was done with an Android app deployed on Samsung Galaxy Tabs S7, which offers an interface to draw the expected walking paths and collect inertial data. It was deployed on tablets to maximize usability, as tablets have larger screens than smartphones. The specifications of the devices selected for this experiment match the requirements listed in Sect. 2, that is, it embeds an accelerometer, a gyroscope, and a Bluetooth 5.2 chipset.

The data collection process took place according to the following protocol. Each participant starts by defining their expected walking path with the Android app. This is achieved by placing points along the expected walking trajectories that would be followed by the participant. This walking trajectory, defined as the *groundtruth*, is used later to compare how each tracking algorithm performs.

During the data collection process, each participant walks at his normal pace while holding their device. The Android app collects inertial data from the accelerometer and the gyroscope at a fixed sampling rate of 300 ms (or 3.33 Hz) when the participant walks on the predefined groundtruth.

At the end of this phase, we collected 16 trajectories associated with devices and depicted in Fig. 3. Each square represents the real location of a participant at a given time. The blue circles represent the detection range of each device, and the solid lines represent the groundtruths.

Data Aggregation. For the second phase of the experiment, we aggregated the collected trajectories to replay them on the same execution. As described in Sect. 3, we started by running the local PDR algorithm to generate the inertial trajectories. The parameters required to run the local PDR algorithm are the initial location, the step length and the orientation.

As participants start collecting data from a predefined location, we consider them aware of their initial location and capable of communicating it using the Android app. Proposing novel approaches for determining an initial location is out of scope as this issue has been studied extensively in the literature [1,6].

For the step length, each participant provides it if known. Otherwise, we estimate the step length by allowing participants to walk on a predefined line with known coordinates. Then, the line length is divided by the number of steps to obtain an average step length.

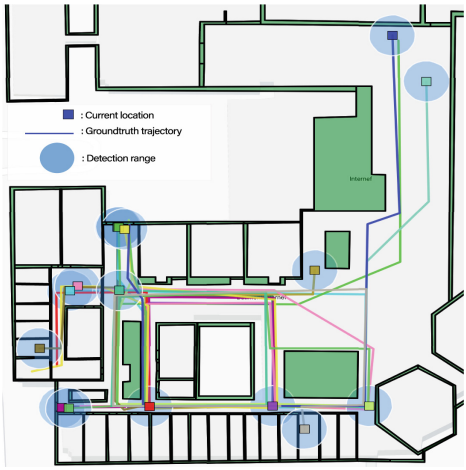


Fig. 3. Groundtruth trajectories

In our experiment, participants know their initial heading as they walk through predefined locations. For other configurations, an approach extensively discussed in the literature consists of estimating the initial heading using magnetic sensors embedded in most recent consumer electronics [14,16].

Trajectories computed using the local PDR algorithm are illustrated in Fig. 1. This figure shows how the trajectory generated by the local PDR algorithm accumulates errors over time and rapidly diverges from the groundtruth.

Execution Replay. For the third phase of the experiment, the testbed is configured to run the proposed collaborative algorithm. The experiment is done by replaying the real walking paths of each participant, allowing them to move simultaneously, i.e. each participant starts executing their walking path a few seconds after a previous path execution begins. Using the local PDR algorithm, we generate estimated trajectories, and with their detection ranges, each device held by participants detects other nearby devices along their walking path. We simulate the BLE data exchange by running the proposed collaborative algorithm when the detection ranges of two devices overlap.

The execution replay approach allows us to reproduce participants' movements with real-life data and fine-tune the collaborative algorithm parameters. The impacts of these parameters on the proposed collaborative algorithm are described hereafter.

4.2 Evaluation Metrics

The next step after running the experiments is to evaluate how the proposed collaborative algorithm compares with the local PDR algorithm. The evaluation is done with two metrics described below.

Trajectories Similarities. To evaluate how the estimated trajectories diverge from their respective groundtruths, we compute their similarities using the Discrete Fréchet Distance (DFD). It is a measure of similarity proposed by Eiter et Heikki [7] and described in Eq. 5.

$$dfd(i,j) = \begin{cases} d(P_i, Q_j) & \text{if } i = j = 1 \\ max \begin{cases} d(P_i, Q_j) \\ min \begin{cases} dfd(i-1, j) \\ dfd(i, j-1) \\ dfd(i-1, j-1) \end{cases} \end{cases} & \text{otherwise} \end{cases} \tag{5}$$

where P and Q represent trajectories such as $P = \langle p_1, ..., p_m \rangle$ and $Q = \langle q_1, ..., q_n \rangle$, with p_i and q_i representing points on each of the trajectories. Then, $d(P_i, Q_j)$ is the ground distance d between points pertaining to their respective trajectories P and Q. The ground distance between two points $p_l = (\phi_l, \lambda_l)$ and $q_k = (\phi_k, \lambda_k)$ is computed with the harvesine formula, first proposed by Jamez Andrew in 1805 [5] and defined in Eq. 6.

$$d = 2R \arcsin \sqrt{\sin^2\left(\frac{\varphi_l - \varphi_k}{2}\right) + \cos(\varphi_k)\cos(\varphi_l)\sin^2\left(\frac{\lambda_l - \lambda_k}{2}\right)} \tag{6}$$

where R is a constant representing the radius of Earth.

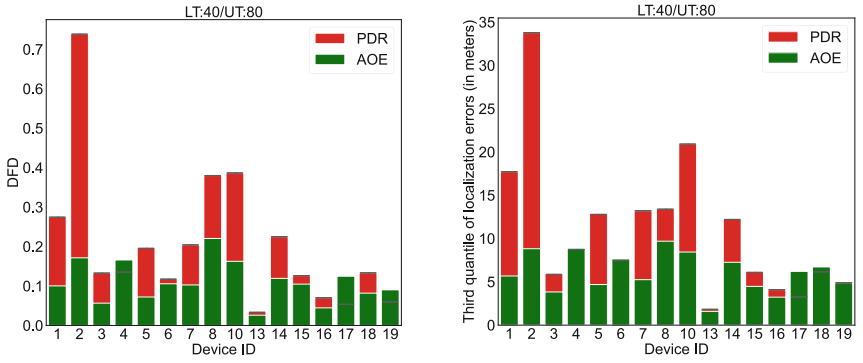
Localization Errors. This second metric represents a pairwise ground distance between individual points of two trajectories. To compute the accuracy of the local and collaborative algorithms, we use the third quantile of the localization errors. This metric is less prone to outliers than the mean localization errors, and it is now commonly used in the literature as a performance metric [23].

To evaluate a tracking algorithm, the third quantile of errors and the DFD are complementary, as the first offers the advantage of being robust to outliers and easy to interpret. Conversely, the latter is robust to temporal variability when dealing with complex data and considers the entire trajectories by capturing non-linear relationships between points of each trajectory.

4.3 Evaluation Results

The evaluation shows how our AOE algorithm performs compared to the local PDR algorithm regarding similarity and localization errors.

Trajectories Similarities. For collaborative algorithms, it is essential to highlight the similarity as some devices might negatively impact their peers, usually in the early stage of the collaboration, before progressively correcting their estimates. For our experiments, as observed in Fig. 4a, the AOE algorithm substantially increases the similarities of 13 devices out of the 16. In this plot, the x-axis represents devices identified by their unique identifier. The y-axis represents the DFD of each trajectory generated by a device. The lower the DFD, the higher



(a) DFD score (b) Third quantile of localization errors

Fig. 4. DFD and AOE Algorithms evaluation

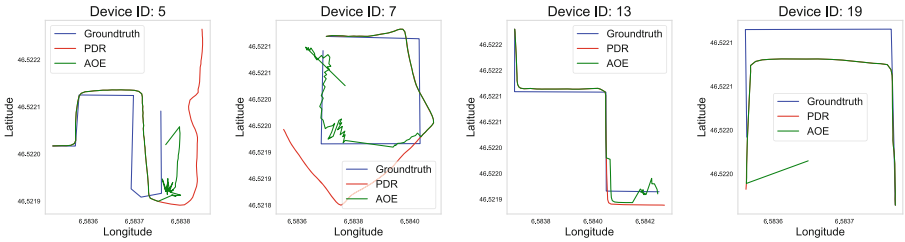


Fig. 5. Sample groundtruths and their respective trajectories generated by PDR and AOE algorithms

the similarity with the groundtruth. The red bars are the DFD of the local PDR algorithm, and the green bars are the DFD of the AOE algorithm.

Figure 5 shows a sample of trajectories generated with the local PDR algorithm (in red), the AOE algorithm (in green) and their respective groundtruths (in blue). We observe how successive collaborations bring the estimated locations closer to their groundtruths.

Localization Errors. During the collaboration, some devices tend to deviate from their actual locations after encountering other devices with wrong estimates. This behaviour tends to decrease the localization accuracy between actual and observed locations at a given time. However, as mentioned previously, the third quantile of localization errors does not consider the marginal points that deviate significantly from their actual trajectories.

Figure 4b shows the third quantile of localization errors for all the trajectories. As we observed, the AOE algorithm shows better results than the local PDR algorithm on 11 trajectories out of 16. With the remaining trajectories, there is either no improvement or a slightly negative impact of less than 3 m.

To mitigate the impact of wrong estimates on other devices during collaboration, we used a lower threshold of 40 and an upper threshold of 80. This

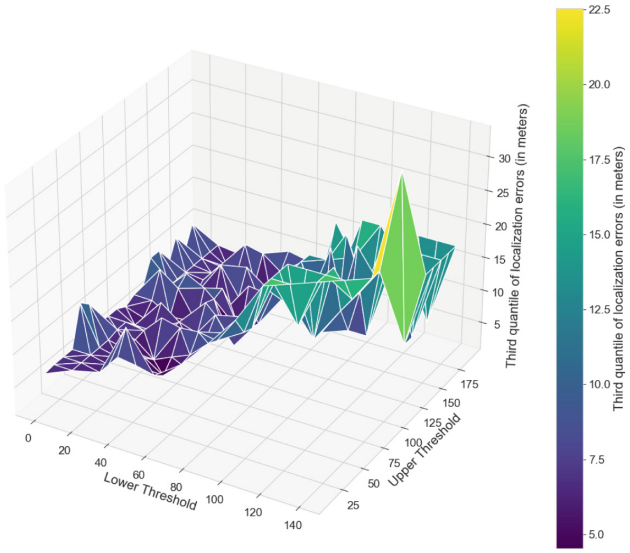


Fig. 6. Impact of thresholds on the third quantile localization errors

means that devices start collaborating only after accumulating an error of 40, and they do not consider the estimates of devices with errors above 80. In the next section, we explain how we selected these thresholds and how they impact the performance of our proposed collaborative algorithm.

5 Discussion and Future Work

In this section, we discuss the role of the thresholds, highlight some observations made after the evaluation, and conclude by showing some of the points not addressed in this paper.

5.1 Importance of Thresholds

The two thresholds introduced in Sect. 3.2 play an essential role in collaborative algorithms as they define conditions for collaboration. In inertial-based ITS, as errors accumulate over time, there is little benefit for devices to collaborate at the beginning of their walking paths. On the other hand, devices with a large amount of errors might negatively impact the estimates of other devices.

Figure 6 shows the impact of the lower and upper thresholds on the localization errors. As we observe, a high value for the lower threshold inhibits collaboration as few devices are capable of accumulating large errors. In this case, the third quantile of localisation errors of the AOE algorithm is the same as with the local PDR algorithm.

Alternatively, the AOE algorithm shows better results with a small lower threshold, below 60. This means that, for our experiment, devices correct their

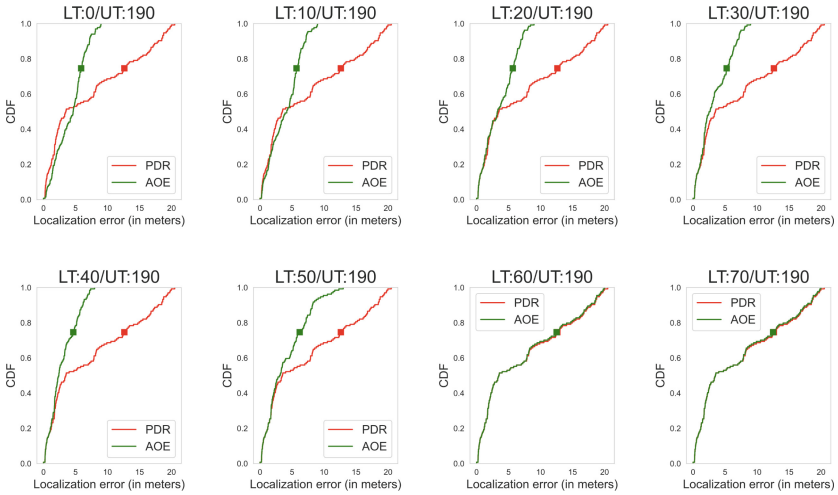


Fig. 7. Cumulative Distribution Function of the Localization Errors in meters from a single device

positions only when they accumulate a certain amount of errors. We also observe that a low margin between the lower and upper thresholds has a negative impact on the performance of the AOE algorithm. Since thresholds are preconditions for collaboration, the more restricted these conditions are, the less likely it is for devices to collaborate.

This can be observed in Fig. 7, which presents a Cumulative Distribution Function (CDF) of localization errors (in meters). This figure shows the localization errors of trajectories generated by the local PDR algorithm (in red), and those generated by the AOE algorithm (in green). The squared dots on the plots represent the third quantile of localization errors.

From these plots, we observe that the AOE algorithm significantly improves the trajectories estimated by the local PDR algorithm. This can be explained by continuous collaborations done with a small lower threshold. These plots also show that a high lower threshold reduces the chances of collaboration, thus degrading the accuracy over time. At a certain point, when there is no collaboration, the localization errors of the AOE algorithm match those of the local PDR algorithm. Therefore, the estimated trajectories follow the same pattern as the groundtruth.

5.2 Performance in Dense Environments

During our experiments, we observed that the AOE algorithm performs well in dense environments with higher chances for collaborating. Figure 5 shows four sets of trajectories obtained after running the local PDR and the AOE algorithms with a lower threshold of 40 and an upper threshold of 80. During the execution of their walking path, Device 5 made 140 collaborations with

other devices. Device 7 made 123 collaborations, whereas Device 13 made 53 collaborations, and only 29 were made by Device 19. As observed with Devices 5 and 7, the higher the number of collaborations, the higher the similarity with the groundtruth. For Device 13, most collaborations happened at the end of the execution of their walking path. By decreasing the accumulated errors after a collaboration, Device 13 acts more like a reset point, as its errors tend towards 0. In this scenario, stationary devices positively impact other devices after some collaborations.

An interesting observation comes from the estimates of Device 19, which encounters few devices and starts collaborating at the end of its walking path. These first collaborations negatively impact its estimates by a slight margin. Indeed, the estimates of Device 19 using the AOE algorithm show a DFD score of 0.09, whereas the estimates of the PDR algorithm have a DFD score of 0.05. However, this small difference is not noticeable using the third quantile of localization errors as this metric does not consider outliers. Using the third quantile of localization errors, both estimates have an accuracy of 4.88 m.

These observations lead to the conclusion that several factors must be considered before deploying a CITS. Some of these factors are the number of participants willing to collaborate and the surface area of the deployment environment.

5.3 Interest in Using a Similarity Measure

Unlike many papers presented in the literature, we chose the DFD as a metric for evaluating the performance of our proposed collaborative algorithm. Highlighting trajectory similarities should be an essential performance metric for collaborative algorithms as it shows how location estimates diverge over time. This factor is essential as one of the most common use cases for indoor tracking is navigation. For this use case, the criteria for selecting a collaborative algorithm - and its parameters - must be how the location estimates diverge from the groundtruth.

5.4 Limitations and Future Work

The collaborative algorithm proposed in this paper is well-suited for rapid deployment in large areas as it does not require any infrastructure and uses collaboration to improve the accuracy of location estimates. However, one of the limitations we do not cover in this paper is the human aspects of adopting such a solution. For instance, the reluctance of users to collaborate must be studied thoroughly. A solution to address this limitation was proposed by Pourabdollah et al. They recommend offering extra services to reward users actively participating in the collaboration process or obliging non-collaborative users to pay a subscription fee to use the service [24]. As discussed in Sect. 5.2, we consider an environment where devices often encounter other devices and collaborate. In the future, we intend to consider environments where collaboration happens less often because there are fewer devices or few of them are willing to collaborate.

5.5 Datasets

The experiments and the evaluation can be reproduced using the data available on the following repository: https://github.com/doplab/DCIT-Evaluation.

6 Related Work

This section presents the related work addressing the challenges of building an ITS for infrastructure-free environments. These solutions are either built with a decentralized architecture or use some collaboration.

Most of the solutions built for infrastructure-free environments are either motivated by afforbable-cost [21,24], ease of deployment [19] and the ability to deploy on most smartphones [10]. These three factors justify using widely available technologies such as wireless communication interfaces (BLE, WiFi, etc.) or IMU. Thus, these solutions target mobile devices with some processing power to run the tracking algorithms. To tackle the performance issues that may arise from the restricted computing capabilities of these devices, some authors also introduced the notion of collaboration in their decentralized systems.

This is the case for Ho et al., who proposed a decentralized ITS based on BLE [10]. Their solution integrates two logical entities in the tracking process: anchor and target nodes. Anchor nodes, defined in our paper as trackers, are BLE beacons that can scan and broadcast information. Target nodes, defined as trackees, can be smartphones carried by a human, a robot or an IoT sensor. Their paper presents a way of encoding information into advertising data sent by trackers at predefined locations. Their solution eliminates the use of a central server, as the coordinates of the anchors are sent directly to the trackees via Bluetooth. In addition, the trackers exchange data collaboratively with each other. After receiving signals from nearby trackers, the trackers calculate the average signal strength from other anchors. One of the limitations of their solution is the requirement of a set of beacons that cover a deployment environment. This approach is impractical in real life because BLE beacons are challenging to maintain, and using such devices significantly increases the cost of the solution. Another limitation is the requirement of BLE beacons with computing capabilities to broadcast, scan, store and analyze positioning data. The essence of BLE is power efficiency. That is the main reason why manufacturers design BLE beacons with the sole purpose of broadcasting data at predefined intervals.

In the literature, common use cases justifying the deployment of a decentralized CITS are time-critical team operations such as firefighting, urban military and search/rescue missions [17–19]. In such a constrained environment, installing any infrastructure is not feasible. That is why most decentralized CITS built for such scenarios use inertial sensors (i.e. dead reckoning) to compute the trackees' positions and exchange data in a peer-to-peer approach.

Noh et al. proposed a decentralized CITS for infrastructure-free environments [19]. Their solution works by generating an RSS map obtained with the path loss values of every coordinate on a map computed with a ray-tracing-based simulation. During the tracking process, each device obtains its position

by performing WiFi beaconing through which the RSS values of the reachable devices can be observed. These values are then matched against the RSS map to generate a list of feasible coordinates. To correct the errors in positioning, each device uses dead-reckoning with inertial sensors to eliminate all the false positives from the list of feasible coordinates. If the obtained estimates lead to a dead-end, the invalid estimates are removed from the list, and the tracking process continues until each member gets unique coordinates.

Like our proposed decentralized CITS, their solution offers the advantage of relying only on embedded sensors from the devices involved in the tracking process. However, their solution shows some limitations that need to be considered. The first limitation is that constructing an accurate RSS map with an RF ray-tracing model is challenging even in outdoor environments [29]. Another limitation is the constant communication required for running the tracking algorithms. In their paper, the devices communicate via WiFi beaconing, which is unsuitable for most tracking systems as WiFi consumes much power and might deplete the devices' battery quickly [27].

In the literature, collaboration for tracking is not only experimented on mobile devices. A growing community of researchers uses collaboration for Simultaneous Localization and Mapping (SLAM) with Autonomous Vehicles [13,22]. The literature demonstrates that collaboration among the devices decreases the communication overheads generated by centralized systems while effectively achieving positioning in challenging environments [12]. However, most of these solutions use sensors not widely available on mobile devices such as the Lidar or laser pointers [30].

Some solutions proposed in the literature are related to ours in the sense that they use collaboration to reduce the accumulated errors of PDR [15,26]. However, these solutions require a central server to operate. This is mainly because of the high computational cost of running their tracking algorithms.

Another approach presented in the literature consists of using a Kalman Filter to correct the position estimates by fusing the data from different sources affected by noise or prone to errors. This approach is mainly used for solutions using sensor fusion to correct the estimates of single devices. For instance, it is commonly used for drift removal or for filtering RSSI signals by fusing estimates from wireless and inertial-based systems [8,28,32]. In a more complex environment, this approach can suffer from high computational complexity when processing high dimensional data with frequent updates, substantially impacting real-time positioning, especially if it runs on constrained devices.

The solution presented in our paper was tested and deployed in an infrastructure-free environment. In such a constrained environment, the initial location estimates are made only using embedded inertial sensors. Collaboration using wireless signals is used occasionally to correct estimation errors. In addition, our solution can be deployed on devices with limited computing capacity and battery life, hence the interest in reconciling efficiency in estimation while using as few resources as possible.

7 Conclusion

In this paper, we proposed a novel collaborative algorithm to improve the location estimates of Pedestrian Dead Reckoning. This algorithm consists of low-complexity geometric operations that can run on mobile devices with limited battery life and constrained computing capabilities. By relying on mobile devices for running the tracking algorithms, we remove the burden of deploying costly infrastructures and the risks associated with storing sensitive information on a remote server. To demonstrate the effectiveness of our proposed collaborative algorithm, we used real-life data in the experiments. We analyzed the movement of participants and selected parameters to maximize the accuracy of the proposed algorithm. By collecting both inertial data and groundtruths, we obtained accurate measurements highlighting the advantages of collaboration in an infrastructure-free environment. In the future, it would be interesting to study how the general public might perceive such systems, especially their willingness to collaborate. We will also evaluate the privacy risks, from a user perspective, of exchanging location data with nearby devices.

References

1. Alzantot, M., Youssef, M.: CrowdInside: automatic construction of indoor floorplans. In: GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems (2012). https://doi.org/10.1145/2424321.2424335
2. Badihi, B., Sheikh, M.U., Ruttik, K., Jantti, R.: On performance evaluation of BLE 5 in indoor environment: an experimental study. In: 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5. IEEE, London (2020). https://doi.org/10.1109/PIMRC48278.2020.9217132, https://ieeexplore.ieee.org/document/9217132/
3. Brajdic, A., Harle, R.: Walk detection and step counting on unconstrained smartphones. In: Proceedings of the 2013 ACM international Joint Conference on Pervasive and Ubiquitous Computing, pp. 225–234. ACM, Zurich (2013). https://doi.org/10.1145/2493432.2493449, https://dl.acm.org/doi/10.1145/2493432.2493449
4. Constandache, I., Bao, X., Azizyan, M., Choudhury, R.R.: Did you see Bob? Human localization using mobile phones. In: Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, pp. 149–160 (2010)
5. Dauni, P., Firdaus, M.D., Asfariani, R., Saputra, M.I.N., Hidayat, A.A., Zulfikar, W.B.: Implementation of Haversine formula for school location tracking. J. Phys.: Conf. Ser. **1402**(7), 077028 (2019). https://doi.org/10.1088/1742-6596/1402/7/077028, https://iopscience.iop.org/article/10.1088/1742-6596/1402/7/077028
6. Dinh, T.M.T., Duong, N.S., Sandrasegaran, K.: Smartphone-based indoor positioning using BLE iBeacon and reliable lightweight fingerprint map. IEEE Sens. J. **20**(17), 10283–10294 (2020). https://doi.org/10.1109/JSEN.2020.2989411, https://ieeexplore.ieee.org/document/9075151/
7. Eiter, T., Mannila, H.: Computing discrete fréchet distance. Technical report, CD-TR 94/64, Information Systems Department, Technical University of Vienna (1994)

8. El-Naggar, A., Wassal, A., Sharaf, K.: Indoor positioning using WiFi RSSI trilateration and INS sensor fusion system simulation. In: Proceedings of the 2019 2nd International Conference on Sensors, Signal and Image Processing, SSIP 2019, pp. 21–26. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3365245.3365261
9. Hall, E.T., et al.: Proxemics [and comments and replies]. Curr. Anthropol. **9**(2/3), 83–108 (1968)
10. Ho, Y.H., Chan, H.C.B.: Decentralized adaptive indoor positioning protocol using Bluetooth Low Energy. Comput. Commun. **159**, 231–244 (2020). https://doi.org/10.1016/j.comcom.2020.04.041, https://www.sciencedirect.com/science/article/pii/S0140366419309831
11. IndoorAtlas: The rise of indoor positioning (2016). https://www.indooratlas.com/wp-content/uploads/2016/09/A-2016-Global-Research-Report-On-The-Indoor-Positioning-Market.pdf
12. Kachurka, V., et al.: WeCo-SLAM: wearable cooperative SLAM system for real-time indoor localization under challenging conditions. IEEE Sens. J. **22**(6), 5122–5132 (2022). https://doi.org/10.1109/JSEN.2021.3101121
13. Leung, K.Y., Barfoot, T.D., Liu, H.H.: Distributed and decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks. In: 2011 IEEE International Conference on Robotics and Automation, pp. 3841–3847. IEEE (2011)
14. Li, W., Chen, R., Yu, Y., Wu, Y., Zhou, H.: Pedestrian dead reckoning with novel heading estimation under magnetic interference and multiple smartphone postures. Measurement **182**, 109610 (2021). https://doi.org/10.1016/j.measurement.2021.109610, https://www.sciencedirect.com/science/article/pii/S0263224121005832
15. Li, Y.T., Chen, G., Sun, M.T.: An indoor collaborative pedestrian dead reckoning system. In: 2013 42nd International Conference on Parallel Processing, pp. 923–930. IEEE, Lyon (2013). https://doi.org/10.1109/ICPP.2013.110, http://ieeexplore.ieee.org/document/6687434/
16. Li, Z., Song, C., Cai, J., Hua, R., Yu, P.: An improved pedestrian navigation system using IMU and magnetometer. In: 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC), pp. 1639–1642 (2017). https://doi.org/10.1109/ICCSEC.2017.8446992
17. Liu, R., Yuen, C., Do, T.N., Zhang, M., Guan, Y.L., Tan, U.X.: Cooperative positioning for emergency responders using self IMU and peer-to-peer radios measurements. Inf. Fusion **56**, 93–102 (2020). https://doi.org/10.1016/j.inffus.2019.10.009, https://www.sciencedirect.com/science/article/pii/S1566253518306158
18. Morrison, A., et al.: Collaborative indoor navigation for emergency services personnel. In: 2017 IEEE Aerospace Conference, pp. 1–10. IEEE, Big Sky (2017). https://doi.org/10.1109/AERO.2017.7943729, http://ieeexplore.ieee.org/document/7943729/
19. Noh, Y., Yamaguchi, H., Lee, U.: Infrastructure-free collaborative indoor positioning scheme for time-critical team operations. IEEE Trans. Syst. Man Cybern.: Syst. **48**(3), 418–432 (2018). https://doi.org/10.1109/TSMC.2016.2615652
20. O'Keefe, K.: Availability and reliability advantages of GPS/Galileo integration. In: Proceedings of the 14th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS 2001), pp. 2096–2104 (2001)
21. Pascacio, P., Torres-Sospedra, J., Casteleyn, S., Lohan, E.S.: A collaborative approach using neural networks for BLE-RSS lateration-based indoor positioning. In: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 01–09 (2022). https://doi.org/10.1109/IJCNN55064.2022.9892484. iSSN 2161-4407

22. Paull, L., Huang, G., Seto, M., Leonard, J.J.: Communication-constrained multi-AUV cooperative SLAM. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 509–516. IEEE, Seattle (2015). https://doi.org/10.1109/ICRA.2015.7139227, http://ieeexplore.ieee.org/document/7139227/
23. Potortì, F., et al.: Comparing the performance of indoor localization systems through the EvAAL framework. Sensors **17**(10), 2327 (2017)
24. Pourabdollah, A., Meng, X., Jackson, M.: Towards low-cost collaborative mobile positioning. In: 2010 Ubiquitous Positioning Indoor Navigation and Location Based Service, pp. 1–5. IEEE, Kirkkonummi (2010). https://doi.org/10.1109/UPINLBS.2010.5654023
25. Putra, G.D., Pratama, A.R., Lazovik, A., Aiello, M.: Comparison of energy consumption in Wi-Fi and bluetooth communication in a smart building. In: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–6. IEEE, Las Vegas (2017). https://doi.org/10.1109/CCWC.2017.7868425, http://ieeexplore.ieee.org/document/7868425/
26. Qiu, C., Mutka, M.W.: CRISP: cooperation among smartphones to improve indoor position information. Wirel. Netw. **24**(3), 867–884 (2018). https://doi.org/10.1007/s11276-016-1373-1
27. Sadowski, S., Spachos, P.: RSSI-based indoor localization with the internet of things. IEEE Access **6**, 30149–30161 (2018). https://doi.org/10.1109/ACCESS.2018.2843325, https://ieeexplore.ieee.org/document/8371230/
28. Sheikh, H.S., Abbas, A., Abbasi, S.F., Rustam, H., Qasim, G.: Enhanced position estimation based on hybrid bluetooth and pedestrian dead reckoning using sensor fusion algorithm in indoor navigation. In: 2023 IEEE International Conference on Emerging Trends in Engineering, Sciences and Technology (ICES&T), pp. 1–5 (2023). https://doi.org/10.1109/ICEST56843.2023.10138847
29. Vitucci, E.M., et al.: Ray tracing RF field prediction: an unforgiving validation. Int. J. Antennas Propag. **2015**, e184608 (2015). https://doi.org/10.1155/2015/184608, https://www.hindawi.com/journals/ijap/2015/184608/
30. Waniek, N., Biedermann, J., Conradt, J.: Cooperative SLAM on small mobile robots. In: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1810–1815 (2015). https://doi.org/10.1109/ROBIO.2015.7419035
31. Wu, M., Ma, B., Liu, Z., Xiu, L., Zhang, L.: BLE-horn: a smartphone-based bluetooth low energy vehicle-to-pedestrian safety system. In: 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6 (2017). https://doi.org/10.1109/WCSP.2017.8171180. iSSN 2472-7628
32. You, Y., Wu, C.: Hybrid indoor positioning system for pedestrians with swinging arms based on smartphone IMU and RSSI of BLE. IEEE Trans. Instrum. Meas. **70**, 1–15 (2021). https://doi.org/10.1109/TIM.2021.3084286
33. Zou, H., Chen, Z., Jiang, H., Xie, L., Spanos, C.: Accurate indoor localization and tracking using mobile phone inertial sensors, WiFi and iBeacon. In: 2017 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), pp. 1–4 (2017). https://doi.org/10.1109/ISISS.2017.7935650



High-Performance Features in Generalizable Fingerprint-Based Indoor Positioning

Andrea Brunello[1], Angelo Montanari[1], Nicola Saccomanno[1(✉)], and Joaquín Torres-Sospedra[2]

[1] Data Science and Automatic Verification Lab, University of Udine, Udine, Italy
{andrea.brunello,angelo.montanari,nicola.saccomanno}@uniud.it
[2] Centro ALGORITMI, Universidade do Minho, Guimarães, Portugal
jtorres@algoritmi.uminho.pt

Abstract. In the context of WiFi fingerprint-based indoor localization, the present work systematically investigates the generalization capabilities, within a k-Nearest Neighbor framework, of meta-distances learned through a genetic programming approach, considering sixteen well-known and widely used benchmark datasets. Our study reveals clear variations in meta-distances performance, emphasizing that some training datasets lend themselves to superior generalization capabilities compared to others. We identify salient features, independent from the parameter k, that appear to contribute to such a superior performance: geographically large radio-map where the average number of access points detected per fingerprint is not too high. Finally, we propose a simple approach to combine the most successful meta-distances derived from our investigation. The latter leads to an overall improvement in positioning accuracy with respect to classical distances used the literature and the previously constructed individual meta-distances, confirming the soundness of the overall experimental workflow.

Keywords: Genetic programming · WiFi · Machine learning · Localization

1 Introduction

The digital age has made indoor localization a vital technology, widely used in many areas, ranging from logistics to emergency response systems [23]. Over the years, indoor localization has seen a variety of techniques being developed and explored [12,36] and, among these, WiFi fingerprint-based indoor positioning systems have gained considerable attention thanks to their ease of deployment [18]: WiFi access points (APs) are, nowadays, ubiquitous and ad-hoc equipment is not required to exploit them. In addition, signal strengths received from

Andrea Brunello and Nicola Saccomanno are co-first authors.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 46–67, 2024.
https://doi.org/10.1007/978-3-031-63989-0_3

the APs, rather than their exact location, are enough to derive position information [36].

Classical WiFi fingerprinting uses the Received Signal Strength (RSS) from one or more APs for positioning. The overall process involves two phases. In the *offline phase*, a site survey collects RSS vectors from predetermined reference points (RPs) in the indoor environment, with each vector capturing the RSSs from visible APs at the specific RP. Fingerprints are (RSS vector, position) pairs, which are stored in a database called *radio-map*. The exploitation of the positioning system by a user happens in the *online phase*, where an RSS vector sampled at an unknown location is matched against the radio-map elements to estimate the position. In this paper, we focus on deterministic fingerprinting. It often relies on the k-Nearest Neighbor (k-NN) algorithm [2], which builds its predictions based on the k most similar candidates. The reason behind such a choice is that, even in the most recent literature, k-NN still emerges as one of the most simple and yet competitive in terms of performance approaches [1,28]. Similarity can be computed using predefined and fixed formulas such as the *Euclidean*, *Cosine*, and *Matusita* distances. Alternatively, machine learned distances or meta-distances, that may better adapt to the characteristics of the considered scenarios, can be employed [6]. While a typical distance function takes the RSS values of two fingerprints as input, a meta-distance function works at a higher level of abstraction by taking as input two or more pre-calculated distance values between these fingerprints.

Despite the growing body of research in the field [26], the issue of the generalizability of machine-learned (meta) distances used for fingerprint-based indoor positioning is still a critical concern, since the ability to accurately perform on unseen data, or generalizing, is crucial for the practical implementation of indoor positioning systems. Here, we focus on this issue. To the best of our knowledge, the only related work is one of ours [6], where we evaluate the generalization capability of meta-distances trained on the UJIIndoorLoc dataset [32] using machine learning and symbolic regression via genetic programming. We also show how the meta-distance function derived from genetic programming is the best solution, among tested ones, when used in a (k-NN) positioning framework.

Starting from [6], we consider 16 well-known and widely used indoor positioning benchmark datasets, learning a different meta-distance on each of them and then analyzing their generalization capabilities across the same sources of data. Our analyses not only unveil significant variations in the performance of the meta-distances, but they also identify specific datasets that, when used for learning a meta-distance, lead to better results. Furthermore, we shed light on the dataset characteristics that underpin these superior outcomes, providing valuable insights to the research community about the collection of novel test beds, as well as to the industrial practitioners when building a radio-map. Finally, we straightforwardly combine the most successful individual meta-distances, building a model that leads to an improvement in positioning performance, paving the way for further research in this direction. Summarizing, we cover the following:

– is there any variability in the datasets, concerning the performance of the meta-distances learned over them?

– if the answer is yes, what are the most convenient datasets to be used for meta-distance learning? Can they be characterized by any features?
– is it possible to combine the best performing meta-distances to obtain an even better performance?

The paper is organized as follows. Section 2 provides background knowledge and a description of the datasets that we consider in our study. Section 3 discusses the experiments and their outcomes. Conclusions provide an overall assessment of the work done.

2 Materials and Methods

In this section, first, we present the fingerprint distance functions under consideration; they constitute the input of the learned meta-distances, as in the work of [6]. Then, an overview of the reference datasets follows. We conclude with a brief account of the fundamental concepts of symbolic regression, evolutionary computation, and genetic programming, and of their application to our case study.

2.1 Distance Functions

Literature [31,33] reports the use of various fingerprint distance functions for indoor localization. A thorough study of their ability to maintain real-world spatial relationships and to foster positioning accuracy can be found in [25]. Here, we focus on three distances that, irrespective from the positioning task, exhibited distinct behaviors, namely, *Euclidean*, *Matusita*, and *Cosine*.

Let $AP_1, \ldots, AP_n$ be the list of all the access points that can be detected in a given scenario, let $\mathbf{p}$ and $\mathbf{q}$ be two array-like fingerprints, collected in the environment, with $|\mathbf{p}| = |\mathbf{q}| = n$, and, for all $i \leq n$, let $p_i \in \mathbf{p}$ (resp., $q_i \in \mathbf{q}$) be the RSS value (which is negative) in fingerprint $\mathbf{p}$ (resp., $\mathbf{q}$) for the access point with identifier i. The first considered distance is the *Euclidean* one:

$$d_{\text{euclidean}}(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}. \tag{1}$$

Similar, except for the square root applied to the single elements, is the *Jeffries-Matusita* distance:

$$d_{\text{matusita}}(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2}. \tag{2}$$

The third distance is the *Cosine* one:

$$d_{\text{cosine}}(\mathbf{p}, \mathbf{q}) = 1 - \frac{\sum_{i=1}^{n} p_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \cdot \sqrt{\sum_{i=1}^{n} q_i^2}}. \tag{3}$$

Note that, to evaluate fingerprint distances, APs not detected by fingerprints need to be dealt with, as well as RSS negative values, which may not be supported by some distance functions, like *Matusita*. To overcome these issues, a rescaling step is applied to the fingerprint values before computing distances. Here, following the suggestion in [25], we proceed as follows. Given a fingerprint, we define function *Positive*, that subtracts from each AP the minimum recorded RSS, selected from all the observations, after subtracting 1 to it (such a value is called *min*). As a result, all fingerprint values are made positive in the interval $[1, |min| + 1]$, and the lowest admissible value 0 is used to represent those APs that have not been detected in a fingerprint. Formally, let $\mathbf{p}$ be a fingerprint and i be an AP, with RSS value p_i. Then:

$$Positive_i(\mathbf{p}) = \begin{cases} (p_i - min) & \text{if AP } i \text{ is detected,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Zero-to-One rescaling is defined as follows:

$$Zero\text{-}to\text{-}One_i(\mathbf{p}) = \frac{Positive_i(\mathbf{p})}{-min}. \tag{5}$$

Note that the function maps all RSS values in the range $[0, 1]$. We refer to the rescaled RSS values as RSSI (Received Signal Strength Indicator).

2.2 Datasets

In this work, we base our analyses on a selection of 16 datasets, well-known and widely used within the indoor positioning community. Table 1 reports their distinctive features, which are quite heterogeneous, distinguishing between the official training and test splits. As we will see in Sect. 3, apart from these features, which have been reported from previous literature, we computed an extended set of attributes to determine possible relationships linking them to their generalizability performance. In the following, we give a short account of the datasets.

DSI1 and DSI2 [20] have been collected in the same building at the University of Minho, Portugal. The first is obtained from the second by removing FPs sampled at the same RP. LIB1 and LIB2 [17] model a library at the Universitat Jaume I, Spain. They differ in the acquisition date: 2016 for the former, and 2017 for the latter. MAN1 and MAN2 have been collected on a single floor at the University of Mannheim, Germany [13]. Specifically, MAN2 is obtained from MAN1 by applying an additional post-processing step that averages FPs in 10 blocks of 10 fingerprints each. SIM001 is a synthetic dataset, generated according to a simple path loss model with additive Gaussian noise [33]. All TUT datasets have been collected at Tampere University, Finland. TUT1 [27], TUT3 [16], TUT4 [16] (which is TUT3 with swapped training and test sets), and TUT6 [15] all consider the same premises, and differ from one another in their collection strategies. TUT1 relies on cell averaging with a 1-meter grid size, TUT3 exploits crowdsourcing, and TUT6 is collected in a classical way (like, e.g., DSI1). As for

Table 1. Characteristics of the considered datasets

DB	Dim/Area	Setting	#u	#d	#b	#f	#r	FP x RP	#FP	#$\widetilde{\text{FP}}$	#AP	#$\widetilde{\text{AP}}$	#RP	FPρ	$\widetilde{\text{FP}}\rho$	Valid APs	Valid $\widetilde{\text{APs}}$
DSI1	100 m × 18 m	university floor	1	1	1	1	25	1 to 9	1369	348	152	125	230	0.73 ± 0.27	0.70 ± 0.27	24.8 ± 8.6	23.6 ± 7.8
DSI2	100 m × 18 m	university floor	1	1	1	1	25	1 to 5	576	348	152	125	230	0.31 ± 0.11	0.29 ± 0.12	24.8 ± 8.6	23.6 ± 7.8
LIB1	15 m × 10 m	two-floor library	1	1	1	2	2	12 or 36	576	3120	77	174	48	2.42 ± 0.59	2.41 ± 0.70	14.2 ± 4.6	21.0 ± 6.3
LIB2	15 m × 10 m	two-floor library	1	1	1	2	2	12 or 36	576	3120	119	165	48	2.42 ± 0.59	2.41 ± 0.70	21.6 ± 7.0	18.8 ± 5.1
MAN1	50 m × 36 m	office hallway	1	1	1	1	–	10 or 110	14300	460	27	24	130	20.88 ± 4.48	20.55 ± 5.12	10.3 ± 2.5	10.5 ± 2.5
MAN2	50 m × 36 m	office hallway	1	1	1	1	–	10	1300	460	26	25	130	1.90 ± 0.41	1.87 ± 0.47	14.1 ± 2.9	14.1 ± 2.7
SIM001	50 m × 20 m	simulated	–	–	1	1	–	1 or 10	10710	1000	8	8	1071	8.86 ± 1.80	8.86 ± 1.50	8.0 ± 0.0	8.0 ± 0.0
TUT1	124 m × 57 m	university building	1	1	1	4	<822	1 to 17	1476	490	309	183	1476	0.33 ± 0.13	0.41 ± 0.10	32.9 ± 12.1	25.0 ± 7.3
TUT2	145 m × 88 m	university building	1	1	1	3	NA	1	584	176	353	248	584	0.11 ± 0.05	0.12 ± 0.05	47.9 ± 19.0	21.9 ± 7.1
TUT3	130 m × 62 m	crowdsourced	8	21	1	5	822	1 to 14	697	3951	779	989	694	0.17 ± 0.07	0.16 ± 0.08	48.0 ± 38.4	49.7 ± 38.7
TUT4	130 m × 62 m	crowdsourced	8	21	1	5	822	1 to 14	3951	697	989	779	3843	0.96 ± 0.50	0.91 ± 0.48	49.7 ± 38.7	48.0 ± 38.4
TUT5	85 m × 145 m	university building	NA	NA	1	3	NA	1	446	982	489	296	446	0.08 ± 0.03	0.07 ± 0.02	42.5 ± 21.1	34.8 ± 13.5
TUT6	135 m × 62 m	university building	NA	NA	1	4	<822	1	3116	7269	562	566	3116	0.65 ± 0.31	0.63 ± 0.31	34.9 ± 15.9	34.7 ± 15.9
TUT7	88 m × 137 m	university building	NA	NA	1	3	NA	1	2787	6504	767	770	2787	0.48 ± 0.30	0.47 ± 0.29	27.0 ± 11.4	27.1 ± 11.1
UJI1	108703 m^2	university campus	19	25	3	4 or 5	≫933	1 to 80	19861	1111	465	367	933	2.46 ± 1.84	2.45 ± 1.62	18.1 ± 7.3	16.5 ± 6.9
UTS	44000 m^2	university building	1	1	1	16	NA	1 to 35	9108	388	557	500	1466	0.46 ± 0.29	0.51 ± 0.3	35.1 ± 13.5	42.7 ± 10.7

DB = dataset name; Dim/Area = size of the premises; Setting = collection environment; #u = number of involved users; #d = number of employed devices; #b = number of buildings; #f = number of floors; #r = number of rooms; FP x RP = fingerprints sampled per position; #FP = number of fingerprints; #AP = number of APs seen at least one time; #RP = number of distinct positions (x, y, z); FP_ρ = average FP density within a 5 m radius from each RP; Valid APs = average number of detected APs per FP; superscript ˜denotes test data, – denotes meaningless information, NA denotes not available information.

TUT2 [27], TUT5 [24], and TUT7 [15], they have been independently collected in the same university building. TUT5 follows an approach similar to TUT1, considering cell averaging with a 5-meter grid size, while the way in which TUT2 and TUT7 have been collected is the same as TUT6. UJI1 [32] has been again collected at the Universitat Jaume I. It is the largest and most complex dataset among the considered ones, as it is multi-device and the only one that considers a multi-building scenario. Finally, UTS [29] describes the FEIT Building at the Sidney University of Technology, Australia (a single building with a high number of floors).

2.3 Genetic Programming for Symbolic Regression

Drawing inspiration from [6], we rely on a symbolic regression approach, based on genetic programming, to learn a meta-distance from each of the 16 considered datasets, starting from the three distances *Euclidean*, *Cosine*, and *Matusita*. The aim is to use the meta-distance for fingerprint distance evaluation, to support k-NN based positioning.

Symbolic regression (SR) is an optimization technique that explores the space of mathematical expressions to find an optimal model, e.g., in terms of predictive capacity, for a given dataset. Unlike classical regression, which optimizes the parameters of a pre-defined model structure, SR learns both the model structure and its parameters [21]. The SR algorithm typically begins with a set of fully random expressions, formed from a combination of mathematical operators, constants, and variables, rather than a specified starting model. Each expression can be visually depicted as a tree, where leaves represent input values (here, either

constant values or the results of the three considered fingerprinting distance functions) and the root represents the output value.

We solve the aforementioned optimization problem through the *Evolutionary algorithm* outlined in [6], that implements a genetic programming task. Evolutionary algorithms (EAs) are population-based metaheuristics inspired by biological evolution and genetics, that proved themselves effective in solving combinatorial optimization problems [9]. Unlike random search, EAs leverage historical information to guide the exploration towards the most promising search space regions, by simulating natural adaptive evolution processes. In this context, the evolution process can be considered as a sequence of generations. Each generation in the EA consists of a population of candidate solutions (aka individuals) which undergo processes analogous to those in biological evolution, such as selection, crossover, and mutation. During each generation, the selection process favours individuals with higher fitness values, thereby simulating survival of the fittest. These selected individuals then proceed to crossover and mutation stages. As generations advance, the algorithm converges towards an optimal or near-optimal solution, guided by the continuous improvement of individual fitness within the population.

In the context of EA, *Genetic Programming* (GP) is a technique that evolves programs from a population of random solutions [22]. In GP, an individual is represented as a computation tree with internal nodes encoding primitives (like mathematical operators) and leaves indicating input values. The output is given by the root-encoded primitive. Due to its formulation, GP is well-suited for symbolic regression tasks.

Meta-distance Learning Task. Given that the meta-distance learning task is at the core of our work, in the following we briefly summarize the evolutionary algorithm implementation we adopted. For further details, we refer the interested reader to our previous work [6]. The proposed solution is based on the DEAP (Distributed Evolutionary Algorithms in Python) framework [10], which facilitates the quick prototyping of custom evolutionary algorithms. DEAP's core comprises two structures: the *creator* that aids in creating individuals and populations of any data structure, thus supporting all kinds of evolutionary algorithms, and the *toolbox*, a container for operators which can be used for operations like genetic mutation and individual selection.

Overall Task. The objective is to symbolically learn a distance function between two fingerprints. This function takes as input the results of the *Euclidean*, *Cosine*, and *Matusita* distance calculations made over the two fingerprints. Since the learned distance is an algebraic combination of values obtained through the traditional distances, we refer to the learned distance as a *meta-distance* (a distance over distances) in this work. For instance, Fig. 1 depicts the meta-distance learned, in [6], over the dataset UJI1, where the leaves either represent a fixed numerical coefficient or the distance value provided by the Euclidean distance (EUC), Cosine distance (COS) or Matusita distance (MAT); ST (·) nodes represent parameterized instantiations of the ST (A) subtree. Please, note that the

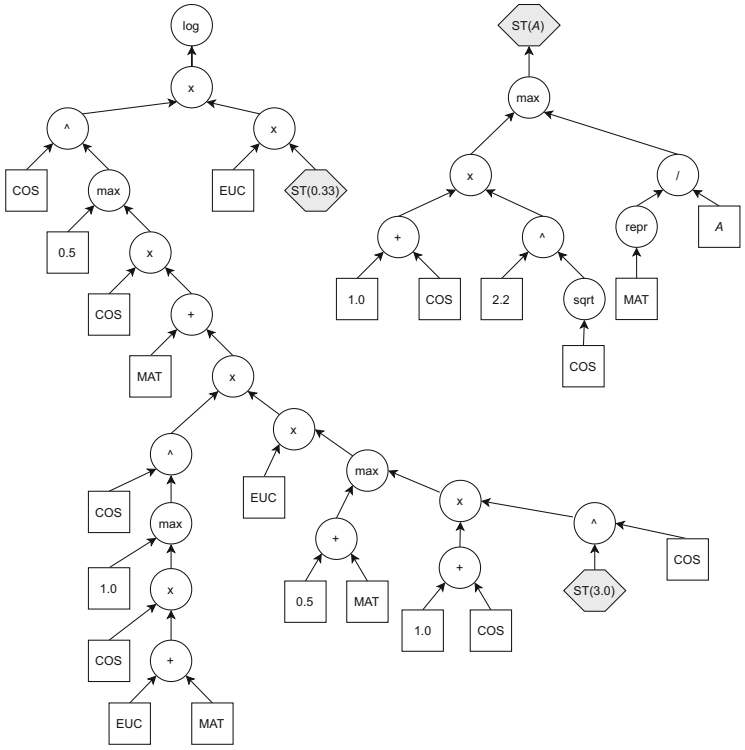


Fig. 1. The computation tree generated by the evolutionary algorithm when applied on dataset UJI1, where $ST(\cdot)$ nodes represent parameterized instantiations of the $ST(A)$ subtree. The meta-distance makes use of *Euclidean* (EUC), *Cosine* (COS), and *Matusita* (MAT) distance functions (image from [6]).

meta-distance do not use explicitly any RSSI values. Of course, such a meta-distance can then be used within any positioning framework, as it happens with classical distances. Here, as already mentioned, we will consider a k-NN approach. Figure 2 provides a schematic account of the whole genetic algorithm operation, whose single blocks are described in the following.

Population and Initialization. Each individual in the population is a computation tree that combines the values derived from *Euclidean*, *Matusita*, and *Cosine* fingerprint distances, to produce a fingerprint meta-distance. The population is initialized using DEAP's *genHalfAndHalf* method, producing random computation trees with a maximum height of 6 [14]. Half of the time, a tree with leaves at the same depth is generated; otherwise, leaves may have varying depths.

Primitives. A computation tree node can represent a single value - either one of the chosen fingerprint distance functions, a DEAP's *EphemeralConstant* (a random value from $[-10, 10]$), or a constant from $-1, 0.5, 1, 2, 3$ (square boxes

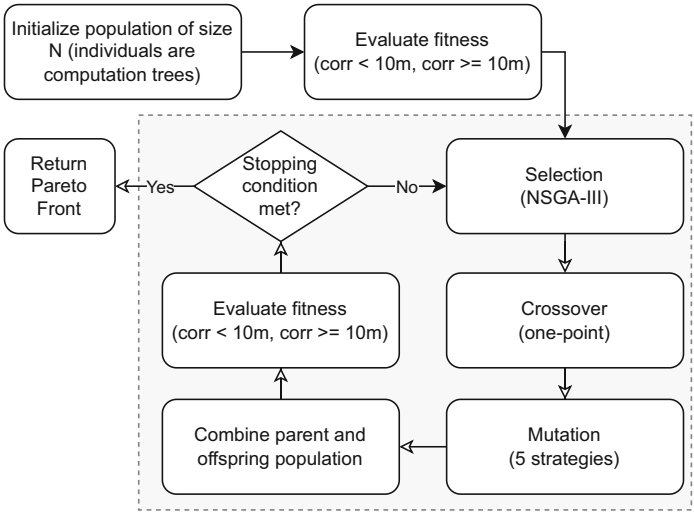


Fig. 2. Overview of the evolutionary algorithm, where the term *corr* stands for the Pearson correlation between distances in the fingerprint and real-world space. The shaded area includes the steps performed during an EA generation.

in Fig. 1). It can also denote a unary function applied to a value x, chosen from $\{log_e(x),\ \sqrt{x},\ x^2,\ |x|,\ -x,\ 1/x$ (reciprocal, repr in Fig. 1), $sin(x),\ cos(x)\}$ as well as a binary function applied to a pair of values x, y, from $x+y, x-y, x \cdot y, x/y, x^y,\ max(x,y), min(x,y)$ (circles with one or two inputs respectively in Fig. 1). To prevent invalid calculations, such as $x/0$, each function has a 'protected' implementation that substitutes NaN values with 0, positive infinite values with 1, and negative infinite values with -1.

Training Data. Fingerprint pairs are extracted from a given dataset's official training split. For each pair, *Euclidean*, *Matusita*, and *Cosine* fingerprint distance functions are calculated, along with the real-world 3D Euclidean distance between their associated locations.

Fitness Function and Best Solution Extraction. A bi-objective fitness function is used. Given an individual of the population, each objective determines the Pearson correlation between the distance values computed by such an individual over all the fingerprint pairs and the corresponding real-world Euclidean distances where such fingerprints were collected (thus, no positioning task is considered within the EA). Specifically, the first objective considers fingerprint pairs with real-world distances in the $[0, 10)$ meters range, while the second handles distances in the $[10, +\infty)$ meters range, aligning with the results presented in [25]. The two objectives are to be maximized. In computing the fitness, counter-overfitting measures are undertaken, based on the concept of training instances rotation [11]. The rationale behind this is that the fitness function should foster

the creation of individuals that encode fingerprint distances that, to the greatest extent possible, retain the real-world distances within the fingerprint space. This ensures that the relationships and structure present in the real-world are effectively mirrored within the fingerprint-based representation.

Given the two objectives, no unique best solution can be directly selected from a population. Instead, a *Pareto front* of optimal solutions is identified, including all *non-dominated* ones.[1] From the front, we select the solution with the highest hypervolume[2], among those that perform better than *Cosine* with respect to both the objectives.

Termination Criteria. They include an upper limit on the number of generations (500) and an *early stopping* strategy based on the increase of the *hypervolume* measure along the generations.

Crossover and Mutation. Given two parent solutions, two offspring are produced via one-point crossover (DEAP's *cxOnePoint*), which selects a node within each individual and exchanges the subtrees originating from that point. In addition, several mutation operators from DEAP are considered, each selected uniformly at random. These include *mutUniform*, which swaps a node in an individual with a randomly generated mathematical expression; *mutInsert*, which adds a new subtree at a random position in an individual; *mutNodeReplacement*, which replaces a randomly selected node in the individual; *mutEphemeral*, which changes the value of a single constant within an individual; and *mutShrink*, which randomly selects a subtree in an individual and replaces it with one of the subtree's arguments, also randomly chosen. Note that, in performing both crossover and mutation, anti-bloat strategies that limit the height of the generated computation trees are employed [14].

Selection. In order to foster diversity in the population, we adopt the elitist strategy implemented in NSGA-III [8], based on the concepts of *reference points* and *niche preservation*.

3 Experimental Evaluation

As already stated, we consider 16 indoor positioning datasets, each partitioned into an official training and test split. On each training set, following the symbolic regression approach described in [6], we learned a meta-distance (namely, MD_TRAINDATASETNAME) for fingerprint distance evaluation that combines the results computed by *Euclidean*, *Cosine*, and *Matusita* distances.

Then, for positioning and evaluation purposes, we rely on k-NN with $k = 3$ and the 16 above-mentioned learned meta-distances. For comparison purposes,

[1] A set $\mathcal{S}$ of solutions is said to be *non-dominated* if no solution in $\mathcal{S}$ is entirely superior to any other in terms of all objectives.

[2] The hypervolume measures the volume of the search space, bounded by a given reference point, that is weakly dominated by the points on the Pareto front [7].

the individual *Euclidean*, *Cosine* and *Matusita* distances are also taken into consideration in the evaluation. Thus, for each dataset, using the instances from the corresponding training split as the radio-map, we have run 19 (16 + 3) k-NN models. We collect the positioning performance in terms of success rate (fraction of instances for which the building and floor were correctly identified) and average positioning error in meters.

With those results at our disposal, in the subsequent sections, we will address the following research questions:

RQ1 Is there any variability in the datasets, concerning the performance of the meta-distances?
RQ2 What are the most convenient datasets to be used for meta-distance learning?
RQ3 Are there any features that characterize the datasets that perform well for meta-distance learning?
RQ4 Is it possible to combine the best performing meta-distances to obtain an even better performance?

Note that, to make our results comparable with [6], we have maintained a consistent choice of three neighbors ($k = 3$) for all results presented hereafter. While our primary objective in this research is comparative analysis rather than the pursuit of optimal positioning performance, we still recognize the potential limitation inherent in maintaining a fixed value for k. Consequently, Sect. 3.5 is devoted to comprehensively addressing this particular issue.

3.1 RQ1: Variability in the Datasets

When assessing the performance of a positioning system, a high success rate is desirable, together with a low positioning error. The two dimensions are strongly intertwined, as one cannot be evaluated aside from the other. However, a unanimous consensus about how to evaluate indoor positioning systems with a single measure is still to be reached [34]. In light of this, in analogy to [19,34], we proceed as follows. Given a positioning approach p evaluated over a dataset, we start from its average positioning error $err(p)$, calculated over all instances. Then, we consider the fraction of instances which were predicted a wrong building $w_{bd}(p)$ and a wrong floor $w_{fl}(p)$. Finally, we adjust the average positioning error:

$$err_{adj}(p) = err(p) + w_{bd}(p) \cdot dist_{p50} + w_{fl}(p) \cdot fl_{dist_{p50}},$$

where, for the given dataset, $dist_{p50}$ and $fl_{dist_{p50}}$ represent the median of the distances among all radio-map fingerprints and the median of the distances among radio-map fingerprints on a same floor of a same building, respectively. The rationale, without any other information on the topology of the premises, is that if an user were randomly positioned in any location within a floor or building, the worst-case situation would find he/she at one end and the staircase (or the correct building) at the other, necessitating a traversal of the entire floor or scenario. Conversely, the best-case situation would place the user right next to

a staircase or the correct building. Averaging these two extremities provides a median or most probable estimate, hence the half-floor or half-scenario traversal. In essence, by using this average, a balanced viewpoint that is neither overly optimistic nor pessimistic is considered.

The above described methodology allows us to easily compare approaches over the same dataset. Now, to make such a comparison fair among datasets, it should be considered that reducing the positioning error of 0.5 m may be negligible in some settings, and a considerable achievement in others. Thus, we rescale $err_{adj}(p)$ on the basis of the performance exhibited by the individual classical distances *Euclidean*, *Cosine* and *Matusita* over the same dataset, obtaining our final score:

$$score(p) = \frac{err_{adj}(c_{min}) - err_{adj}(p)}{err_{adj}(c_{max}) - err_{adj}(c_{min})},$$

where c_{min} and c_{max} are the lowest and the highest err_{adj} among the classical distances. The idea is that, in a dataset where it is difficult to reduce the positioning error, all classical distances should perform in a similar way. As for the subtraction at the numerator, it gives a positive score to an improvement over the best performing classical distance.

We are now ready to explore the results of our experiments. To begin with, Fig. 3 reports, for each dataset, the boxplot and the scores reported by the 16 meta-distances. The single scores are represented by a colored dot, being the learned meta-distance for the dataset highlighted with a cross (e.g., MD_UTS applied on UTS). Boxplots[3] as well as single data points are shown. Datasets are sorted according to the variability exhibited in the scores.

As it can be seen, especially for the datasets TUT6, TUT4, UTS, and TUT7 the choice of the meta-distance seems to play a major role when it comes to positioning performance, given the high variability exhibited by the scores. This can be easily explained for UTS, as it models a very rich scenario that, from an empirical point of view, seems more difficult to deal with than another complex dataset, UJI1.

Usually, the distance derived from a training set achieves the highest rank for its corresponding dataset (e.g., MD_UJI1 is top ranked for UJI1). Interestingly, there are some notable exceptions. For some datasets, like LIB1, the performance from its own training set (represented by a cross) is not the best. In fact, it's near the 1st quartile, indicating other training sets might offer better performance in these cases. Note also how LIB1 and LIB2 exhibit a rather different score distribution, which may be tied to peculiarities connected to their time of collection. Different behaviours are exhibited by MAN1 and MAN2 as well, probably due to the post-processing step employed in the latter. DSI1 and DSI2 are instead quite similar, and this is not surprising as the former is obtained from the latter just by removing redundant fingerprints. Finally, TUT datasets exhibit rather het-

[3] As in all following graphs, boxes extend from the 1st to the 3rd quartile values of the data, with a line at the median. Whiskers extend to the smallest and largest observations which are not outliers (assuming 1.5 times the interquartile range).

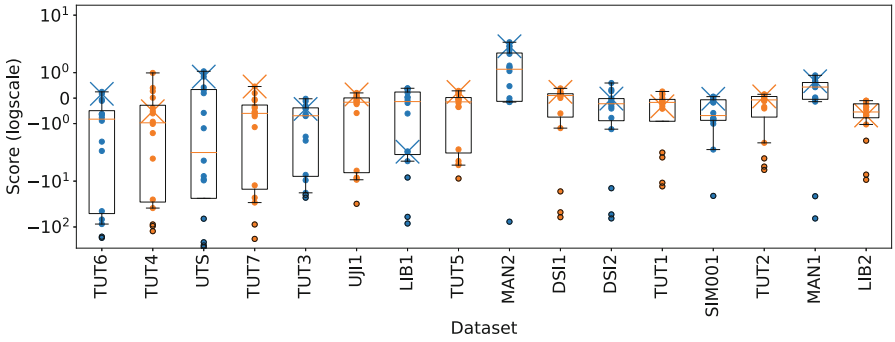


Fig. 3. Score values from the 16 learned meta-distances across each dataset, with higher scores being better. The X-axis is sorted based on score variability (higher - left, lower - right). Meta-distances derived from their respective datasets are marked with a cross.

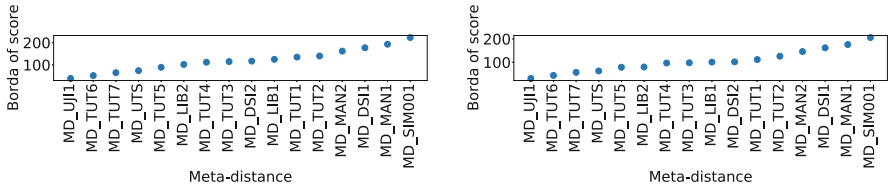
erogeneous behaviours, due to their different collection strategies and considered premises.

Answer to RQ1: there is indeed variability in the datasets concerning the meta-distances performance. For some of them, due to their peculiarities, the choice of the meta-distance severely affects the positioning outcome.

3.2 RQ2: Best Performing Datasets for Meta-distance Learning

Given the answer to *RQ1*, we now investigate which dataset(s) perform best when it comes to the meta-distance learning task. We proceed as follows. For each dataset, we sort the meta-distances according to their score, from the highest (best performance, rank position 0) to the lowest (worst performance, rank position 15) one. Then, we calculate the Borda count [4] pertaining to each meta-distance, where the voting contributions are expressed by the positions in the ranks (thus, a lower Borda corresponds to a better overall performance, and the highest/worst possible Borda is $15 \times 16 = 240$).

Figure 4 reports the results obtained for each trained meta-distance, considering all test datasets (Fig. 4a) as well as excluding the test split related to the training data of each meta-distance (e.g., MD_UJI1 is not tested on UJI1, Fig. 4b). In each graph, meta-distances are sorted from the lowest (best performing) to the highest (worst performing) Borda count. As it can be seen, MD_UJI1 performs the best, probably thanks to the complexity and richness of the premises considered in UJI1. The worst result is by far that of MD_SIM001, which raises concerns about the adequacy of the artificial generation strategy used to create the corresponding dataset. Further studies in this direction are therefore warranted, considering other synthetic datasets and approaches. Focusing on MD_DSI1 and MD_DSI2 it emerges that, at least for learning purposes, removing redundant instances brings to adverse results. Comparing MD_MAN1 and MD_MAN2, instead, the post-processing step carried out in



(a) All test datasets. (b) Without same test dataset.

Fig. 4. Borda counts for learned meta-distances. Lower is better.

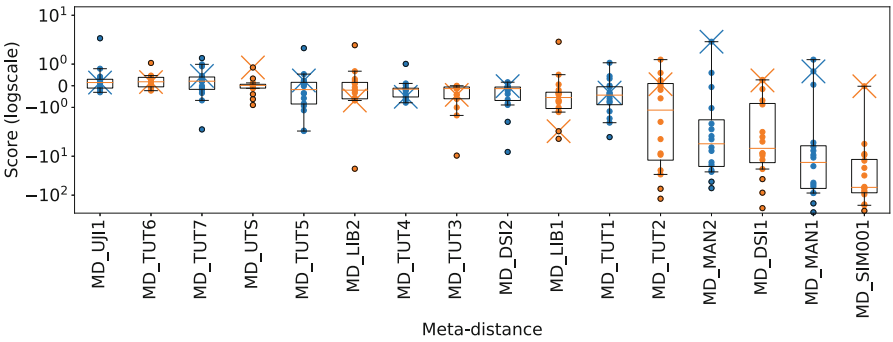


Fig. 5. Score values for each meta-distance, higher is better. Results of the meta-distances when applied on the same test set are highlighted with a cross.

MAN2 (based on multiple fingerprint average) seems to be effective. Finally, observe how the order meta-distances seems to be rather stable between the two pictures (with the exception of a close swap between MD_DSI2 and MD_LIB1), meaning that the generalization capabilities of the meta-distances are not confounded/contaminated by the presence of their corresponding test datasets. For this reason, for the remaining part of the analyses, we are going to consider the Bordas of Fig. 4a.

After the Bordas overview, we now consider some more detailed results. Figure 5 shows all scores grouped by meta-distance, the latter being sorted according to their Borda count of Fig. 4a. Results over the same dataset are highlighted with a cross. Here, the UJI1-based meta-distance emerges as the best, i.e., MD_UJI1 fits generally well in all datasets, closely followed by MD_TUT6, MD_TUT7, and MD_UTS. Meta-distances MD_TUT2, MD_MAN2, MD_DSI1, MD_MAN1, and MD_SIM001 are instead clearly under-performing. Interestingly, MD_TUT3 and MD_TUT4 behave similarly, with the former only marginally worse, despite the corresponding datasets including far less FPs (recall that the two datasets just swap their training and test splits).

To formally determine which learned meta-distance performed better than the others, we used a statistical procedure based on the pairwise evaluation of Wilcoxon signed-rank test [35], with Benjamini-Hochberg (BH) procedure to

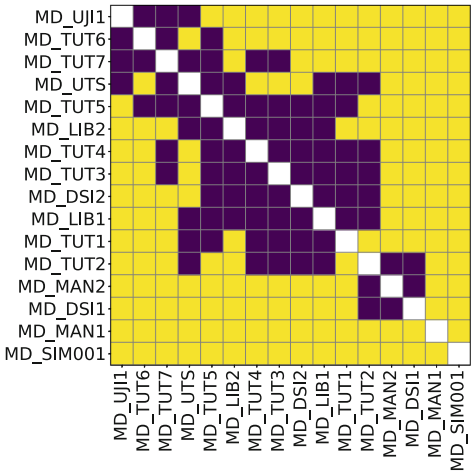


Fig. 6. Outcome of the pairwise statistical procedure based on Wilcoxon signed-rank (BH correction, α=0.05). White: same dataset; Dark: no evidence that $meta\text{-}distance_{row} \neq meta\text{-}distance_{column}$; Yellow: evidence that $meta\text{-}distance_{row}$ has a different behaviour than $meta\text{-}distance_{column}$.

account for multiple tests [3], considering a false discovery control level $\alpha = 0.05$. Figure 6 reports the overall results. Notice how the matrix confirms our previous intuitions: MD_UJI1's performance cannot be considered to be statistically different from that of MD_TUT6, MD_TUT7, and MD_UTS, and is instead superior with respect to the remaining meta-distances (considering the order provided by Fig. 4a). MD_SIM001 is statistically different (and inferior, again considering the order of Fig. 4a) with respect to all the other meta-distances. Finally, the difference between MD_TUT3 and MD_TUT4 is not statistically significant; this also holds between MD_TUT1 and MD_TUT5, which are indeed very similar considering the collection strategies and modeled premises of the corresponding datasets.

Answer to RQ2: there is a clear difference between the training sets when it comes to the performance of the meta-distances learned over them. Datasets UJI1, TUT6, TUT7, and UTS are better to generate generalizable meta-distances than the others; SIM001 is the worst one.

3.3 RQ3: Characterization of Well Performing Training Sets

Since the dataset used for meta-distance learning matters, what are the key features characterizing the well performing ones? To answer that, we define a set of attributes over the training splits of the 16 datasets. We start from the features of Table 1, disregarding those with a too-low variance, such as *#b,* or those with unknown values, like *#r*. In addition, to better describe some of the attributes, we calculate statistics over them, i.e., percentiles (perc), mean, and standard

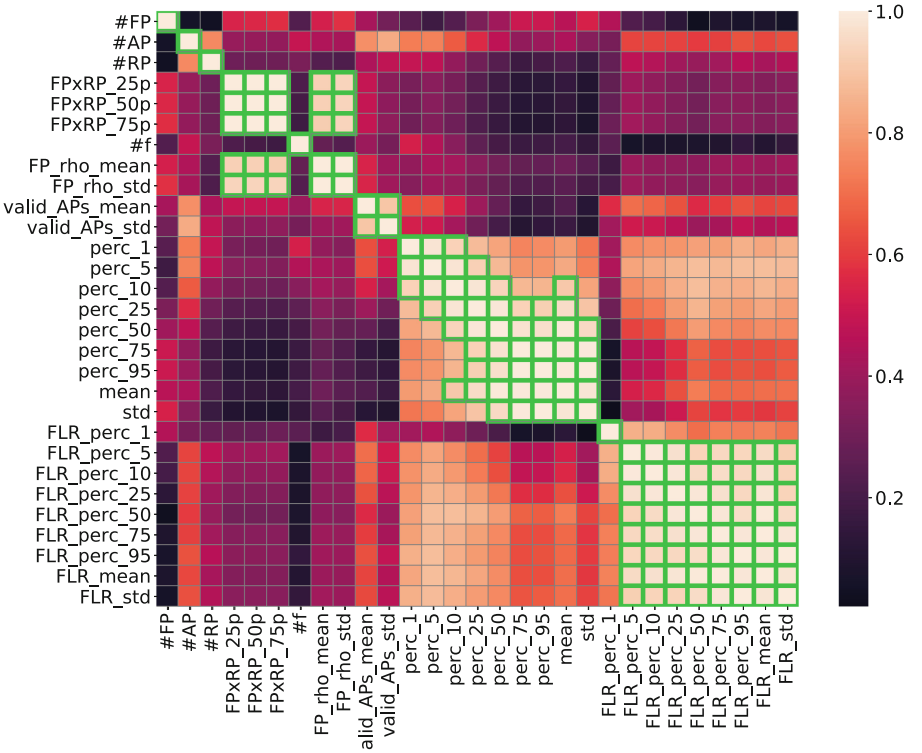


Fig. 7. Attributes and their Pearson correlations (color: darker=lower correlation; lighter=higher correlation). Correlations ≥ 0.9 are highlighted in green. (Color figure online)

deviation (std). Finally, we add features pertaining to the spatial characteristics of the radio-maps: 1st, 5th, 10th, 25th, 50th, 75th, 95th percentiles, mean, and standard deviation of the distances between the collected fingerprints, both overall and considering those on a same floor.

Figure 7 provides an overview of the considered attributes, with their pairwise Pearson correlations (colors). Correlations ≥ 0.9 are highlighted in green. The matrix is rather self-explanatory. As an example, it shows that the attributes related to the radio-map fingerprint distance values over a same floor are highly correlated, with the notable exception of the 1st percentile (robust approximation for the *minimum* value).

Before proceeding with our analyses, given the evident redundancies, we perform a feature selection phase consisting of two steps. First, we iteratively remove attributes so to break all correlations ≥ 0.9, i.e., near collinearities. Then, to solve possible multi-collinearity issues, we rely on the Variance Inflation Factor (VIF) to filter the remaining features, considering, as it is typically done [5], a threshold of 10. We obtain the following subset of attributes: *#AP*, *#RP*, *#f*, *FP_rho_mean*, *valid_APs_mean*, *perc_50*, and *FLR_perc_1*.

Analyzing how, for each dataset, the value of the selected attributes relates to the Borda counts of the corresponding meta-distances, some relationships emerge (see Fig. 8). Attributes *FP_rho_mean* and *FLR_perc_1* exhibit a weak correlation. As for the remaining ones, their increase seems to be connected with a decrease in the Borda scores and, thus, better learned meta-distance performance.

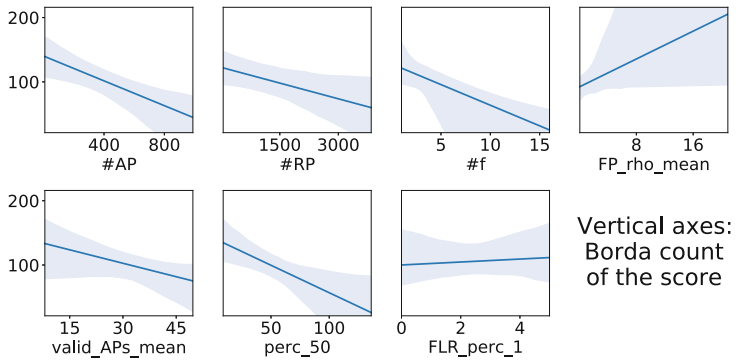


Fig. 8. Different attributes vs. meta-distances Borda count. Lines have been obtained through linear interpolation of the 16 points pertaining to the different training datasets. The 95% confidence interval is also depicted.

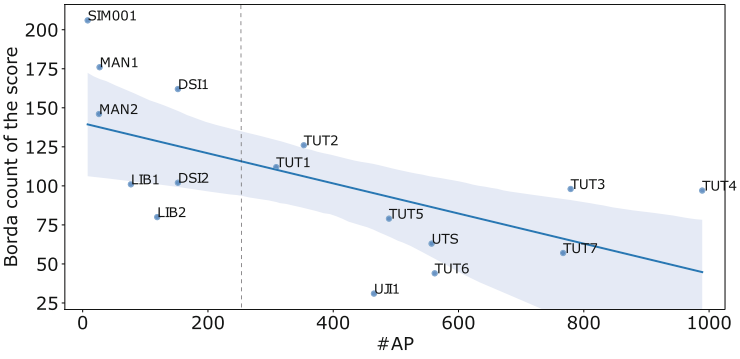


Fig. 9. Number of APs seen at least one time vs. meta-distances Borda count. The solid line has been obtained through linear interpolation. The 95% confidence interval is also depicted. The dashed line separates two groups of datasets according to their number of APs.

Let us now focus on *#AP*. Figure 9 shows that, besides the previously-described general trend, some datasets have a divergent behaviour, as in the case of DSI1 and LIB2. The two have a very similar number of APs, but the corresponding meta-distances are characterized by rather different Bordas. It might be that such discrepancies are explained by taking into account more than one attribute. To verify that, we build a Sklearn's *DecisionTreeClassifier* which,

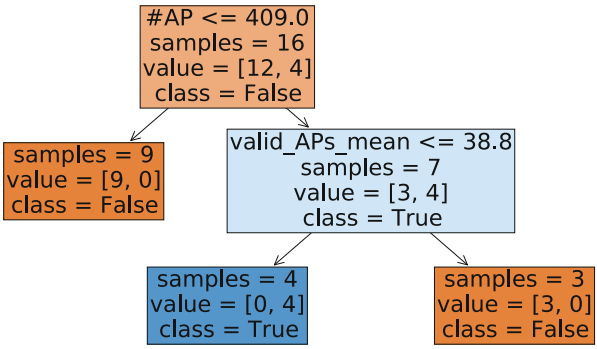


Fig. 10. Decision tree that predicts good vs bad datasets.

based on the selected features, tries to discriminate the datasets into two classes: those that in Fig. 5 led to a learned meta-distance with a median score greater than zero (class = True) and the others (class = False). Intuitively, the former represents good datasets, that are expected to produce meta-distances that perform better, on average, than the best classical distances, namely, UJI1, TUT6, TUT7, and UTS. Figure 10 shows the generated decision tree, which achieves an accuracy of 100% and, thus, perfectly explains the instances.[4] Interestingly, good datasets are characterized by a rule of the form *"in their radio-map, the number of APs seen at least one time is large enough, and the average number of APs detected per fingerprint is not too high"*. The first condition is strongly linked to the size of the geographical area covered by the radio-map: indeed, ordering the datasets by $\#AP$ (i.e., scanning Fig. 9 left to right), it can be noticed that UJI1, UTS, and all TUT datasets are clearly distinguished with respect to DSI, LIB, MAN, and SIM001 datasets (i.e., first considerable gap according to x-axis in Fig. 9, highlighted by a dashed vertical line). This is a sensible result, as larger radio-maps are expected to represent more heterogeneous scenarios, which can in turn support the meta-distance learning tasks. As for the other condition, it seems, at first glance, to be linked with the *curse of dimensionality*: distances may be less effective when working with vectors composed of several non-zero values, producing less meaningful distance evaluations that hinder the meta-distance learning phase. From a broader perspective, the learning degradation is supposed to happen also when a large number of common APs is witnessed in several, spatially dispersed fingerprints. This point, pertaining to the APs distribution among fingerprints and spatial locations, seems to be linked to the concept of information entropy, and it surely deserves attention in future work.

Answer to RQ3: the well performing training sets are characterized by a geographically large radio-map where the average number of APs detected per fin-

[4] Note that here we use the decision tree model to perform *explanation*, not *predict* over new instances: thus, we are not interested in its generalization capability.

gerprint is not too high. Notably, the sheer number of fingerprints is not enough to characterize a high-quality dataset.

3.4 RQ4: Meta-distances Combination

In this section, we build a straightforward combination of the meta-distances based on UJI1, TUT6, TUT7, and UTS datasets (i.e., MD_UJI1, MD_TUT6, MD_TUT7 and MD_UTS), which we determined to be the best-performing ones. We proceed as follows. Let us recall that meta-distances are used within a 3-NN algorithm which, given a test set fingerprint, determines the 3 closest fingerprints in the corresponding training set radio-map according to a given distance function. Thus, considering four meta-distances, each test set instance has four distances with respect to every radio-map fingerprint. To derive a single value, we simply calculate the l_2-norm of the vector composed of the four distances. The underlying reason is that the different meta-distances, being trained independently on different datasets, produce distance values with differing ranges. Utilizing the l_2-norm allows us to seamlessly combine them without being affected by such range discrepancies.

To assess the achieved localization performance, results in terms of success rate and positioning error are reported in Table 2. Here, we consider also distance *Cosine*, since it exhibited the best positioning behavior in [25], and the meta-distances based on UJI1, i.e., the winner of our previous Borda score experiments, TUT6, TUT7, and UTS, i.e., the others involved in the combination process. In the table, the number of times each method scored as the best one (i.e., the row denoted with "# 1st") is determined separately for the two evaluation parameters. As it can be noticed, the simple combination already leads to an overall better positioning performance, given that the combined metric ("Comb."in Table 2) ranks first with the highest frequency. This paves the way for further research to enhance the performance of localization systems, possibly by means of more sophisticated ensemble techniques.

Answer to RQ4: Despite the very simple combination strategy, a gain in performance is obtained with respect to the *Cosine* distance and all the single meta-distances involved in the combination.

3.5 On the Number of k Nearest Neighbors

As previously mentioned, all the results obtained so far in this study were based on a fixed number of $k = 3$ neighbors within a k-NN framework. It is generally understood that different datasets may require a specific number of neighbors k for optimal positioning performance (see, for instance, [30]). However, our primary objective here is to examine the relative performance of the meta-distances across various datasets, rather than to achieve the absolute best in localization accuracy. Therefore, for our research objectives, it is sufficient to demonstrate that, even with a different k value, the observed similarities and differences among the meta-distances persist.

Table 2. Positioning performance of meta-distances combination

Dataset	Success rate						Avg Positioning Error [m]					
	Cosine	MD_TUT6	MD_TUT7	MD_UJI1	MD_UTS	Comb	Cosine	MD_TUT6	MD_TUT7	MD_UJI1	MD_UTS	Comb
DSI1	–	–	–	–	–	–	5.4187	5.1501	5.2443	5.1711	5.3266	**4.9248**
DSI2	–	–	–	–	–	–	4.7864	4.9256	4.9169	4.7817	**4.5835**	4.6458
LIB1	**0.9990**	0.9987	**0.9990**	**0.9990**	0.9984	0.9984	2.7940	2.7214	2.7241	2.7330	2.7759	**2.6910**
LIB2	0.9997	**1.0000**	0.9968	0.9997	0.9984	**1.0000**	2.8403	3.0970	3.2323	**2.8286**	2.9551	3.0229
MAN1	–	–	–	–	–	–	2.7242	2.5560	**2.3733**	2.5517	2.7136	2.4565
MAN2	–	–	–	–	–	–	2.6138	2.4425	2.2171	2.0456	2.5752	**1.9692**
SIM001	–	–	–	–	–	–	2.9739	2.7946	3.2019	2.9579	2.9503	**2.7907**
TUT1	0.9061	0.9061	0.9061	**0.9102**	0.9041	0.9082	6.4019	6.8945	7.0979	6.2686	**6.1739**	6.3588
TUT2	0.8864	0.8807	0.8466	**0.9318**	0.8977	0.8977	11.1418	9.4256	9.5493	**9.0762**	10.7162	10.1477
TUT3	0.8932	0.9015	0.8884	0.8881	0.8927	**0.9036**	**7.8789**	7.9837	8.2745	8.3370	8.2318	8.1098
TUT4	0.9498	0.9584	**0.9656**	0.9555	0.9527	0.9584	5.5748	5.6628	5.5302	5.6289	5.6651	**5.4385**
TUT5	0.8921	0.9318	0.9226	**0.9593**	0.8870	0.9155	6.3685	**5.8878**	5.9721	6.4486	6.3387	5.9040
TUT6	0.9986	0.9996	0.9997	0.9996	0.9975	**1.0000**	2.4867	2.5008	**2.4814**	2.7035	2.6469	2.5004
TUT7	0.9906	**0.9926**	0.9924	0.9903	0.9889	0.9917	2.5727	2.5904	**2.3249**	2.6483	2.7153	2.4988
UJI1	0.9199	0.9199	0.9145	0.9226	**0.9298**	0.9217	7.5601	**6.7333**	7.1024	6.8471	7.4413	6.8007
UTS	0.9235	0.9367	0.9367	0.9261	0.9340	**0.9446**	7.4480	7.2641	7.6870	7.8682	**7.1957**	7.4349
# 1st	1	2	2	**4**	1	**4**	1	2	3	2	3	**5**

Table 3. Pearson correlations between the scores, varying k

k	1	3*	5	7	9	11
Correlation	0.9938	1.0	0.9994	0.9985	0.9977	0.9975
P-value	1.38e−244	0	0	0	1.27e−298	2.23e−293

*: For validation purposes, we also report the correlation between the 3-NN scores and themselves.

To do that, we proceed as follows. We rerun the experimental procedure outlined in Sect. 3, applying the same set $k = \{1, 3, 5, 7, 9, 11\}$ employed in [30]. For each k, we collect the *scores* of each meta-distance with respect to every test set. Then, for each k, we sort the scores according to the meta-distance and test set, obtaining k lists ordered in the same fashion. As the final step, we calculate the Pearson correlation between the list corresponding to $k = 3$ and every other list. Results are reported in Table 3. As it can be seen, all correlation values are statistically significant and exceed 0.99, meaning that the relationships between the scores are linearly preserved when changing the value of k.

This should not come as a surprise, as the meta-distances were trained by means of the evolutionary algorithm described in Sect. 2.3, optimizing the correlation between the fingerprint distance values and those of the corresponding real-world locations, irrespective of any specific positioning task. This confirms the intuition reported in [6]: optimizing the correlation, or in other words, learning to retain real-world distances within the fingerprint space, proves to be effective, regardless of the specific downstream tasks (such as positioning) to be subsequently undertaken.

4 Conclusions

In this work, we systematically investigated, based on 16 well-known datasets, the generalization capabilities of meta-distances learned through a genetic programming approach in the context of WiFi fingerprint-based indoor positioning. We found marked variations: some datasets lend themselves to superior generalization capabilities compared to others when used for meta-distance training. Moreover, we identified salient features linked to such superior results: datasets characterized by a geographically large radio-map and a moderate average number of APs detected per fingerprint tended to perform well. Finally, we demonstrated the potential of combining the best performing meta-distances through a simple approach, which outperformed all distance and meta-distance contenders. Our research highlights the role of training set selection in meta-distance learning for WiFi fingerprint-based indoor localization. Moreover, the identified features can inform dataset selection and creation phases for indoor positioning.

As for future work, besides those already mentioned in the paper, we will investigate the effectiveness of genetic programming in learning distances directly from RSS values. Additionally, we will explore the potential of other symbolic regression approaches, such as those based exclusively on or in conjunction with deep learning techniques. Finally, note that, despite its simplicity and intuitiveness, the method we considered here to build the ranks, i.e., Borda count, does not take into account the magnitude of the differences between the elements. Further studies will be conducted considering approaches capable of dealing with continuous data.

Acknowledgements. Angelo Montanari and Nicola Saccomanno acknowledge funding from *ublox AG*, project *Towards a uniform modeling framework for fingerprint-based positioning*; Andrea Brunello, Angelo Montanari, and Nicola Saccomanno also acknowledge funding from the Italian INdAM-GNCS, 2023 project *Analisi simbolica e numerica di sistemi ciberfisici*, CUP_E53C22001930001.

References

1. Alitaleshi, A., Jazayeriy, H., Kazemitabar, J.: EA-CNN: a smart indoor 3D positioning scheme based on Wi-Fi fingerprinting and deep learning. Eng. Appl. Artif. Intell. **117**, 105509 (2023)
2. Bahl, P., Padmanabhan, V.N.: RADAR: an in-building RF-based user location and tracking system. In: Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM), pp. 775–784. IEEE Computer Society (2000)
3. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Stat. Soc. Ser. Stat. Methodol. **57**(1), 289–300 (1995)
4. Borda, J.D.: Mémoire sur les élections au scrutin. Histoire de l'Academie Royale des Sciences pour 1781 (1784)
5. Brunello, A., Civilini, M., De Martin, S., et al.: Machine learning-assisted environmental surveillance of Legionella: a retrospective observational study in Friuli

Venezia Giulia region of Italy in the period 2002–2019. Inform. Med. Unlock. **28**, 100803 (2022)
6. Brunello, A., Montanari, A., Saccomanno, N.: A genetic programming approach to WiFi fingerprint meta-distance learning. Pervasive Mob. Comput. **85**, 101681 (2022)
7. Cao, Y., Smucker, B.J., Robinson, T.J.: On using the hypervolume indicator to compare Pareto fronts: applications to multi-criteria optimal experimental design. J. Stat. Plan. Inference **160**, 60–74 (2015). https://doi.org/10.1016/j.jspi.2014.12.004
8. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2013). https://doi.org/10.1109/TEVC.2013.2281535
9. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
10. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)
11. Gonçalves, I., Silva, S.: Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş, Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 73–84. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37207-0_7
12. Khalajmehrabadi, A., Gatsis, N., Akopian, D.: Modern WLAN fingerprinting indoor positioning methods and deployment challenges. IEEE Commun. Surv. Tutor. **19**(3), 1974–2002 (2017)
13. King, T., Kopf, S., Haenselmann, T., et al.: CRAWDAD dataset mannheim/compass (v. 2008-04-11) (2008). https://doi.org/10.15783/C7JS3P
14. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Stat. Comput. **4**(2), 87–112 (1994)
15. Lohan, E.S.: Additional TAU datasets for Wi-Fi fingerprinting- based positioning (2020)
16. Lohan, E.S., Torres-Sospedra, J., Leppäkoski, H., et al.: Wi-Fi crowdsourced fingerprinting dataset for indoor positioning. Data **2**(4), 32 (2017)
17. Mendoza-Silva, G.M., Richter, P., Torres-Sospedra, J., et al.: Long-term WiFi fingerprinting dataset for research on robust indoor positioning. Data **3**(1) (2018)
18. Mendoza-Silva, G.M., Torres-Sospedra, J., Huerta, J.: A meta-review of indoor positioning systems. Sensors **19**(20), 4507 (2019)
19. Moreira, A., Nicolau, M.J., Meneses, F., Costa, A.: Wi-Fi fingerprinting in the real world-RTLS@ UM at the EvAAL competition. In: Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–10. IEEE (2015)
20. Moreira, A., Silva, I., Torres-Sospedra, J.: The DSI dataset for Wi-Fi fingerprinting using mobile devices (2020)
21. Orzechowski, P., La Cava, W., Moore, J.H.: Where are we now? A large benchmark study of recent symbolic regression methods. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 1183–1190 (2018)
22. Poli, R., Langdon, W., Mcphee, N.: A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd. (2008)
23. Potortì, F., Crivello, A., Palumbo, F., Girolami, M., Barsocchi, P.: Trends in smartphone-based indoor localisation. In: Eleventh International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–7. IEEE (2021)
24. Richter, P., Lohan, E.S., Talvitie, J.: WLAN (WiFi) RSS database for fingerprinting positioning (2018)

25. Saccomanno, N., Brunello, A., Montanari, A.: What you sense is not where you are: on the relationships between fingerprints and spatial knowledge in indoor positioning. IEEE Sens. J. **22**, 4951–4961 (2021)
26. Shang, S., Wang, L.: Overview of WiFi fingerprinting-based indoor positioning. IET Commun. **16**(7), 725–733 (2022)
27. Shrestha, S., Talvitie, J., Lohan, E.S.: Deconvolution-based indoor localization with WLAN signals and unknown access point locations. In: International Conference on Localization and GNSS (ICL-GNSS), pp. 1–6. IEEE (2013)
28. Song, X., et al.: CNNloc: deep-learning based indoor localization with WiFi fingerprinting. In: IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation Conference (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 589–595. IEEE (2019)
29. Song, X., Fan, X., Xiang, C., et al.: A novel convolutional neural network based indoor localization framework with WiFi fingerprinting. IEEE Access **7**, 110698–110709 (2019)
30. Torres-Sospedra, J., Gaibor, D.P.Q., Nurmi, J., Koucheryavy, Y., Lohan, E.S., Huerta, J.: Scalable and efficient clustering for fingerprint-based positioning. IEEE Internet Things J. **10**(4), 3484–3499 (2022)
31. Torres-Sospedra, J., Montoliu, R., Trilles, S., et al.: Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems. Expert Syst. Appl. **42**(23), 9263–9278 (2015)
32. Torres-Sospedra, J., Montoliu, R., Usó, A.M., et al.: UJIIndoorLoc: a new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems. In: Fifth International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 261–270. IEEE (2014)
33. Torres-Sospedra, J., Moreira, A.J.C.: Analysis of sources of large positioning errors in deterministic fingerprinting. Sensors **17**(12), 2736 (2017)
34. Torres-Sospedra, J., Silva, I., Klus, L., et al.: Towards ubiquitous indoor positioning: comparing systems across heterogeneous datasets. In: Eleventh International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–8. IEEE (2021)
35. Woolson, R.F.: Wilcoxon signed-rank test. Wiley Encycl. Clin. Trials 1–3 (2007)
36. Xia, S., Liu, Y., Yuan, G., et al.: Indoor fingerprint positioning based on Wi-Fi: an overview. ISPRS Int. J. Geo-Inf. **6**(5), 135 (2017)



SCORE: Scalable Contact Tracing over Uncertain Trajectories

Avinaba Mistry(✉), Xichen Zhang, Suprio Ray, and Sanjeev Seahra

University of New Brunswick, Fredericton, Canada
{avinaba.mistry,xichen.zhang,sray,sseahra}@unb.ca

Abstract. In the context of a global pandemic, mitigating contagion risk requires an integrated analysis of global positioning data from location-based services and complex disease dynamics varying across geography and demography. However, the mobility datasets have inherent issues of imprecision and of being high volume. This is compounded by the challenges of changing pharmacological and non-pharmacological context of contagion behaviour, geography, demography, public health strategies across the globe.

In this paper, we propose a comprehensive framework, SCORE, to provide new analytical tools for public health strategy and planning. We also propose a novel data structure, DisCoUnt, which serves as a distributed uncertain trajectory index for moving objects as well as infection event data. We conduct extensive experiments to demonstrate the scalability of our query workflow for an infection risk measure over uncertain trajectories.

Keywords: location-based-service applications · mobile data · multi-dimensional index · Contact Tracing

1 Introduction

As of September 27, 2023, the cumulative positively identified infections due to COVID-19 (SARS-CoV- 2) had risen to 770,875,433 with a death toll of 6,959,316 as reported by WHO Coronavirus (COVID-19) Dashboard [55]. The original contagion strain showed rapid mutation and exhibited changes in infection propagation, latent period, symptoms and fatality proportions [45]. COVID-19 is not an isolated phenomenon. In fact, throughout human history a number of major pandemics affected millions of human beings, and thus changing the course of human civilization. For instance, the Plague known as the Black Death, during 1346 to 1353 resulted in the death of one third of the entire population of Europe [44]. Besides Covid-19, in recent years there have been a number of major outbreaks, including Middle East Respiratory Syndrome coronavirus (MERS-CoV) [39], the Ebola virus (EVD) [41], and the Zika virus (ZIKV) [42]. Epidemiologists predict that future outbreaks are highly likely [43].

During the course of a pandemic, such as Covid-19, public health agencies need to conduct real-time infection risk-assessment and develop mitigation

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 68–93, 2024.
https://doi.org/10.1007/978-3-031-63989-0_4

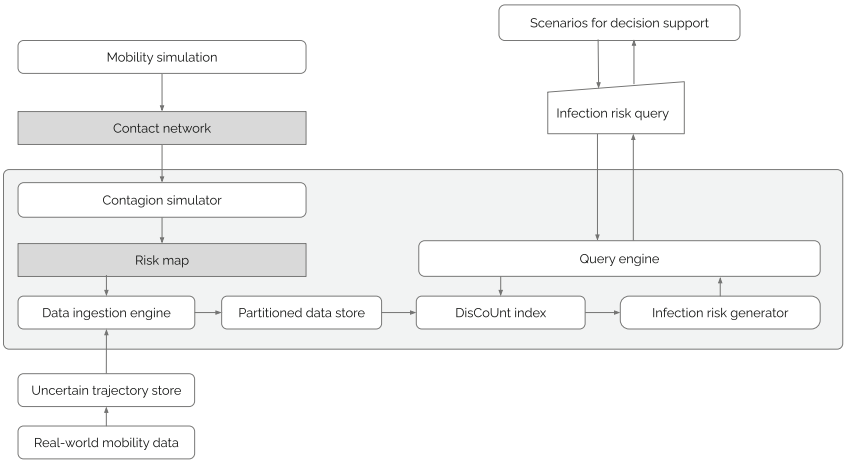


Fig. 1. Scalable Contagion Risk Estimation (SCORE) architecture

strategies utilising interventions to contain the risk of infection by obstructing the propagation of contagion. These strategies include effective pharmaceutical interventions like vaccination strategies as reported by Pritchard et al. [23], Makhoul et al. [24], Layan et al. [25] and Bian et al. [26]. Non pharmaceutical interventions (NPIs) have also been effective in mitigating public health risk. NPIs include *community masking*, *physical distancing*, *social bubbles*, *closures* or *lockdowns* as stated by Abueg et al. [31], Hens et al. [32], Lai et al. [33], Cheng et al. [34] and de Souza Santos et al. [35].

However, depending on the selection of intervention measures, the time of implementation and the coverage area of its effects, the efficacy of the intended outcome may exhibit significant variance. In addition to the technical complexities, executing these interventions requires mobilisation of large amount of resources and has psycho-social consequences. Therefore it is of utmost importance for public health policy makers to be informed with optimal timely intervention strategies for a given scenario. And more often than not this involves determining short-term estimates for multiple different mitigation strategies before context drifts farther away from initial model assumptions. Contagion modelling has been an integral part of any decision support system for public health to inform their strategies. Contagion modelling utilising ordinary differential equation (ODE) based disease propagation models abstracts the infection propagation of contagion over groups of individuals, which is computationally fast but does not capture spatio-temporal dynamics and demographic heterogeneity. On the other hand, there are individual level agent based infection propagation models, where each human's movement is simulated over a real world geo-spatial area but it is computationally expensive to model both movement and contagion propagation. A mid-way approach between these two is to extract a contact network of individuals at POIs based on detailed simulation

and simulate contagion propagation over said network, which preserves the heterogeneity of a real-world community. In other words, contagion modelling may be considered as construction of a risk of infection map across a geography with reasonable spatial and temporal resolution. We have witnessed efforts in identification of real-world contact networks with large scale contact tracing efforts from public health agencies.

It has been well established that contact among people due to human mobility plays a significant role in epidemic spreading. The issue of contagion risk identification and assessment at scale may be formulated as a problem of identifying sufficient spatio-temporal overlaps between moving objects. In other words, for a given geographic area and overarching temporal bound, we need to filter moving objects that may have the opportunity to transfer infection inducing viral load. For e.g. Public Health of Canada defined a temporal overlap of 15 min over 24 h within a proximity of 2 m for contact identification and management purposes [2]. Utilisation of mobility data to assess contagion risk has been explored in the following works [3–7]. However, these approaches have limited applicability due to imprecise nature and very high volume of spatio-temporal data. Cicala et al. [8] note that proximity based contact tracing (PCT) protocols, which work by logging distances between smartphones with PCT apps, may estimate the distance with a degree of uncertainty. For one, Bluetooth Low Energy (BLE)-based PCT techniques use the signal strength for estimating the distance and two, the individuals may be carrying their smartphones further from their position, for instance in their luggage instead of in their pockets.

To address the limitations of existing approaches, we propose a decision support system for public health agencies called SCORE (Scalable COntagion Risk Estimator). The overall architecture of SCORE is shown in Fig. 1. The aims of SCORE include providing decision support to public health efforts for estimating risk posed by a contagion, for a specific socioeconomic and geographic context with respect to changing intervention scenarios through individual level models in combination with real-world trajectory data. Our framework includes a contagion simulator (as shown in Fig. 1) that captures real-world spatio-temporal complexities of modelling a rapidly changing contagion with respect to intervention strategies and geo-spatial variance.

The key contributions of this paper are as follows:

- We propose a novel, easy to parallelise distributed spatio-temporal index DisCoUnt, to enable efficient uncertain trajectory indexing as well as infection data.
- We illustrate how infection data may be dis-aggregated into specific points of interests (POIs) of a geography utilising contagion simulators.
- Our experimental evaluation demonstrate the efficacy of DisCoUnt in a distributed environment.

The rest of the paper is organized as follows. In Sect. 2, we outline the research questions. We describe our overall architecture of SCORE in Sect. 3. We discuss the essential elements of a contagion simulator in Sect. 4. In Sect. 5 we describe in-depth our infection risk query processing and distributed index. In Sect. 6

we describe our experimental evaluation. In Sect. 7 we discuss the limitations of this paper. In Sect. 8, we present related work. Finally, we draw conclusions in Sect. 9.

2 Research Questions

In this paper we focus on the following three questions:

RQ1 How could we combine analyses of infection data and trajectory with an appropriate spatial resolution?
RQ2 Given an appropriate combination of trajectory and infection data, how could we define a risk measure for a given moving object?
RQ3 Given the definition of a risk measure of a moving could, how might we efficiently compute it?

3 SCORE Architecture

As shown in Fig. 1, SCORE relies on two sources of data, namely the *Contact network* and *Uncertain trajectory data* pre-processed from *Real world mobility data*. It utilises a network based contagion simulator with an appropriate disease model and the input *Contact network* to produce a *Risk map* for a community. The *Data ingestion engine* is responsible for distributing *Risk map* and *Uncertain trajectory store* into the available nodes in the distributed network. The proposed Distributed Contact detection over Uncertain trajectory *(DisCoUnt)* index is responsible for distributed execution of a *Infection risk query* and gathering of filtered data points. The data points are passed to *Infection risk generator* for a final output, which is passed back to *Query engine* and finally back to the user.

In the next sections we describe each component of our system.

3.1 Mobility Simulation

Human mobility simulation entails analysis of available mobility trace datasets derived from cellular network, location based services, public transportation and survey to produce a model that can output reasonable movement patterns of a community. For example, Toronto Area Scheduling Model for Household Agents (TASHA) is a prototype activity scheduling micro-simulation model to generate a typical 24-hour workday or *schedule* for individuals of a household proposed by Miller et al. [46]. The model has been subsequently validated by Roorda et. al [47] for the Greater Toronto Area, Canada for the 1996 travel survey data from which the model is derived from and the comparison with forecast for 2001 travel survey data.

3.2 Contact Network

A contact network with respect to network-based individual-level contagion model is an abstraction that captures the co-locations of individuals for a given time-step. Here each vertex is an individual and each edge represents an event of shared time and space. A representative example is show in Fig. 2.

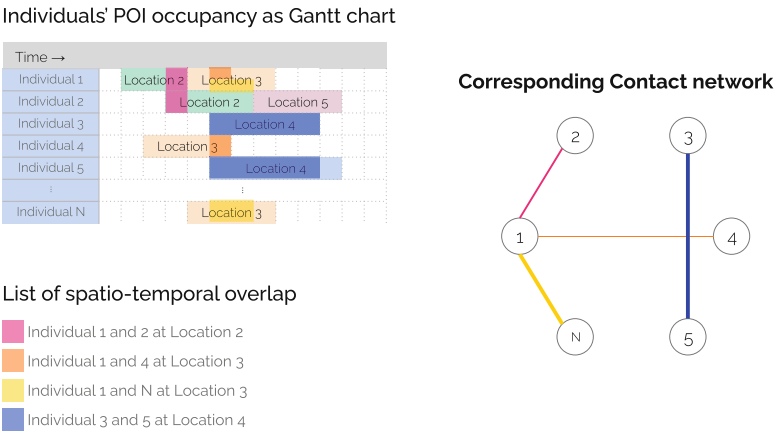


Fig. 2. Generation of contact network for 6 individuals from their POI occupancy data

3.3 Contagion Simulator

A contagion simulator that operates on contact networks, infers the propagation of a contagion based on a disease model, intervention strategies and underlying health factors associated with the demography and POIs contained within the geography. Their output includes time series data of infection events across POIs. We describe parts of a typical network-based contagion simulator relevant to this framework in Sect. 4.

3.4 Risk Map

Based on the output of a calibrated contagion simulator, the infectious individuals' POI occupancy data is dis-aggregated across the spatial bounds of a POIs as discrete tiles.

3.5 Uncertain Trajectory Store

Given any two-dimensional trajectory data as a sequence of observations $o_1, o_2, ...o_n$, where each o_i is position data and the associated timestamp: $o_i = (x_i, y_i, t_i)$, it may be encoded as uncertain trajectory segments whenever appropriate. In our framework we choose a shape-based encoding of uncertainty of moving objects called *Beads* and *Necklaces*.

A **Bead** is a shape-based model for an uncertain trajectory as proposed by Trajcevski et al. [10]. The construction of a Bead relies on a pair of observation (o_i, o_{i+1}) of a moving object m_k and a maximum possible velocity m_k may attain for the particular trajectory segment v^i_{max} dictated by real-world constraints.

Mathematically, a Bead B_i may be constructed from an observation pair (o_i, o_{i+1}) is an ellipse with the following equation:

$$\frac{(2x - x_i - x_{i+1})^2}{{v^i_{max}}^2(t_{i+1} - ti)^2} + \frac{(2y - y_i - y_{i+1})^2}{{v^i_{max}}^2(t_{i+1} - ti)^2 - (x_{i+1} - x_i)^2 - (y_{i+1} - y_i)^2} = 1 \quad (1)$$

A sequence of Beads B_1, B_2, ... B_n created from a sequence of observations o_1, o_2, ... o_{n+1} is referred to as a **Necklace**. In [10] the authors recommend the following strategies to approximate Beads and Necklaces with reasonable geometric constraints to balance the conflict between imprecision and encoding and analysis efficiency:

- **Large Bound** (LB): The entirety of a necklace is bounded by a single minimal vertical cylinder.
- **Individual Bead Bound** (IBb): Each Bead B_i is approximated by a circle with their radius r_i as the semi-major axis of the underlying Ellipse. The r_i is dependent on the choice of v^i_{max}, the maximum possible velocity between observations o_i and o_{i+1}.
- **Uniform Bead Bound** (UBb): Similar to IBb, each Bead B_i is approximated by a circle with their radius r_i as the semi-major axis of the underlying Ellipse. But the choice of v^i_{max}, is constant as the maximum possible velocity between any two points o_i and o_{i+1}.

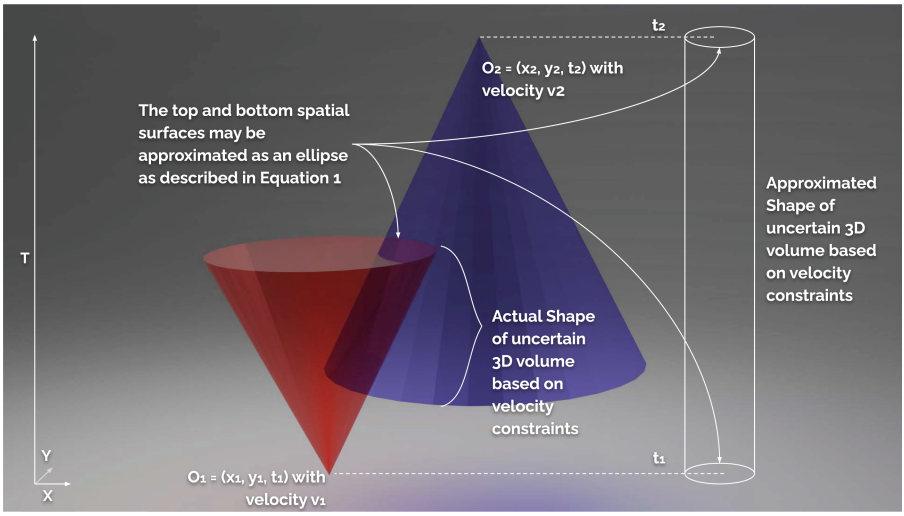


Fig. 3. Approximating trajectory segments as ellipses extruded over T axis as proposed by Trajcevski et al. [10]

Please note, even with the least lossy bead encoding such as, **Individual Bead Bound,** the spatial bound of a moving object can be approximated by an ellipse across the time bound of two subsequent observations, as shown in Fig. 3. For a detailed consideration of pruning strategies we refer the reader to Section III-B of the paper by Trajcevski et al. [10]. For example of IBb Bead

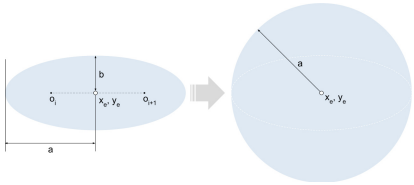


Fig. 4. Approximating spatial bound of Beads from ellipses to circles as proposed by Zhang et al. in [1]

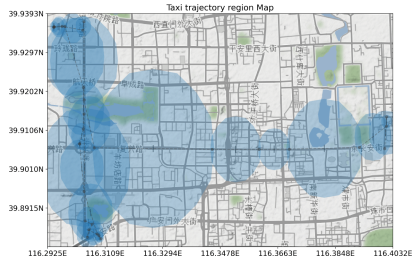


Fig. 5. A necklace with approximated circular beads distorted on Equirectangular projection with GIS mobility trace data provided by Yuan et al. [11]

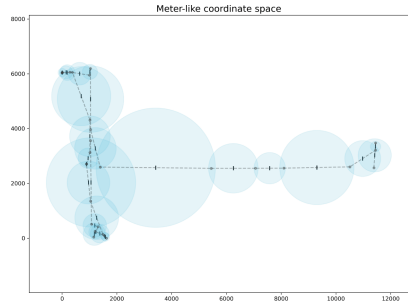


Fig. 6. Same necklace as shown in Fig. 5 projected on Cartesian plane

Necklace based on real-world trajectory data as provided by Yuan et al. [11], please refer to Figs. 5 and 6. Note that, although the spatial bounds of beads are approximated as circles from ellipses as shown in Fig. 4, they appear as ellipses due to distortions introduced by cartographic projection.

3.6 Combining Risk Map and Uncertain Trajectory Store

Taking a similar approach to model spatial-congestion in different POIs as illustrated by Alix et al. [21], we chose the discrete tiles for infectious individual POI occupancy to be hexagonal with appropriate resolution for humans. Next we approximate the spatial bounds of hexagons with its corresponding circumscribed circle or circumcircle to get a cylinder in $X \times Y \times T$ space. This is a data structure equivalent to our encoding of an uncertain trajectory. Thus, a Bead may be utilised to store both data points of real-world mobility as well as space-time occupancy of infectious individuals in a POI. Please note, although the trajectory data of humans are technically in 3 spatial and 1 temporal dimensions, most of human activities may be considered to be taking place on 2D planes with connectors like stairs or elevators and therefore without loss of generality we may consider the human trajectories as points in $X \times Y \times T$ space.

3.7 Data Ingestion Engine and Partitioned Data Store

The *Data ingestion* component of the framework is responsible for creating a *Partitioned data store* for data points of both *Riskmap* and *Uncertain trajectory store*. Two partitioning heuristics are discussed in Sect. 5.

3.8 Infection Risk Query

This query involves the identification of infection risk associated with a moving object present in the *Uncertain trajectory store*.

For example, given a scenario "How risky is to visit this mall?" one may translate it into a SQL-like query as follows:

```
INFECTION_RISK(*) FROM UNCERTAIN_TRAJECTORY_STORE, SIMULATION_STORE,
WHERE RISK_MAP = SIMULATION_STORE.RISK_MAP,
UNCERTAIN_TRAJECTORY_BEADS = UNCERTAIN_TRAJECTORY_STORE.BEADS AND
POLYGON == "MALL_01" AND
TIME BETWEEN YESTERDAY AND TODAY
```

The details of infection risk query processing is described in Sect. 5.

3.9 Query Engine

A query engine is the component responsible for interpreting a user's query into a query plan along with execution details taking into account the indices and possible speedups. Here, the proposed query engine augment interprets a spatio-temporal query and produces a query execution plan using *DisCoUnt index*.

3.10 DisCoUnt Index

This is our proposed novel distributed index, which facilitates efficient search and retrieval from *Partitioned data store*. In particular, the index is designed to support scalable evaluation of infection risk queries. This is described in Sect. 5.2.

3.11 Infection Risk Generator

This component computes the probability of infection risk with respect to the retrieved data points of a *Infection risk query*. Details of infection risk measure and its computation is described in Sect. 5.1.

4 Contagion Simulator

A detail discussion of contagion simulation in general or even network-based stochastic contagion simulation in specific is out of scope for this paper. Therefore, here we briefly discuss how an individual level network based geo-spatial contagion simulator combines demography data, geo-spatial data and disease models.

4.1 Combining Demography and Geo-Spatial Data

Due to privacy and security issues being among the primary concerns, the resolution of trajectory or trajectory traces of humans is limited and may not be suitable to determine the spatio-temporal overlaps required for infection propagation. Therefore one technique to generate human mobility data that captures the variations of the age distribution and the network of POIs is to simulate it via human mobility models. A simplified description of this process is outlined below.

Step 1 First, a statistically similar individual level population data is synthesised based on the available aggregated census and labour data for a community.

Step 2 In parallel, road-network and POI data are extracted for the same community from sources like OpenStreetMaps.

Step 3 A mobility simulation is performed with a relevant time-step, say 24 h, with the data obtained in **Step 1** and **Step 2** to obtain movement patterns of the individuals.

This movement pattern data is transformed into individual occupancy data consisting of POIs and finally transformed into a contact network.

4.2 Infectious Disease Modelling as Compartments

Over the years, several infectious disease models have been proposed. According to Keeling et al. [48], in these models, individuals of a population are put into compartments in relation to the contagious disease, which are denoted by labels such as, *Susceptible*: individuals who may get infected, *Infectious*: individuals who may infect susceptible individuals, *Latent*: an individual who is infected but not yet infectious, *Removed*: Individuals who are not participating in the propagation of disease due to immunity or death and so on. At any given time, a person may be in exactly one of the compartments. One of the simplest of contagion models comprises of three compartments, namely Susceptible, Infectious and Recovered and is called the SIR model.

For our simulation purview, we invert this perspective. That is, we do not consider mixing of groups of individuals (as compartments) and investigating the parameters from the real world to fit into the measures of central tendencies. Instead we simulate the movement of the population to the best of our abilities and capture the POI occupation events as an emergent property of the simulation to generate the contact network. Then we simulate disease propagation on that network, without loosing individual variations of health statuses and keeping a portion of the geo-spatial heterogeneity.

4.3 Interpreting Infection Propagation

Irrespective of the design of an infectious disease model, it must always contain a mechanism of transition from a non-infectious state to an infectious state. As

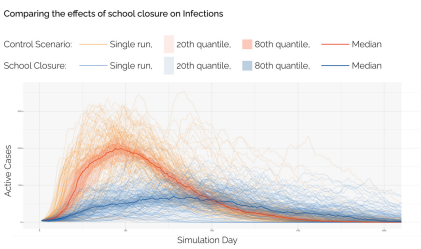


Fig. 7. Simulation of the effect of school closure on the infection propagation in a community from [38]

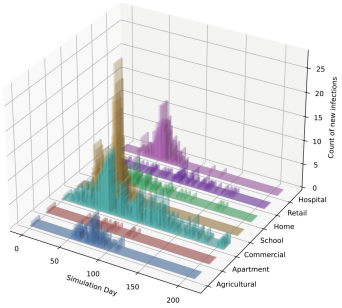


Fig. 8. Simulation of relative infection counts with respect to the venue types in a community from [38]

per the mechanistic model of viral load transmission by Challenger et al. [49], each infectious individual may be thought of as a emitters of air-born virus by virtue of exhalation. If and only if there is sufficient transfer of said virus (a.k.a viral load) from an infectious to susceptible individual, there is a possibility that, infection may spread to the susceptible individual. This interaction is modelled as a infection probability function f, which in the simplest case is proportional to the shared time t between a pair of susceptible and infectious individual in a POI.

4.4 Example Outputs of Contagion Simulation Without Calibration from [38]

As part of separate research activities on contagion simulation over real-world communities in [38] we have utilised CitiSketch am [56] to generate representative contact networks. The figures are example outputs of infection spread in a community of 4602 individuals from the Campbellton region of New Brunswick, Canada. The demography and labour data were procured from Statistics Canda [57] and the Road network and POIs from OpenStreetMap [58] respectively.

Figure 7 shows relative infection risk with school closure as a mitigation strategy. Figure 8 illustrates the relative infection risks for venue types.

Upon calibration, an individual level contagion simulator can provide POI level occupancy data of expected number of infectious individuals for an outbreak.

5 Infection Risk Query Processing

Before we begin to describe the query workflow in-depth, we answer **RQ2** with the following definitions.

5.1 Infection Risk Measure

Taking inspiration from Zhang et al. [1], we approach to define the probability of transmission of contagion from an infectious individual described by an occupancy encoding bead to a susceptible individual described by an uncertain trajectory encoding bead as follows:

We begin with the premise that two beads B_1 and B_2 describing object O_1 and O_2 respectively have spatio-temporal overlap with area of spatial intersection ρ_{12} and maximum possible time overlap of τ. Let us also assume that the probability of finding an object within the spatial bound defined by a bead is uniform and remains constant across time.

Definition 1 ***Probability of spatial contact.*** *This probability may be decomposed into three events as follows:*

- $P(E_1)$*: When sampled at random the probability of* O_1 *to lie within spatial intersection.*
- $P(E_2)$*: When sampled at random the probability of* O_2 *to lie within spatial intersection.*
- $P(E_3)$*: Given the a priori probabilities of* E_1 *and* E_2*, the probability that the sampled points are at most* δ *distance apart.*

Since the events are independent of each other the resulting probability is as follows:

$$P_{space}(B_1 \cap B_2) = P(E_1) \cdot P(E_2) \cdot P(E_3) \tag{2}$$

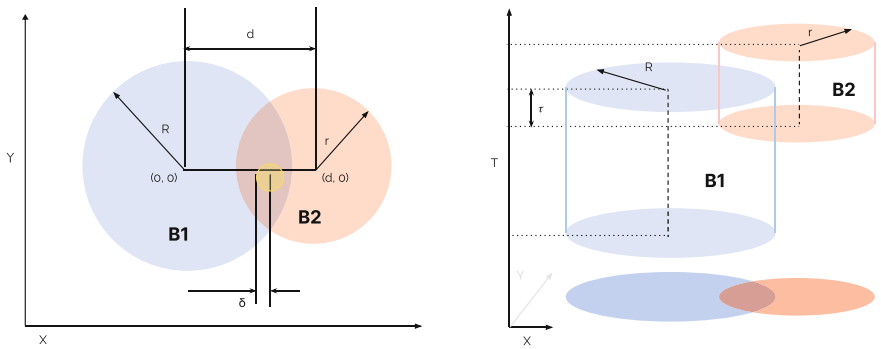


Fig. 9. Schematic describing probability of spatial contact and temporal overlap between beads

Without loss of generality, consider the two Beads in Fig. 9. Here, B_1 *is translated to* $(0, 0)$ *and has a radius of* R *with* B_2 *at* d *distance with radius* r*. Then, by construction, we have the following results:*

$$P(E_1) = \frac{\rho_{12}}{\pi R_1^2} \textit{ first object is on the overlapping area} \tag{3}$$

$$P(E_2) = \frac{\rho_{12}}{\pi r_1^2} \textit{ second object is on the overlapping area} \tag{4}$$

$$max(P(E_3)) = \frac{\pi\delta^2}{\rho_{12}} \textit{ both of them are "close enough"} \tag{5}$$

And the spatial area of intersection ρ may be computed as follows:

$$\begin{aligned}\rho_{12} &= r^2 \cos^{-1}\left(\frac{d^2 + r^2 - R^2}{2dr}\right) \\ &+ R^2 \cos^{-1}\left(\frac{d^2 + R^2 - r^2}{2dr}\right) \\ &- \frac{1}{2}\sqrt{(-d + r + R)(d + r - R)(d - r + R)(d + r + R)}\end{aligned} \tag{6}$$

Putting it all together, we have the following probability of spatial contact between the objects O_1 and O_2 described by two beads B_1 and B_2 with a threshold of minimum distance of effective contact as δ.

$$\begin{aligned}P_{space}(B_1 \cap B_2) &= \frac{\rho_{12}}{\pi R_1^2} \cdot \frac{\rho_{12}}{\pi r_1^2} \cdot \frac{\pi\delta^2}{\rho_{12}} \\ &= \frac{\rho_{12} \cdot \pi\delta^2}{\pi R_1^2 \cdot \pi r_1^2}\end{aligned} \tag{7}$$

Definition 2 ***Probability of infection propagation***
Let us also assume that this pair (O_1, O_2) is a pair of susceptible and infectious objects with a maximum possible temporal overlap be τ_{12}.

A mechanistic model of infection transmission may be generalised as a function of time overlap t between two individuals in close proximity with a minimum cutoff time t_{min}: $f(t, t_{min})$. Therefore, the final probability may be decomposed into two events as follows:

- *$P_{space}(B_1 \cap B_2)$: A priori probability of spatial contact.*
- *$P(transmit_{12})$: $f(\tau_{12}, t_{min})$*

Therefore the probability of infection from O_1 to O_2, for any O_I who is infectious and O_S who is susceptible is as follows:

$$\begin{aligned}P(O_I \xrightarrow{infects} O_S) &= P_{space}(B_1 \cap B_2) \cdot P(transmit_{12}) \\ &= \frac{\rho_{12} \cdot \pi\delta^2}{\pi R_1^2 \cdot \pi r_1^2} \cdot f(\tau_{12}, t_{min})\end{aligned} \tag{8}$$

Thus we have a risk measure for a single space-time overlap. When extended for an complete uncertain trajectory of a moving object with multiple beads we have the following definition.

Definition 3 ***Infection risk measure.*** *for a susceptible moving object O_i with a trajectory T_i may be defined as follows:*

- *Let T_i be decomposed into a Necklace of Beads as* $N_i = [B_{i1}, B_{i2}..., B_{ik}]$
- *Let the set of all infectious beads be* $S_{inf} = B_{inf1}, B_{inf2}, ...B_{infm}$
- *Then, we find the set of all the overlap of pairs of objects from* S_{inf} *and* N_i *as* $S_{overlap}$.
- *Any object* $s \in S_{overlap}$ *is a pair of beads* $(B_i, B_j) \mid B_i \in N_i, B_j \in N_{inf}$
- *Then to compute the overall infection risk of* O_i *with respect to* T_i*, we must compute every* $P(O_j \xrightarrow{infects} O_i)$ *for each* $(B_i, B_j) \in S_{overlap}$

5.2 DisCoUnt Index

To accommodate efficient search over a partitioned Bead store we took inspiration from UTM-tree proposed by Zhang et al. [1]. UTM-tree is constructed by extending a metric-index M-tree [12] by including the radius of a bead as part of the data point and including it in the metric distance function for space based pruning and including a range-tree, proposed by Bentley et al. [50] for time based pruning in a single traversal of the resultant tree. However, UTM-tree, like that of a M-tree relies on a bottom-up insertion approach. This step ensures a height-balance and overlap reduction of the nodes of the tree and therefore subsequent spatial indexing efficiency. Therefore, it is prudent to partition the trajectories according to children or sub-trees of a node to ensure efficiency in parallel or distributed context. However, the tree creation process is bottom up, resulting in shuffling of trajectories among partitions making it unsuitable for partition based scaling up as is. Also, UTM-tree, like M-tree, utilises ball-type partitioning that introduces overlaps between sibling nodes (nodes that are at the same depth) with overlaps that may not be effectively mitigated as it is dependent on stochastic selection of routing objects during tree creation. This in turn poses a non-deterministic challenge of selection of partitions for further data insertion, which may cause data replication across partitions raising memory cost of indexing to an unsatisfactory degree. Although, it is possible to partition the trajectory segments before constructing individual UTM-tree indices out of each partition, the resultant data structure would be a hybrid index, for which ensuring overall efficiency would become a multi-part challenge. Instead, we considered alternative candidates for indexing based on the systematic evaluation of metric-space indices by Chen et al. [13] of range query on main-memory. The highest ranking technique reported was Multiple Vantage Point tree (MVPT) [14], followed by Extreme Pivot Table (EPT*) [15], Bisector Tree (BST) [16] and Geometric Near-Neighbour Access tree (GNAT) [17] and its variants. Since MVPT and partitioning of BST and EPT also utilises ball partitioning, it posed similar challenges to that of an M-tree in a setting of a distributed cluster of nodes. To support efficient uncertain trajectory indexing and infection data, we propose a novel distributed spatio-temporal index DisCoUnt, Next we describe the components of this index.

The membership of Beads and DisCoUnt node, and their attributes from an example index instance are shown in Fig. 10.

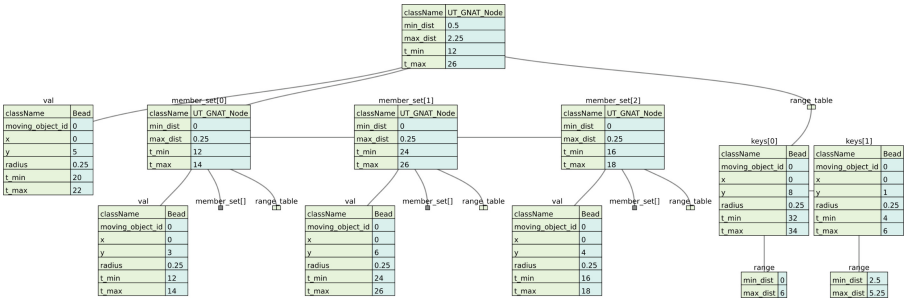


Fig. 10. An example DisCoUnt index instance visualised with JSON tree visualiser [36]

DisCoUnt Node Structure. The following parameters are sufficient for tree construction and pruning both spatial and temporal ranges.

- **Bead:** Contains full information of an Uncertain Trajectory segment:
  - m_{id}: moving object id
  - x, y, $radius$: spatial range of $radius$, centred on (x, y)
  - t_{start}, t_{end}: temporal range
- min_dist: minimum distance to its member set
- max_dist: maximum distance to its member set
- $*t_max$: maximum of all t_ends of its member set
- $*t_min$: minimum of all t_starts of its member set
- $member_set$: set of member DisCoUnt nodes
- $range_table$: set of member DisCoUnt nodes

The fields marked with an asterisk are introduced to keep a temporal range tree like secondary index structure for efficient temporal pruning during spatio-temporal range query computation.

5.3 Distance Function for Symmetric Range in Metric Space

Since we are dealing with distances between symmetric ranges of space, the distances between tWo Beads, B_1 and B_2 may be defined as Eqs. 9, 10 and 11 below and illustrated in Fig. 11:

$$c2c_dist(B_1, B_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{9}$$

$$min_dist(B_1, B_2) = max(0, c2c_dist(B_1, B2) - r_1 - r_2) \tag{10}$$

$$max_dist(B_1, B_2) = c2c_dist(B_1, B2) + r_2 \tag{11}$$

Here, min_dist is a stand-in for overlap detection and therefore has the "symmetry" property, i.e.:
$min_dist(B_1, B_2) = min_dist(B_2, B_1)$
But it, does not follow "triangle inequality" for all B_1, B_2.

Therefore, we need *max_dist*, which does follow "triangle-inequality": $max_dist(B_1, B_2) \geq max_dist(B_1, B_3) + max_dist(B_2, B_3)$.
This is true by construction as, $r_2, r_3 \geq 0$ and $c2c_dist(B_1, B_2)$ is Euclidean distance between (x_1, y_1) and (x_2, y_2).

Please note, *min_dist* does not strictly have the "identity" property, since any two overlapping pair of *Beads* yields a 0 and neither does *max_dist* as, $max_dist(B_1, B_2) > 0$ for all $r_2 > 0$.

For split point selection and membership assignment, we utilise the *min_dist* between DisCoUnt nodes and construct the DisCoUnt index. And, utilise *max_dist* for relative range based pruning and as a stand-in for maintaining minimum bounding circle for *member_set* of a *Bead*. Thereby making sure during each operation, only one kind of distance calculation is performed.

5.4 Algorithmic Complexity of a Single Index Construction

In this section we discuss the algorithmic complexity of our proposed index, particularly, the space and time complexity.

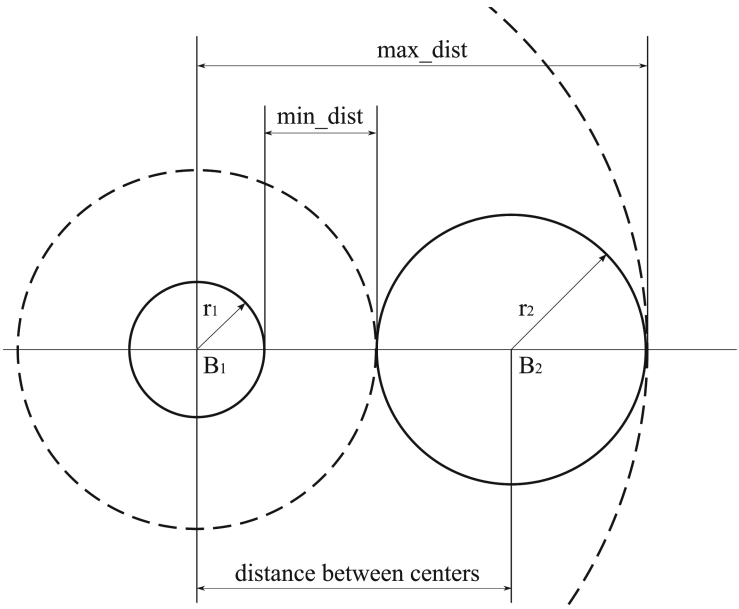


Fig. 11. Max and min spatial distances between two Beads

Space Complexity. Space complexity of DisCoUnt is the same as the data structure GNAT. It is as reported by the author of GNAT in [17] the space complexity is $O(nk_2 + Ns)$ where

1. n: number of nodes.
2. k_2: the average of the squares of the degrees.
3. N: number of data points (in a partition).
4. s: maximum space requirement for a data point.

Time Complexity. Time complexity of DisCoUnt construction is the same as GNAT. The time complexity for a completely balanced GNAT [17], is: $O(Nklog_kN)$, where k is the average degree (N/n).

Please note, our approach does not change the number of distance calculations. Hence, the O of space and time complexity of index building is the same in spite of accounting for maintaining an orthogonal range tree like structure for temporal comparison.

5.5 Bead Overlap Detection Query

The spatio-temporal range search starts from the root node of a DisCoUnt index; it utilises the pre-computed distance and temporal range information present the current node and its siblings *range_table* with respect to the query bound to prune the sub-trees. Please note, bead overlap detection through a spatio-temporal range search is accomplished by setting the querying each Bead of a susceptible moving object and looking for overlaps with an infectious moving object or vice versa.

This process is detailed in Algorithm 1.

The double-range search is an extension of the simplified algorithm described in Sect. 4.1 of the author of GNAT in [17] and re-framed by authors of EGNAT [52] in Algorithm 1. Since we added one additional computation per candidate solution and pivot point, it does not adversely effect the time complexity of search. Furthermore, Kimmo Fredriksson in their pre-print submission [51] states that for range-searching in a GNAT, for any surviving non-pruned pivot point without further child nodes, the search algorithm takes a $O(k)$ distance comparison in average, where k is the average degree. In our case it would be another $O(k)$ for the time overlap comparison, but given sufficiently larger k, the complexity of searching within a DisCoUnt node will still be $O(k)$.

Algorithm 1: Range Search to detect bead overlap

Input : P, query
Output: R : set of overlapping Beads

Let P be the list of root level DisCoUnt nodes
query contains both spatial and temporal bound to search
Randomly select a split point $p \in P$
Check if p spatio-temporally overlaps with *query*, then add to R
$d \leftarrow dist(query, p)$ $r \leftarrow query.spatial_range$ $relative_range \leftarrow [d - r, d + r]$
$t \leftarrow query.temporal_range$ **while** $q \in P - p$ **do**
    if $relative_range \cap range(p, D_q) = \phi$ **then**
        Remove q from P
                                                    // Space Pruning
    if $t \cap [q.t_min, q.t_max] = \phi$ **then**
        Remove q from P
                                                    // Time Pruning
Double Range Search for each D_p for p left in P
return R

5.6 Partitioning Strategies

To gauge the efficiency gain of the index, the following partitioning strategies were considered:

1. **Clustered:** Beads closest to a split point were assigned to the same index.
2. **Round-robin:** Beads that would have been part of the same partition in the clustered strategy are allocated across all available partitions in round-robin fashion.
3. **Random:** Random allocation of Beads for comparison.

5.7 Infection Risk Query Processing

For SCORE, we build two DisCoUnt instances, one for indexing uncertain trajectories and the other for indexing risk maps. Since every infection risk query has a implicit spatio-temporal range, of say a city or a town, depending upon the scope of the risk map, we decompose the query as follows:

Step 1 Gather all the trajectory Beads in the overarching query range of the relevant moving object id i as S_i.
Step 2 Gather all the risk map Beads in the same range S_inf.
Step 3 Pass the sets to infection risk generator to detect pair-wise overlap
Step 4 Compute probability of infection propagation for each overlap and return the result set to the user.

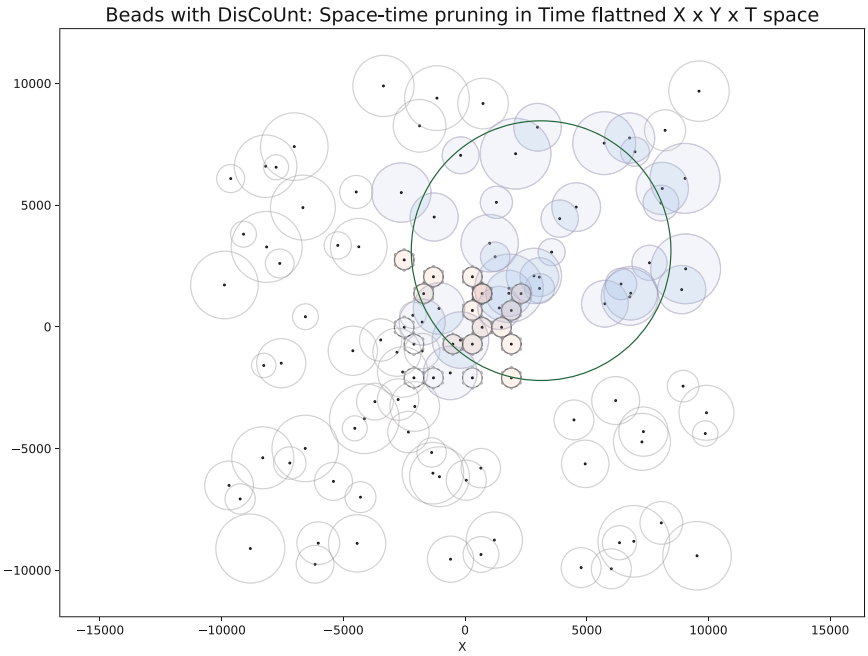


Fig. 12. Schematic diagram describing the gathering of trajectory Beads in blue and risk Beads in red within the scope of range query outlined in green (Color figure online)

A simplified spatial description of **Step 1** and **Step 2** is presented in Fig. 12, where the overarching spatial query range is outlined in green. The hexagonal tiled occupancy data has been approximated to its corresponding circle. The trajectory beads are constructed as Individual Bound Beads to encode movement with variable velocity in between trajectory observations. The result set S_i and S_inf are passed to the infection risk generator to detect pair-wise overlap. This is the proposed solution to **RQ3**.

6 Experimental Evaluation

In this section DisCoUnt index is evaluated, firstly in a single machine and secondly on a cluster of machines. We chose to perform these experiments on synthetic data for both trajectory and infection risk map. The synthetic data was generated to evaluate effects of varying *data distribution*, *data volume*, *data partitioning*, and *query selectivity* on the proposed index that we expect to find in real-world mobility data and risk maps.

6.1 Experiment Setup

The components of SCORE system and DisCoUnt index were implemented in Python 3. First, the experiments involving standalone DisCoUnt index were conducted on a single machine with Intel(R) Core(TM) i7-10875H CPU @ 2.3 GHz running Ubuntu 20.04.2 LTS x86_64-bit OS with 32 GB RAM. An uncertain trajectory data store was generated for testing by increasing data volumes from 1K to 1M Beads (Fig. 13).

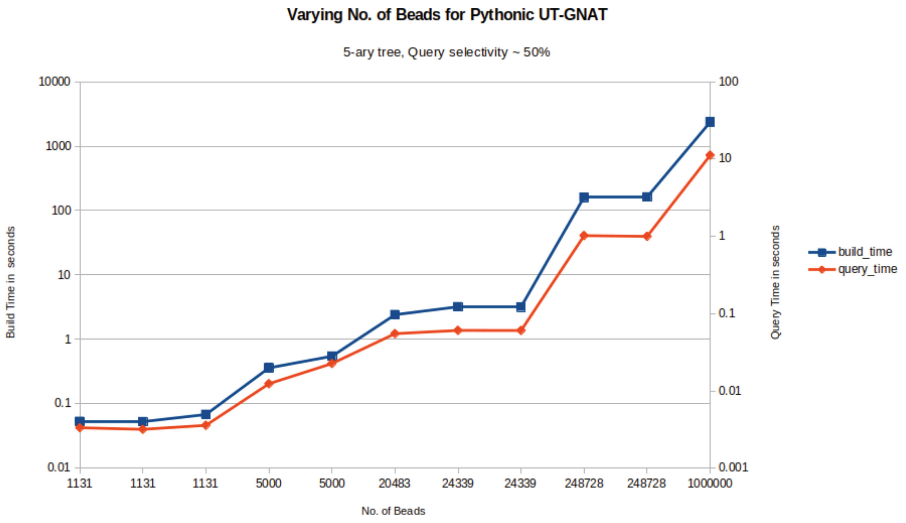


Fig. 13. Time complexity of single DisCoUnt index. Please note that there are separate logarithmic time axes for building and querying on the left and right side respectively

The results of searching in parallel DisCoUnts for partitioned data store in a local machine shows clear efficiency gain over a single index as shown in Fig. 14.

Second, a two instance DisCoUnt index was set-up over a partitioned bead store with an associated risk map of 20% of total number of trajectory Beads in a cluster of 4 machines with Intel(R) Core(TM) i5-2400 CPU @ 3.10 GHz running Ubuntu 14.04.5 LTS x86_64-bit OS with 8 GB RAM. The machines were networked via wired Ethernet TP-LINK(R) TL-SG1005D Gigabit Switch. Initial implementation of DisCoUnt was extended for processing of query in a distributed environment utilising Dask [37]. Specifically, the *Actor* API of Dask, which enables stateful computation at worker nodes. Please note that for the scope of this paper, we conclude our evaluation at gathering the relevant Beads from the partitioned store and do not proceed with computing the overlaps and the subsequent probabilities. Since the Infection risk measure calculation as

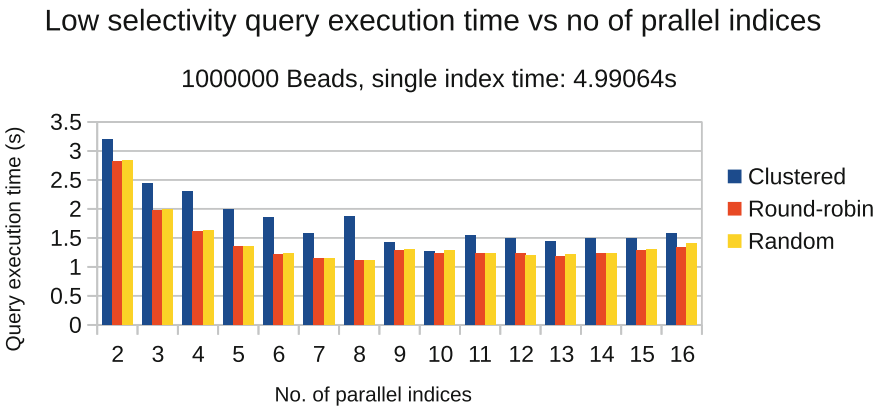


Fig. 14. Low selectivity, parallel query on 1000000 Beads showing expected speedup on increasing partitions

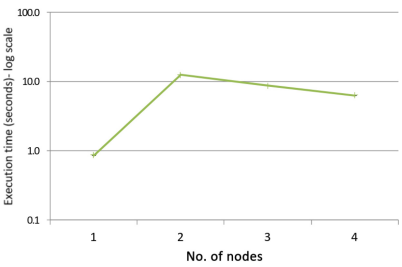


Fig. 15. Query time vs. number of nodes in cluster for 100K Beads

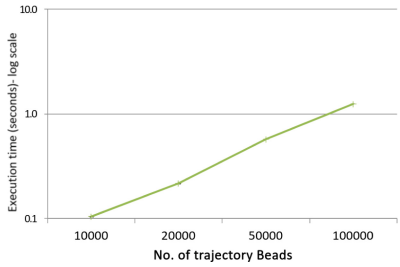


Fig. 16. Query time vs. No of Beads

described by Definition 3 is a sum of multiplications with a sampling of PRNG, it is expected to not exceed the delay of data transfer over network and affect the evaluation metrics.

6.2 Effect of Data Distribution

In this experiment, we vary the number of nodes as 1, 2, 3 and 4, and we report the corresponding query execution times in Fig. 15. The figure shows an increase in efficiency of query execution with an increase in the number of nodes. However, it is not clear how many nodes may be needed to outweigh the delays introduced due to data transmission over a network.

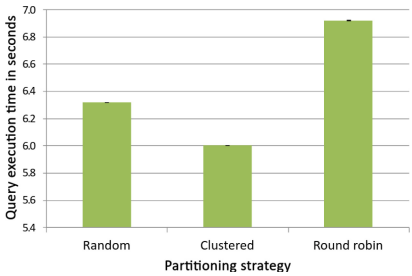


Fig. 17. Query time vs. Partitioning strategy for 100K Beads across 4 nodes

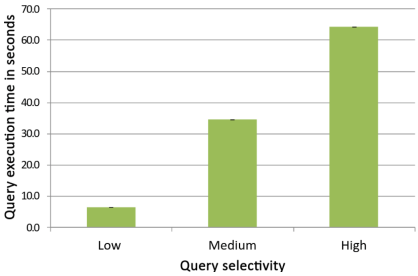


Fig. 18. Query time vs. Query selectivity of High (0.6), Medium (0.3) and Low (0.04) for 100K Beads across 4 nodes

6.3 Effect of Data Volume (Number of Beads)

We vary the number of beads from 10k to 100k and measure the query execution time. Figure 16 shows expected trend in increase in query execution time with respect to data volume, similar to that of a non-networked implementation of DisCoUnt.

6.4 Effect of Partitioning Heuristics

For this experiment, the trajectory Beads and the risk maps beads were partitioned separately with respect to two sets of split points. We evaluate the impact of the partitioning strategies (i.e. clustered, random and round robin, mentioned in Sect. 5.6) on the query execution time. The result in Fig. 17 suggests that the clustering partitioning scheme works slightly better than random allocation of Beads and the round-robin scheme performs worse.

6.5 Effect of Query Selectivity

In this experiment, we vary the query selectivity and evaluate its impact on the query execution time keeping the data volume constant at 100K Beads. We used three selectivity values: low, medium and high with selectivity of 0.04, 0.3 and 0.6 respectively. Therefore the query results of low medium and high selectivity would contain 400 Beads, 30K Beads and 60K Beads respectively. Figure 18 demonstrates that query execution time is linearly proportional to volume of result. This is to be expected for a networked environment of our experimental setup where over 90% of the total query execution time is spent transferring query result from worker nodes.

7 Discussions and Future Work

In this section we discuss the limitations of the experimental evaluation, the scope of SCORE, and future lines of enquiry.

- Our experiments on the distributed DisCoUnt index demonstrates that it has the potential to scale similar to its parallel variant as shown in Fig. 14. However, as shown in Fig. 15 we will need additional experiments to determine optimum multi-machine configuration.
- The experiments are performed on synthetic data that varies query load and data volume. Although it is representative of worst case scenarios, performing similar suite of experiments on real-world trajectory store remains as a future work.
- At a framework level, comparing the accuracy and efficiency of SCORE would require multiple data sources for a well-documented outbreak of a particular contagion in a specific community. It would also require a calibrated contagion simulator. Calibration of a contagion simulator is out of scope of this paper and by extension a full framework evaluation of SCORE remains as future work.

Please note, in this paper we focus on a "uncertain proximity detection" framework and the key definitions, algorithms, and data structures to compute it in a distributed context. This enables the proposed SCORE architecture to use outputs of any mechanistic or proximity based disease modelling system.

8 Related Work

The literature surveyed during the course of this research were found to focus on a part of the problems we aim to tackle through the SCORE architecture. In this section, we mention some of the relevant research.

Authors of [3–5] and [6] have utilised mobility data to produce outputs analogous to the proposed *Risk map*, however the spatial resolution of their analysis spans areas as large as entire cities and districts. Hence, these models do not incorporate individual level mobility. Alix et al. [21], also represents single POIs with multiple hexagonal tiles, as in SCORE. They also assign spatial dispersion within a POI with respect to different categories of POI. Similarly, Anastasiou et al. [7], also spatially divide a POI into square tiles rather than hexagonal tiles. They run micro-simulation of expected movement of individuals within a POI with respect to ingress, exists and other architectural features. They aggregate them to produce regions of high-congestion and assign predictive risk scores future trajectory plans. However, both risk measures only take into account the occupancy data of individuals and not the disease dynamics of the contagion question, nor the expected changes due to intervention measures.

An et al. [53] explore agent based contagion simulation but they consider their spatial boundaries to be at the scale of countries and the co-location data was inferred from synthetic contact networks instead of real-world mobility data. Lee et al. [54] propose an efficient graph transformation to simulate reactive and preventative vaccination strategies. Their spatial resolution was of an administrative division of a US state and their co-location graph of choice was of routers and P2P file sharing network. We believe that approaches like these would benefit from distributed index, DisCoUnt proposed by us. At the time of writing this

manuscript, we did not find any previous work that integrates calibrated individual level agent based contagion modelling using real-world mobility data, with uncertain trajectory analysis to assess infection risk as an overarching framework.

9 Conclusion

In this paper, we propose a holistic framework called SCORE that combines simulated disease propagation model with real-world trajectory data to facilitate infection risk analysis for a community. We formally define a probability measure of spatial contact given a pair of uncertain trajectory segments. We combine this probability of spatial contact with a mechanistic model of infection transmission to define the probability of infection risk. We propose a novel index DisCoUnt, to efficiently search and retrieve data points to calculate infection risk measure for a moving object given their trajectory. Finally, we demonstrate with our experimental results that ours is a viable approach for computing infection risk measure using a distributed data store. However, our approach is incumbent on the calibration of an agent based contagion simulator, which is a non-trivial task. Our future directions include the exploration of different partitioning techniques and conducting a comparative evaluation of alternate candidates for distributed spatio-temporal indices.

Acknowledgements. We thank Bilal Husain, Gaia Noseworthy, Jarod Kelly, and Robert Santacruz, along with Dane Sheppard of The Black Arcs for their inputs and inspiration. We also thank Suvam Kumar Das a member of our Big data systems and analytics lab for his insightful discussions on metric indices that helped in planning early research activities. Finally we thank the reviewers for their comments that improved this paper.

References

1. Zhang, X., Ray, S., Shoeleh, F., Lu, R.: Efficient contact similarity query over uncertain trajectories. In: EDBT, pp. 403–408 (2021)
2. Canada, P. Public health management of cases and contacts associated with COVID-19 - Canada.ca. (2021). https://www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/health-professionals/interim-guidance-cases-contacts.html. Accessed 19 May 2023
3. Rambhatla, S., Zeighami, S., Shahabi, K., Shahabi, C., Liu, Y.: Toward accurate spatiotemporal COVID-19 risk scores using high-resolution real-world mobility data. ACM Trans. Spat. Algorithms Syst. (TSAS). **8**, 1–30 (2022)
4. Zeighami, S., Shahabi, C., Krumm, J.: Estimating spread of contact-based contagions in a population through sub-sampling. ArXiv Preprint ArXiv:2012.06987 (2020)
5. Kuddus, M., Rahman, A.: Analysis of COVID-19 using a modified SLIR model with nonlinear incidence. Results Phys. **27**, 104478 (2021)

6. Chiang, W., Liu, X., Mohler, G.: Hawkes process modeling of COVID-19 with mobility leading indicators and spatial covariates. Int. J. Forecast. **38**, 505–520 (2022)
7. Anastasiou, C., Costa, C., Chrysanthis, P., Shahabi, C., Zeinalipour-Yazti, D.: ASTRO: reducing COVID-19 exposure through contact prediction and avoidance. ACM Trans. Spat. Algorithms Syst. (TSAS) **8**, 1–31 (2021)
8. Cicala, F., et al.: PURE: a framework for analyzing proximity-based contact tracing protocols. ACM Comput. Surv. (CSUR) **55**, 1–36 (2021)
9. Pfoser, D., Jensen, C., Theodoridis, Y., et al.: Novel approaches to the indexing of moving object trajectories. In: VLDB, pp. 395–406 (2000)
10. Trajcevski, G.: Uncertainty in spatial trajectories. In: Zheng, Y., Zhou, X. (eds.) Computing With Spatial Trajectories, pp. 63–107. Springer, Cham (2011). https://doi.org/10.1007/978-1-4614-1629-6_3
11. Yuan, J., Zheng, Y., Xie, X., Sun, G.: Driving with knowledge from the physical world. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 316–324 (2011)
12. Ciaccia, P., Patella, M., Zezula, P.: M-tree: an efficient access method for similarity search in metric spaces. VLDB **97**, 426–435 (1997)
13. Chen, L., et al.: Indexing metric spaces for exact similarity search. ACM Comput. Surv. (CSUR) **55**, 1–39 (2020)
14. Bozkaya, T., Ozsoyoglu, M.: Distance-based indexing for high-dimensional metric spaces. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, pp. 357–368 (1997)
15. Chen, L., Gao, Y., Zheng, B., Jensen, C., Yang, H., Yang, K.: Pivot-based metric indexing. VLDB Endow. (2017)
16. Kalantari, I., McDonald, G.: A data structure and an algorithm for the nearest point problem. IEEE Trans. Softw. Eng. 631–634 (1983)
17. Brin, S.: Near neighbor search in large metric spaces. VLDB **95**, 574–584 (1995)
18. Mahmood, I., et al.: FACS: a geospatial agent-based simulator for analysing COVID-19 spread and public health measures on local regions. J. Simul. **16**, 355–373 (2022)
19. Horni, A., Nagel, K., Axhausen, K.: The Multi-Agent Transport Simulation MATSim. Ubiquity Press (2016)
20. Müller, S., et al.: Predicting the effects of COVID-19 related interventions in urban settings by combining activity-based modelling, agent-based simulation, and mobile phone data. PLoS ONE **16**, e0259037 (2021)
21. Alix, G., Yanin, N., Pechlivanoglou, T., Li, J., Heidari, F., Papagelis, M.: A mobility-based recommendation system for mitigating the risk of infection during epidemics. In: 2022 23rd IEEE International Conference on Mobile Data Management (MDM), pp. 292–295 (2022)
22. Dong, E., Du, H., Gardner, L. An interactive web-based dashboard to track COVID-19 in real time. Lancet Infect. Dis. **20**, 533–534 (2020). https://www.sciencedirect.com/science/article/pii/S1473309920301201
23. Pritchard, E., et al.: Impact of vaccination on new SARS-CoV-2 infections in the United Kingdom. Nat. Med. **27**, 1370–1378 (2021)
24. Makhoul, M., et al.: Epidemiological impact of SARS-CoV-2 vaccination: mathematical modeling analyses. Vaccines **8**, 668 (2020)
25. Layan, M., et al.: Impact of BNT162b2 vaccination and isolation on SARS-CoV-2 transmission in Israeli households: an observational study. Am. J. Epidemiol. **191**, 1224–1234 (2022)

26. Bian, L., et al.: Effects of SARS-CoV-2 variants on vaccine efficacy and response strategies. Expert Rev. Vaccines **20**, 365–373 (2021)
27. Glasser, J., Feng, Z., Omer, S., Smith, P., Rodewald, L.: The effect of heterogeneity in uptake of the measles, mumps, and rubella vaccine on the potential for outbreaks of measles: a modelling study. Lancet. Infect. Dis. **16**, 599–605 (2016)
28. Killingley, B., et al.: Safety, tolerability and viral kinetics during SARS-CoV-2 human challenge in young adults. Nat. Med. **28**, 1031–1041 (2022)
29. Klomjit, N., et al.: COVID-19 vaccination and glomerulonephritis. Kidney Int. Rep. **6**, 2969–2978 (2021)
30. Puhach, O., et al.: Infectious viral load in unvaccinated and vaccinated individuals infected with ancestral, Delta or Omicron SARS-CoV-2. Nat. Med. 1–10 (2022)
31. Abueg, M., et al.: Modeling the effect of exposure notification and non-pharmaceutical interventions on COVID-19 transmission in Washington state. NPJ Digit. Med. **4**, 1–10 (2021)
32. Hens, N., Vranck, P., Molenberghs, G.: The COVID-19 epidemic, its mortality, and the role of non-pharmaceutical interventions. Eur. Heart J. Acute Cardiovasc. Care **9**, 204–208 (2020)
33. Lai, S., et al.: Effect of non-pharmaceutical interventions to contain COVID-19 in China. Nature **585**, 410–413 (2020)
34. Cheng, V., et al.: The role of community-wide wearing of face mask for control of coronavirus disease 2019 (COVID-19) epidemic due to SARS-CoV-2. J. Infect. **81**, 107–114 (2020)
35. Souza Santos, A., et al.: Dataset on SARS-CoV-2 non-pharmaceutical interventions in Brazilian municipalities. Sci. Data **8**, 1–6 (2021)
36. Ivanov JSON to Tree Diagram Converter (2014). https://github.com/ivan111/vtree. Accessed 10 Nov 2022
37. Dask Development Team Dask: Library for dynamic task scheduling (2016). https://dask.org
38. Mistry, A.: Dynamic contagion spread modelling over emergent spatio-temporal contact networks. Masters thesis (2022)
39. Groot, R., et al.: Middle east respiratory syndrome coronavirus (MERS-CoV): announcement of the coronavirus study group. J. Virol. **87**, 7790–7792 (2013)
40. Great Britain. Magna Carta, 1215 June 15 [manuscript] (1215)
41. Kourtis, A., Appelgren, K., Chevalier, M., McElroy, A.: Ebola virus disease: focus on children. Pediatr. Infect. Dis J. **34**, 893–897 (2015)
42. Noorbakhsh, F., et al.: Zika virus infection, basic and clinical aspects: a review article. Iran. J. Public Health **48**, 20–31 (2019)
43. Marani, M., Katul, G., Pan, W., Anthony J.: Parolari intensity and frequency of extreme novel epidemics. Proc. Natl. Acad. Sci. **118**, e2105482118 (2021). https://www.pnas.org/doi/abs/10.1073/pnas.2105482118
44. Glatter, K., Finkelman, P.: History of the Plague: an ancient pandemic for the age of COVID-19. Am. J. Med. **134**, 176–181 (2020)
45. Rambaut, A., et al.: A dynamic nomenclature proposal for SARS-CoV-2 lineages to assist genomic epidemiology. Nat. Microbiol. **5**, 1403–1407 (2020)
46. Miller, E., Roorda, M.: Prototype model of household activity-travel scheduling. Transp. Res. Rec. **1831**, 114–121 (2003)
47. Roorda, M., Miller, E., Habib, K.: Validation of TASHA: a 24-h activity scheduling microsimulation model. Transp. Res. Part A: Policy Pract. **42**, 360–375 (2008)
48. Keeling, M., Eames, K.: Networks and epidemic models. J. R. Soc. Interface **2**, 295–307 (2005)

49. Challenger, J., et al.: Modelling upper respiratory viral load dynamics of SARS-CoV-2. BMC Med. **20**, 1–20 (2022)
50. Bentley, J., Friedman, J.: Data structures for range searching. ACM Comput. Surv. (CSUR) **11**, 397–409 (1979)
51. Fredriksson, K.: Geometric near-neighbor access tree (GNAT) revisited. ArXiv Preprint ArXiv:1605.05944 (2016)
52. Navarro, G., Uribe-Paredes, R.: Fully dynamic metric access methods based on hyperplane partitioning. Inf. Syst. **36**, 734–747 (2011)
53. An, T., Kim, H., Joo, C.: Prediction of COVID-19 infection spread through agent-based simulation. In: Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, pp. 247–252 (2022). https://doi.org/10.1145/3492866.3557735
54. Lee, C., Tenneti, S., Eun, D.: Transient dynamics of epidemic spreading and its mitigation on large networks. In: Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 191–200 (2019). https://doi.org/10.1145/3323679.3326517
55. WHO COVID-19 Dashboard. Geneva: World Health Organization (2020). https://covid19.who.int/. Accessed 30 Sept 2023
56. The Black Arcs Inc. Citisketch (2018). https://blackarcs.org/. Accessed 18 Oct 2022
57. Government of Canada. Statistics Canada: Canada's national statistical agency (2022). https://www.statcan.gc.ca/en/start. Accessed 18 Oct 2022
58. OpenStreetMap contributors. Planet dump (2017). https://planet.osm.org

IoT



FaultBit: Generic and Efficient Wireless Fault Detection Using the Internet of Things

Koustabh Dolui(✉), Ashok Samraj Thangarajan, Sergii Morshchavka, Zhaoyi Liu, Sam Michiels, and Danny Hughes

imec-Distrinet, Department of Computer Science, KU Leuven, Leuven, Belgium
{koustabh.dolui,ashok.thangarajan,sergii.morshchavka,zhaoyi.liu, sam.michiels,danny.hughes}@kuleuven.be

Abstract. The timely monitoring and maintenance of industrial machines is critical to prevent expensive disruptions and down-time. The Internet of Things (IoT) offers a solution to instrument production processes at lower cost and complexity. However, wireless IoT networks are bandwidth constrained, which precludes the transmission of high frequency signals such as vibration or electrical current. Prior approaches to tackling this problem require a high degree of application-specific re-engineering. In this paper, we argue for a new approach to fault detection that is accurate, efficient and applicable to a large class of fault detection problems. We propose FaultBit, an application independent toolkit for fault classification that can gather, compress and classify data using IoT networks. We evaluate FaultBit in two representative scenarios using current and vibration data. In both cases, FaultBit offers classification accuracy 99%, which is very close to application-specific classifiers, while requiring 512× less bandwidth, enabling a battery life of several years.

Keywords: Internet of Things · Wireless Sensor Networks · Machine Learning · Signal Processing · Energy awareness

1 Introduction

The real-time monitoring of industrial equipment prevents expensive machine failures and downtime by enabling *predictive maintenance*, which significantly reduces operational costs by more effectively targeting maintenance effort. For example, the failure of rolling bearings accounts for 45–55% of rotating machinery failures [24]. The emergence of low-cost IoT devices holds great promise in the domain of predictive maintenance, as it enables the wireless monitoring of large numbers of machines without the cost and complexity of routing dedicated

This research is partially funded by the Research Fund KU Leuven (ReSOS, C3/20/014).

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 97–118, 2024.
https://doi.org/10.1007/978-3-031-63989-0_5

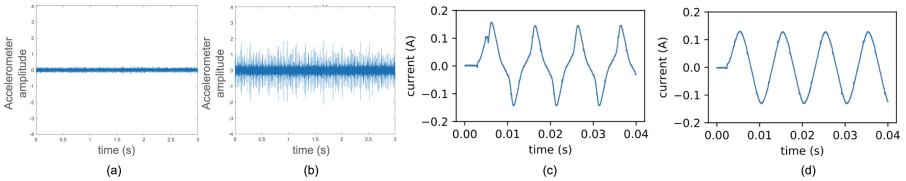


Fig. 1. Illustrative examples of raw fault detection signals taken from our use-cases. (a) shows accelerometer amplitude from a healthy bearing, while (b) shows accelerometer amplitude from a faulty bearing. (c) Shows electrical current signal passing through a healthy solenoid valve, while (d) shows electrical current signal passing through a faulty solenoid valve.

power and network cables [4,37]. Furthermore, wireless networking enables sensors to be embedded in difficult environments, such as dense machinery and on rotating equipment.

While the vision of IoT-enabled predictive maintenance is attractive, significant problems remain. Specifically, in order to deliver multi-year battery life, IoT devices use specialized low power networks such as BLE [5], 6TiSCH [12] or LoRaWAN [3], which offer throughput ranging from a few bps to a few kbps; far too slow to support real-time telemetry from audio, vibration or current sensors. While higher speed networks exist; the fundamental trade-off between bandwidth and energy precludes the combination of long life *and* high speed networking. Furthermore, the hardware design decisions that are required to ensure long battery life also lead to limited memory and processing capacity on the sensor devices themselves, which precludes the local execution of complex models. Neither networking nor computation constraints can be significantly relaxed without negatively impacting battery life and thereby increasing maintenance costs.

The need to reduce bandwidth requirements, while operating within strict memory, processing and power constraints has led to multiple streams of research. On the one hand, generic compression approaches [30,32] enable high frequency sensing by compressing data at its source. However, the bandwidth reductions that can be achieved by these approaches are limited due to their lack of application knowledge. On the other hand, embedded machine learning approaches [11,43] take an orthogonal approach, miniaturising the models themselves so that they can be executed partially, or fully on the IoT device itself. While this family of approaches can achieve dramatic bandwidth and energy savings, the complexity of models is limited and systems developed using this methodology are inherently application specific. This paper takes a different approach, investigating an energy-aware IoT fault detection system that is *generic* in the sense that it can use a wide variety of sensors and provide accurate results across a variety of application domains. FaultBit maximises the efficiency of sensing and compression through continual optimisation, while satisfying application goals such as device lifetime and desired model accuracy. FaultBit achieves that by using an ensemble of standard models that are commonly applied to fault detection and illustrates that faults can be classified efficiently without the need

for use-case specific and customized models. The scientific contributions of this paper are as follows:

1. The first self-adaptive classifier-in-the-loop fault detection system, wherein the classifier is responsible for (re)configuring and optimizing sensors and feature extraction in real-time.
2. An efficient embedded sensing and feature extraction toolkit that offers flexible reconfiguration of the sensor front end and feature extraction models, while operating within the resource constraints of conventional IoT platforms and preserving multi-year battery life.
3. Automated network-wide sensor reconfiguration techniques that empower engineers to easily trade off between fault identification latency and battery lifetime when applying standard fault classifiers.

We use a combination of lab-based benchmarks and open data sets [6] to evaluate FaultBit in two challenging and representative fault diagnosis applications: (i.) using vibration to evaluate bearing wear and (ii.) using electrical current to monitor the state of solenoid valves. In both cases, FaultBit detects faults with an average accuracy that is within 1% of state-of-the-art application-specific techniques, while reducing bandwidth by 512× in comparison to raw data transmission. The accurate classification of normal vs faulty devices facilitates maintenance and earlier interventions in comparison to periodic maintenance.

The remaining sections of the paper are structured as follows. In Sect. 2, we provide background information on fault detection at the edge using IoT devices and derive the requirements for FaultBit. The design of FaultBit is then described in Sect. 3, followed by a reference implementation in Sect. 4. In Sect. 5, we evaluate FaultBit's performance. In Sect. 6, we discuss related work and emphasize the gap that FaultBit fills. Section 7 concludes with discussion and future work.

2 Background

The objective of FaultBit is to detect incipient faults which develop over time in industrial equipment using sensors that are connected by wireless and resource-constrained IoT devices. In particular, we focus on performing fault detection on one dimensional data sharing a common timeline such as single axis acceleration, vibration, distance or electrical current measurements. This data will typically be generated by a large number of wireless sensors that may be distributed across large and complex environments such as factories, trains or warehouses.

2.1 Fault Data Characteristics

Faults in industrial equipment, such as motors and valves, are inevitable due to gradual deterioration caused by corrosion, high loads, and fatigue [40]. Timely monitoring and analysis are essential for predicting fault occurrence and targeting maintenance interventions. For example, predicting Remaining Useful Life (RUL) for solenoid valves can reduce plant shutdowns [25]. However, detecting

incipient faults is challenging because they begin as infrequent events, requiring continuous input data sampling. Constrained IoT devices cannot store raw data for periodic collection or stream it in real-time over low-power wireless networks. We explore these challenges through two representative industrial fault detection applications: vibration monitoring for motors and electrical current monitoring for solenoid valves, which will be used to evaluate FaultBit.

Vibration Monitoring for Motors. Vibration monitoring is widely used for structural health monitoring and fault diagnosis, as accelerometers capture faults earlier than other sensors. Figure 1(a) and 1(b) show contrasting accelerometer amplitude patterns from healthy and faulty bearings. Contemporary MEMS accelerometers sample at 6–26 KHz with a resolution of 1–4 bytes, resulting in a raw data bandwidth requirement of 53–820 kbps, which is orders of magnitude greater than the capacity of low-power IoT networks [19,26].

Solenoid Monitoring. Solenoid valve RUL estimation requires continuous monitoring of their electrical current consumption [25,38], sampled at rates as high as 50 KHz, resulting in data rates of up to 200 kbps per solenoid. This precludes continuous storage or transmission on constrained IoT devices and networks. High-frequency signal monitoring necessitates aggressive compression and filtering, while preserving sufficient data to identify subtle faults, as shown in the electrical current sensor output from a solenoid valve in healthy state and at end-of-life, faulty state in Fig. 1(c) and (d) respectively. Furthermore, in the case of a generic solution, re-configurable data processing algorithms are required to accommodate changing application requirements and environmental conditions. It should be noted that both examples involve cyclo-stationary signals [1], defined as signals having statistical properties that vary periodically with time [15]. A cyclo-stationary signal is illustrated in Fig. 1(c) and Fig. 1(d) showing the pattern of current passing through a solenoid valve over its lifetime. The signal from a faulty bearing in Fig. 1(a) and Fig. 1(b) also represents cyclo-stationarity, however, not as prominent as the solenoid, since the input current signal to the solenoid is cyclo-stationary itself and the solenoid switching is performed uniformly over time. We consider cyclo-stationary signals since this pattern is observable commonly in a wide range of industrial machinery like rotating machinery, gearboxes, piston pumps, bearings and valves [29,35]. As a result, FaultBit can be extended to a wide-range of applications beyond the two examples discussed in this paper.

2.2 IoT Resource Constraints

The limited resources of IoT devices pose challenges in several dimensions.

Sensors. As described in Sect. 2.1, the sample rate and resolution of many fault detection sensors is far beyond that which low-power IoT networks can support. The state of practice in data-driven analytics is to work with raw high

resolution data [42,44]. Likewise, it is extremely difficult to miniaturise fault detection models to the point that they can run directly on the IoT devices themselves. Hence, a generic sensor optimization framework is required together with optimal pre-processing and compression techniques [9]. This enables sensor data workloads to be reduced to the greatest extent possible while preserving full flexibility for fault detection models running in the back-end.

Networking. IoT networks are optimised for low-power operation in order to maximize battery life. This naturally comes at the expense of bandwidth. For example, Low Power Wide Area Network (LPWAN) technologies such as LoRa and Sigfox deliver long range transmissions on a low power budget, but offer a maximum theoretical bandwidth of 290 bps, which is further decreased by a factor of 10–1000× based on regional duty cycle limitations [3]. While higher speed networks are available, the fundamental trade-off between power, range and bandwidth precludes long life *and* high speed networks. This clearly makes the transmission of raw sensor data such as that described in Sect. 2.1 unfeasible. While technologies such as Bluetooth Low Energy (BLE) and 802.15.4 offer somewhat higher bandwidth, this is still insufficient to support any form of raw sensor data transmission. Hence, edge analysis must be performed on the device itself to optimize bandwidth use and transmit only the essential features that are required to train a failure detection model and classify collected data.

Energy. IoT devices may be powered by either batteries or harvested energy from the environment, either of which results in a constrained energy budget [4, 36]. Utilization of network bandwidth as well as local processing resources to perform on-device analysis depletes the battery. However, networking typically consumes the majority of available power. Compression or pre-processing of data on the device reduces transmitted data and thereby power consumption, however it may adversely affect the performance of the fault detection analysis itself. The key role for a generic IoT fault detection middleware such as FaultBit is thus to measure the relationship between data compression and model accuracy and co-optimise the two such that a sensor system achieves its required lifetime while also delivering fault diagnosis of sufficient accuracy.

2.3 Fault Modelling

The data collected for fault detection in various use cases may originate from different sensors with varying data distributions and modeling of such data. Data-driven techniques, such as the use of deep learning models, can solve this issue, however, prior work has typically required raw data to uncover hidden data representations for early fault identification [16,41]. Some prior work on the prediction of RUL for solenoids also utilizes multi-modal sensor data, such as temperature and pressure sensors along with the current measurement sensors, which necessitates either (i) deep learning approaches for data-driven analysis or (ii) considerable domain-specific expert knowledge to build physical models

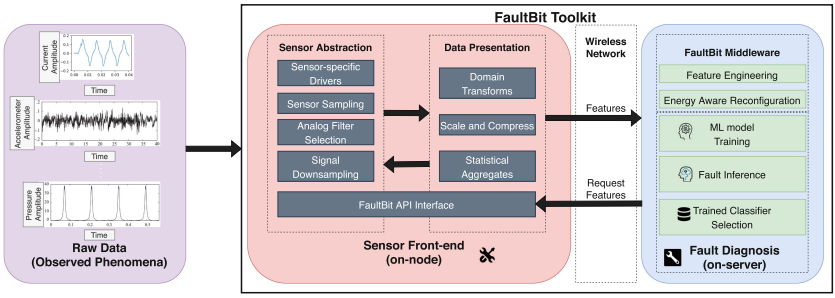


Fig. 2. FaultBit Architecture.

for fault detection. Physical model features also tend to be specific to the fault detection application, limiting the re-usability of these approaches. In contrast to data driven techniques, FaultBit enables deep configuration of sensors followed by generic on-device compression and filtering to minimize required bandwidth. The major challenge here lies in the consistent abstraction of (i.) sensors, (ii.) data compression and (iii.) filtering steps. In order to classify the sensor data as faults, FaultBit applies supervised learning techniques, such as Support Vector Machines (SVM), Random Forest (RF), and neural networks.

2.4 Requirements

The requirements for FaultBit, derived from the above challenges are as follows:

1. Deliver a consistent abstraction of the sensor layer that simplifies the reconfiguration of a variety of sensors.
2. Provide a generic suite of data processing, compression and filtering algorithms that reduce the bandwidth requirements of typical fault detection sensors.
3. Enable intelligent trade-offs between device lifetime, model accuracy and fault diagnosis latency.
4. Deliver self-adaptive software techniques to optimize sensor and edge analysis functionality to meet application requirements and environmental conditions.

3 Design of FaultBit

FaultBit strives to provide a generic deploy-and-forget approach to fault identification using resource-constrained wireless IoT sensors. This demands an end-to-end approach that continuously optimises the configuration of the IoT sensor nodes to minimise energy and bandwidth use, while maintaining a high level of accuracy in fault detection. To balance these contradictory requirements FaultBit adopts a two tier approach connected via a feedback loop. FaultBit includes a common sensor abstraction layer with low-level sensor specific drivers and a common networked API that can adapt key parameters of the driver to meet application-specific fault diagnosis demands.

The FaultBit toolkit operates in two phases (i.) the Training Phase and (ii.) the Fault Detection Phase. In the training phase, FaultBit analyses a range of different features from the acquired signal to identify the minimal yet sufficient *feature vector* that enables diagnosis of faults at sufficient accuracy and latency during the fault detection phase. Throughout both the training and fault detection phases, FaultBit operates in a self-adaptive feedback loop wherein the performance of the models is used to tailor data gathering functionality. The features that are selected for fault identification are identified through an iterative training process using increasing volumes of data. After the model is trained to identify faults, we show that the periodic transmission of very small feature vectors, often fitting within a single BLE or LoRa packet, is sufficient to diagnose a diverse range of faults. Specifically, we evaluate FaultBit on cyclo-stationary signals, defined as signals having statistical properties that vary periodically with time [15], which constitutes majority of signals observed from rotatory machines [1,8]. The training phase can be done with minimal data by imparting expert knowledge, drastically improving the lifetime of the sensor nodes. The block diagram of FaultBit is shown in Fig. 2 and explained in the remainder of this section.

3.1 Sensor Front-End

Contemporary digital sensors offer a wide range of configuration options which, in combination with the computation capabilities of IoT nodes, can be used to efficiently implement data compression and filtering [2,37]. FaultBit exploits this capability with a common device abstraction layer, which is used to configure sensor and data options. The commands are translated to sensor specific parameters and configurations for the algorithm adopted for on-node processing. An example sensor configuration for vibration monitoring may include: a spectrum with a frequency resolution of 1 Hz and a signal range of 0-400 Hz, with a 1-byte feature resolution. This is translated by FaultBit to a sensor sampling rate of 800 Hz and a 512 point FFT. This sensor front-end provides an array of parameters that form the first tunable knob in FaultBit. The configuration of the sensor node and the features extracted is controlled by the FaultBit middleware from the back-end through the aforementioned networked API interface (Table 1).

Sensor APIs are broadly divided into sensor node configuration and data representation. The sensor node processes received API calls as a stream, which allows them to be arbitrarily aggregated in order to improve energy efficiency. The use of a common API abstraction mitigates the need for back-end developers to understand low-level and platform-specific sensing details. Algorithm 1 shows the aggregate command translation of a sensor with a finite number of sampling configuration and frequency domain transform configuration.

3.2 FaultBit Middleware

FaultBit assumes that the signal under analysis is *stationary* (e.g. the frequency of an AC power supply or PWM motor control) or *cyclo-stationary* (e.g.

Table 1. APIs for Configuring FaultBit based Sensor

FaultBit Block	API	Parameter Values
Sensor Node Configs	set_range	r_{min}, r_{max}
	set_dimension	sensor_specific
	set_feature_resolution	data_resolution_in_bytes
	enable_impact_detection	enable/disable, no_of_peaks
	enable_frequency_domain_feature	enable/disable, no_of_peaks
	set_data_resolution	$res_{min}, res_{max}(bits)$
	set_monitor_signal_limit	two_byte_max_monitor_signal_limit
Data Representation	send_enabled_features	-
	config_bandpass_filter	f_L, f_S
	downsampling_coefficient	d
	set_fft_size	n_{fft}
	send_full_data	max_frequency, bin_resolution

accelerometer data from a bearing or electrical current measurements from a solenoid [1]). Based on the observed signal, the FaultBit middleware then optimizes pre-processing on the IoT device using the API described in Sect. 3.1. To accomplish this, during the training phase, a generic set of features are iteratively extracted from the raw data and used to train a family of generic models. The specific features that are extracted are tailored based upon feedback from the middleware during the training phase as described in Sect. 3.3. Once the training phase is complete and an adequate diagnosis accuracy has been achieved, FaultBit identifies a definite feature vector which will then be periodically transmitted by the embedded sensors.

Feature Engineering. In the case of cyclo-stationary signals, faults typically manifest in the frequency domain and consequently, spectrum analysis techniques such as Fast Fourier Transform (FFT) are applied to perform domain transformations. Due to its computational efficiency, FFT is well-suited for fault detection applications on constrained IoT devices. In addition, the FFT facilitates the extraction of a set of statistical features that can be utilized in a variety of applications. FaultBit also provides a configurable a band pass filter to extract signals of the relevant frequency for the application. The sample rate and resolution can often still be too high for IoT networks to support. Hence, the output signal is then down-sampled by a factor d depending on the feature resolution specified by the user.

The spectrograms obtained from the FFT are accumulated and averaged in the frequency domain to provide an output of the required resolution. For example, a 1024 bin FFT may be down-sampled in the frequency domain by averaging adjacent bins to form a 512 bin output, depending upon the required frequency resolution of the application. Peak detection is then applied, wherein peak amplitude and frequency pairs are extracted. Each amplitude-frequency pair serves as

Algorithm 1: Aggregate Query Translation.

Output: S_{src} := sensor sampling rate configuration
Output: B_{rc} := domain transform bin resolution configuration
Function *aggregate_query_translation(signal_range, feature_resolution)* **is**
    `// Finding the sampling rate configuration`
    `// Iterate through the available sampling rates of the sensor`
    for *iter* $\leftarrow 0$ *to length(sampling_rate[]) do*
        if $sampling_rate[iter] \geq signal_range * 2$ **then**
            $S_{src} \leftarrow sampling_rate[iter]$
            break
        end
    end
    `// Finding suitable algorithm configuration for the sensor_sampling_rate_config`
    `// Iterate through the available algorithm resolution of the sensor`
    for *iter* $\leftarrow 0$ *to length(algo_resolution[]) do*
        $current_config_resolution = S_{src}/algo_resolution[iter]$
        if $current_config_resolution \leq feature_resolution$ **then**
            $B_{rc} \leftarrow algo_resolution[iter]$
            break
        end
    end
    return S_{src}, B_{rc};
end

a feature for the fault detection models. The number of features transmitted is determined by the FaultBit middleware based on the requirements of the fault classification model, with priority being given to the highest amplitude features. The above steps are illustrated in Fig. 3. These steps represent the on-device processing that is applied across applications supported by FaultBit.

Energy Awareness. The greatest sources of energy consumption for IoT devices are respectively: wireless communication, computation and sensing. The energy consumption of *sensors* varies significantly from low-power digital sensors to power-hungry analog devices and is proportional to the number of samples collected from the sensor. FaultBit supports both digital and analog sensors and enables energy-efficient data acquisition by sampling the sensor at the optimal rate required for fault detection in the application. *Computation* consumes significant energy, and therefore, FaultBit focuses on simple frequency transforms, filtering and data presentation routines that, when used in combination, can significantly reduce network load, while being feasible to host on embedded IoT devices. All models are trained on a cloud server, implying no computational load for the embedded devices. However, *communication* is inherently a power-hungry activity. FaultBit minimizes communication overhead in two key ways. Firstly,

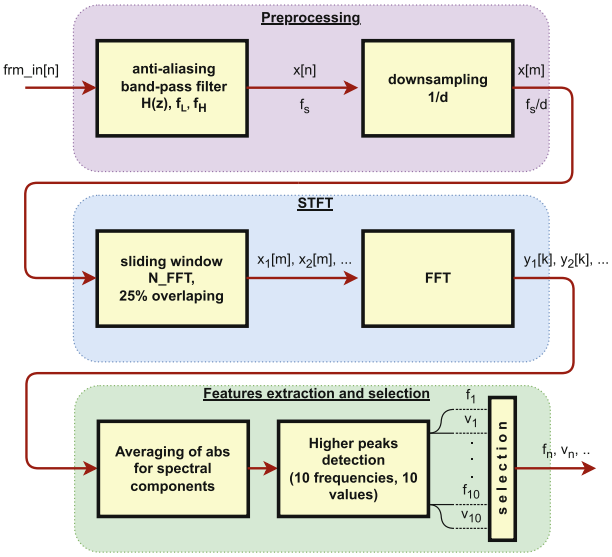


Fig. 3. On-device feature extraction from sensor

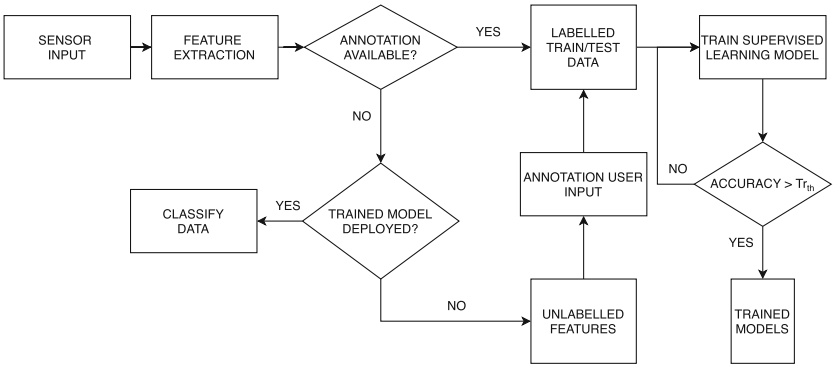


Fig. 4. The life-cycle of a FaultBit application

the data transmitted to the server from the device is compressed, reducing the bandwidth that is required to train the model. Secondly, during the inference stage of fault detection, the rate at which sensor nodes transmit these efficient features is throttled to the minimal rate that is guaranteed to meet the latency requirements of fault detection.

3.3 Fault Diagnosis

The FaultBit toolkit supports the entire life-cycle of model training and inference starting with data collection.

Data Collection and Training. In the data collection stage, the sensor is sampled and the FFT is computed as highlighted in Fig. 3. The entire FFT is transmitted from the device to enable visualization and analysis of the signals required to train various models on the server. The toolkit optimizes data collection at multiple resolutions and sample rates in specified frequency ranges to extract required FFTs for model training. For fault detection, we employ supervised learning, which entails training a model with labelled faulty and non-faulty data. The labelling of data is performed through annotations when periodic monitoring and maintenance of the equipment are performed.

Model Selection. A survey of the state of the art for predictive maintenance [28] reveals that the most popular models used in predictive maintenance include Support Vector Machines(SVM), Decision Tree (DT), Random Forest (RF) Artificial Neural Networks (ANN), Principal Component Analysis (PCA) and k-nearest neighbours (kNN) among others. We select an ensemble of supervised classification models namely Support Vector Machine (SVM), Random Forest (RF) and neural networks to train on the collected data.

Support Vector Machines: SVMs train unsupervised classification and regression models. They use hyperplanes to define class subspaces, and can classify failures as faulty or non-faulty in binary classification. Class overlap reduces accuracy, emphasizing class separability. SVMs are suitable for IoT sensor data, such as from accelerometers and electrical sensors, due to their resilience to noise and ability to perform with minimal training data. Additionally, SVMs use fewer supporting vectors than training instances, facilitating incremental learning [21].

Decision Trees: (DT) methodologies are renowned for their robustness in managing several forms of data, encompassing non-linear, mixed, and categorized data, which is aligned with the requirements of FaultBit. However, it is important to note that DT methods are susceptible to over-fitting and may not always exhibit stability in the presence of noisy data. Random Forest models based on DT approaches exhibits enhanced resistance to noise and a higher likelihood of mitigating over-fitting and in turn yield favorable outcomes in real-world scenarios [7,22]. However, DTs often falter on unbalanced datasets, including bias in data which is the case for CBM datasets due to the fact that the faulty data is typically minimal in real-world scenarios. Nevertheless, we select RFs for their versatility to be applied to various kinds of sensor data and to mitigate over-fitting.

Neural Networks: NNs are beneficial to extract patterns from data and tend to be useful where SVMs or DTs fail to perform or classify faulty data effectively. While SVMs and DTs perform well with limited amounts of data, NNs require larger amounts of training data and hence work efficiently when incrementally trained over a longer period.

Model Training. Once labelled data is available for all classes, the data is split into training and test sets for use with this ensemble of models. The models are

trained starting from only a single feature (i.e. frequency/amplitude pair) on the server with an increasing number of features until (i) the training accuracy meets the application requirements or (ii) the amplitudes of the peak frequencies fall to zero. When the required training accuracy is reached, the set of models satisfying the training accuracy requirement are chosen and their accuracy on the test data is measured. The models satisfying the accuracy requirement on the test data are then sorted by the number of features they use. The model which uses the fewest features while still meeting accuracy requirements is deployed for inference. These steps are shown in Algorithm 2. Each of these approaches can be successfully used with FaultBit, as will be shown in our evaluation.

Inference. The deployed model is used for classification of new incoming data from the device and, as a result, only extracted features are transmitted from the device for inference. FaultBit classifies a device to be faulty based on which maintenance cycles can be performed rather than predicting the time at which the device will fail. During a state of transition from one state to another, data can get mis-classified. Hence, a change of state is considered only when the model consecutively shows a new state for 5 samples of incoming data. The inferred labels cannot be validated in real time. However, when the periodic monitoring of equipment is performed, the ground truth is available and the inference error can be re-calculated. As with sensor configuration, the rate at which sensors send features is configurable and can be tuned to minimise network and energy use while respecting application latency requirements.

Incremental Training. The transmission of the FFT during data collection requires more energy than the transmission of features for inference. Once a model is deployed, data collection is still performed for continuous training of the model. However, during this phase, the full FFTs are transmitted intermittently along side the extracted features. We define the number of full FFT transmissions among the total number of transmissions in a given time window as the *training duty cycle*. In the data collection stage, the duty cycle is set to 1 and is gradually reduced in the inference stage to improve energy efficiency while still retraining the model. However, when the test accuracy of the trained model fails to meet application requirements or the inference error is above threshold, the duty cycle may be increased. In this way, FaultBit maintains the trade-off between device lifetime and model accuracy while also allowing models to evolve to changing environmental conditions. Unlike deep learning models mentioned earlier which classifies faults based on additional input data such as humidity and temperature, FaultBit incrementally learns the pattern of data factoring in deviations resulting from environmental changes. The life-cycle of a FaultBit application is shown in Fig. 4.

Algorithm 2: Deployment Model Selection

Input : Set of starting models M_{start}, train and test accuracy requirements Tr_{th}, Te_{th}, FFT data D, labels D_L, classes C
Output: Chosen model for deployment m_{sel}, Set of trained models $M_{trained}$

Initialize trained model set $M_{trained}$ and deployable model set M_{dep} to *null*
Set $n_{features} := 1$
Function *aggregate_query_translation(signal_range, feature_resolution)* **is**
  while *(*$|D| > 0$ *&* $|D_L| > 0 \forall c \in C$*) &* *(—*$M_{start}| > 0$*)* **do**
    features := $n_{features}$ from D
    if *(any feature is null)* **then**
      break;
    end
    Train models from M_{start} over D
    if *(*Acc_{train} *for* $m_i \geq Tr_{th}$*)* **then**
      $M_{trained} \leftarrow m_i$
      Remove m_i from M_{start}
    end
    if $|M_{trained}| > 0$ **then**
      Calculate Acc_{test} $\forall$ $m_i \in M_{trained}$
      if $Acc_{test} > Te_{th}$ **then**
        $M_{dep} \leftarrow m_i$
      end
    end
    `// Select model with sufficient test accuracy and fewest features`
    if $|M_{dep}| > 0$ **then**
      Sort M_{dep} in ascending order by number of features
      $m_{sel} \leftarrow m_0$ from M_{dep}
    end
    `// No model suffices test accuracy requirement; pick model with best test accuracy`
    else
      Sort M_{train} in descending order of Acc_{test}
      $m_{sel} \leftarrow m_0$ from M_{train}
    end
    $n_{features} += 1$
  end
  return m_{sel};
end

4 Implementation

The FaultBit toolkit is implemented on a representative embedded IoT device; the Nordic nRF52840 MCU [31], which offers a 64 MIPS processor, 512 kB RAM, 2MB flash and built in Bluetooth Low Energy (BLE) radio. For vibration

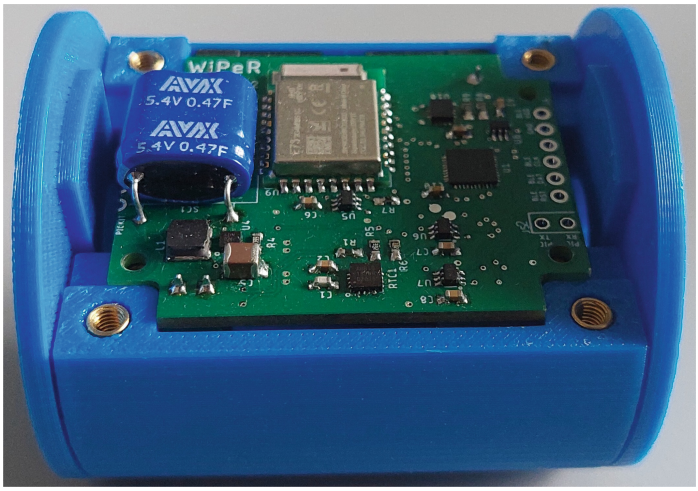


Fig. 5. Device for monitoring bearing faults implementing the FaultBit toolkit

measurements in the bearing monitoring use-case, the accelerometer sensor Kionix KX132-1211-e [19] is used, while the current sensor LEM CTSR 0.3-P is used for the solenoid valve monitoring use-case [20]. The IoT device is powered by a 3 V 620 mAh battery. All energy measurements were performed using a precision desktop multi-meter in a lab environment. The representative node used for deployment in the bearing monitoring application is illustrated in Fig. 5. We employ a class of supervised algorithms to classify the sensor data as faults including Support Vector Machines (SVM), Random Forest (RF) and neural networks. The models are trained using the sklearn Python library while a Flask server is deployed to perform incremental training and inference as discussed in Sect. 3.3.

5 Evaluation

In this section we evaluate the ability of FaultBit to detect and classify faults in two different and representative fault detection tasks; vibration monitoring of motor bearings and current monitoring of solenoid valves.

Vibration Monitoring of Bearing Motors: The vibration monitoring data used in the experiments are acquired from the Case Western Reserve University (CWRU) bearing data set [6]. The data was obtained at a sample rate of 12KHz for three fault conditions (inner race, outer race, and ball fault) and four rotation speeds (1730, 1750, 1772 and 1797 RPM). The bearing motors are first classified to be healthy and faulty. The faulty bearing motors are further classified into 6 classes based on the health of the motor and the root causes of failure namely,

no defects, defects on balls, defects on inner race, defects on outer race: centered, orthogonal and opposite to the load zone.

Solenoid Monitoring: The data from the solenoid monitoring use-case is provided by an industrial partner. The data collection was performed over a solenoid endurance test setup to monitor degradation of 48 closed solenoid valves. The degradation was performed by switching the valves at a rate of 1 Hz with an input voltage of 110 V AC at 50 Hz over a duration of 6 weeks. The solenoid valves undergo three stages in its lifetime, healthy, faulty and End of Life (EOL) which represent the three labels for the data. We assume that data transfer is performed with a standard BLE packet size of 32 bytes for all experiments. The training of the models are performed by randomly dividing the data using the holdout method into 75% training data and 25% validation/testing data. The target test accuracy for both application is set to 99%.

5.1 Accuracy of Fault Analysis

As discussed in Sect. 2.3, FaultBit utilizes generic feature extraction techniques and adapts to the application specific requirements when performing on-device feature extraction. The on-device feature extraction pipeline illustrated in Fig. 3 is used for both types of sensor data.

Vibration Monitoring of Bearing Motors: The accuracy of the trained models are shown in Table 2 for the bearing motors. The columns represent the number of features used to train the model, 'F' representing the peak frequency of the feature while 'A' representing the amplitude at that frequency. It is observed that FaultBit is able to classify the bearing motors as faulty or healthy at 100% accuracy with only 2 features. Furthermore, reusing the same FFT bins, FaultBit classifies the faults to 6 classes by the health or root cause of the fault extracting the required number of features from the FFT. The accuracy for this classification is shown in Table 3. As can be seen from the tables, FaultBit is extremely accurate in classifying faults and their root cause despite requiring the transmission of very little data.

Solenoid Monitoring: FaultBit classifies the data from the solenoid valves with the accuracy shown in Table 4. The accuracy of classification for the solenoid valve data using a state-of-the-art Convolutional Neural Network (CNN) is 99% [38]. FaultBit is able to classify solenoid faults close to the same accuracy with only 20 extracted features. In this case, since the test accuracy threshold is not satisfied, the model with the highest accuracy is chosen and deployed for inference.

5.2 Energy Efficiency

FaultBit functions in two phases, (i) the data collection and training stage and (ii) the inference stage. In the data collection and training phase, for each frame of data, the full FFT is transmitted to the server for training the model. This

Table 2. Accuracy of supervised models trained on bearing motor data to classify if a bearing motor is healthy/faulty with varying number of frequency/amplitude features and $Te_{th} = 99\%$

	1F,1A	1A	1F
Fine Gaussian SVM	**100**	97.2	100
Wide NN	**100**	97.3	97.2
Random Forest (RF)	**100**	96.5	100

Table 3. Accuracy of supervised models trained on bearing motor data to classify the root cause of fault in the bearing motors with varying number of frequency/amplitude features and $Te_{th} = 99\%$

	10F	1F,1A	1A	1F
Fine Gaussian SVM	97.8	96.1	92.6	90.4
Wide NN	99.2	97.1	91.2	82.8
Random Forest (RF)	**99.7**	97.3	91.3	93.9

Table 4. Accuracy of supervised models trained on data collected from solenoid valves to classify valves into three health classes with varying number of frequency/amplitude features and $Te_{th} = 99\%$

	10F,10A	10A	10F	6F,6A	4F,4A	2F,2A	1F,1A	1A	1F
Fine Gaussian SVM	98.1	98.5	91.2	98.1	97.4	97.5	96.5	95.3	73.4
Wide NN	98.4	98.4	91.0	98.2	97.6	97.6	96.6	95.3	73.2
Random Forest (RF)	**98.7**	98.6	92.2	98.5	97.7	97.4	95.0	92.7	75.8

maximises the data available for training at the cost of battery life of the device due to heavy transmission loads. Once a model is trained and is ready for deployment as discussed in Sect. 3.3, the duty cycle of data collection is reduced to only collect data for re-training the model periodically while transmitting only features required for classifying the incoming data.

Figure 6 illustrates the battery life of the device when monitoring for each use-case. We use a nRF52840 processor for this experiment with BLE being the mode of communication between the device and server. The average consumption for the device in the data collection and training phase is 33 µA for solenoid monitoring and 10 µA for bearing motor monitoring. The data collection phase starts with duty cycle = 1 and is reduced when inference phase starts. For example, in the inference stage for a duty cycle of 4%, only 1 full FFT is transmitted for training for 24 set of features transmitted for inference. Hence, as the duty cycle of transmission is reduced the life-time of the device improves drastically in both use-cases as shown in Fig. 6. Reaching the desired level of accuracy in the solenoid monitoring application requires computation of the FFT over more number of windows to the cover the entire frequency spectra resulting in higher

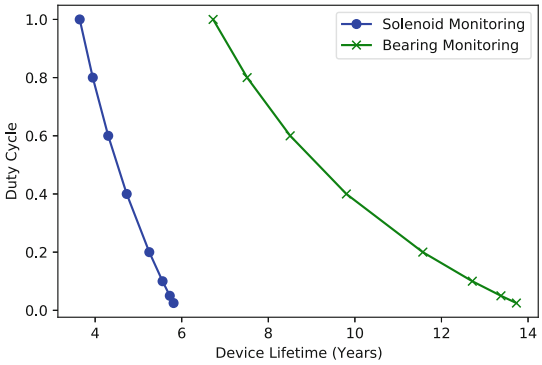


Fig. 6. Device lifetime for fault diagnosis with varying duty cycle of data collection deployed with 620 mAh battery

data transmission in comparison to bearing motor monitoring. This translates to an increased lifetime of the device for the bearing monitoring use-case.

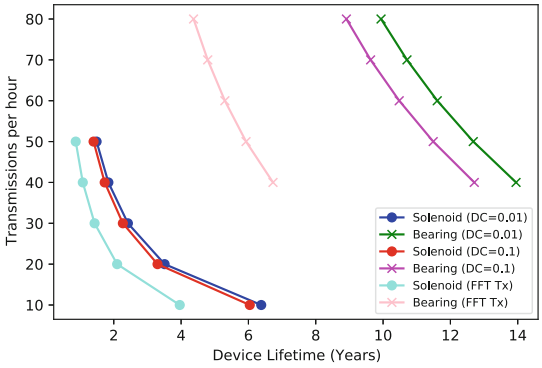


Fig. 7. Device lifetime for fault diagnosis with varying transmissions per hour for fixed duty cycles deployed with 620 mAh battery

We can also illustrate the difference in lifetime caused by transmission of the data at differing intervals for a fixed duty cycle as shown in Fig. 7. The difference between each plotted line in a particular use-case shows the improved lifetime that can be achieved for the same number of transmissions per hour and differing duty cycles. The lifetime significantly drops when we transmit only full FFTs as performed in the data collection and training stage throughout the device lifetime. The transmission of raw data only causes device lifetime to drop far below 1 year and is not suitable for representation in this plot.

5.3 Network Efficiency

The efficiency of network usage is illustrated by comparing the transmission of only the features required to train models and classify data against the raw data transmission as used to train deep learning models over IoT networks.

Vibration Monitoring of Bearing Motors: The accelerometer sensor sampled at 1024 Hz and a 512 byte FFT produces an output of 5 FFT frames per sampled signal. The entire FFT for each sample amounts to 1280 bytes, while the extracted features amounts to 9 bytes of transmission for 2 features. In the data collection phase we see a reduction of 68.75% in network usage while a reduction of 99.8% for the inference phase.

Solenoid Monitoring: The current sensor down-sampled to 5120 Hz and a 512 byte FFT produces an output of 37 FFT frames per sampled signal. The entire FFT for each sample amounts to 9472 bytes of transmission while the extracted features amount to 40 bytes of transmission for 10 features. In the data collection phase we see a reduction of 53.75% in network usage while a further reduction of 99.9% for the inference phase. These readings show that FaultBit is extremely efficient for the network under test compared to sending raw data. During the inference phase, this would enable orders of magnitude more devices to be supported before network bandwidth is exhausted. At the current time, we do not know what re-training duty cycle will be required in real-world deployments. Investigating this is the subject of our future work.

5.4 Discussion

In contrast to traditional methods, which involve the selection of a subset of features from a given pool, FaultBit compresses the data into a collection of features that possess both minimalistic and universal features, enabling their utilization by multiple models. This allows FaultBit to select the most suitable model from a collection of models, while simultaneously minimizing the use of features. FaultBit has been evaluated on two distinct datasets: one derived from an electrical current signal and the other derived from an accelerometer signal. The results show that FaultBit achieves high accuracy comparable to deep learning models, while transmitting only a fraction of data from the IoT device. Overall, FaultBit is a promising approach for fault classification on IoT devices. It is particularly well-suited for low-power and low-bandwidth networks, where it can achieve high accuracy while minimizing data transmission.

6 Related Work

6.1 Machine Learning-Based Industrial IoT

Machine learning techniques have been widely used for industrial IoT applications [23]. Data-driven approaches in fault detection bypass the need for feature extraction from the raw sensor data. The sensor data itself is provided as an

input to a deep neural network where the features are extracted and the signal is classified. The authors of [27] propose a two-layer artificial neural network to classify faults in bearings. Deep neural networks such as [18] utilize three to five layers for fault diagnosis in roller bearings with improved accuracy. Authors of [10] proposed deep neural network-based approaches for segmentation and classification tasks in IoT sensor streams with no or limited labels. In [34], the authors presented a model incorporating ML for low-power transmission among IoT devices using LoRa. Specifically, they introduced a Channel State Information (CSI) learning mechanism in LoRa edge devices, which are then used to control the entire network's power. In [33], Soother et al. used a soft real-time fault diagnosis system for edge equipment based on domain adaptive training strategy using long- and short-term memory networks for remote and real-time detection of bearing faults from vibration data to reduce faults. Many of these approaches are complementary to FaultBit, in the sense that they could be integrated into the collection of available back-end models. However, none of these approaches can be directly applied to the problem of industrial fault detection using resource-constrained IoT devices.

6.2 Fault Detection in IoT Networks

Data compression techniques collect raw sensor data and perform feature extraction on-device to reduce network load. Thangarajan et al. [37] proposed such an approach to compress accelerometer sensor data to perform bearing fault diagnosis while improving battery life and reducing latency. The authors of [39] perform data reduction on a magnetic sensor and an accelerometer on-device for bearing fault diagnosis in a permanent magnet synchronous motor. This results in a 95% reduction in transmitted data and hence optimizes energy usage of the device. IoT devices are resourceful enough for feature extraction on the device [4] along with fault identification on-device itself [17]. This results in a massive reduction of energy expenditure due to data transmission. However, the data resolution is also affected and traditional fault detection algorithms cannot be enabled using this paradigm [14]. The devices are tailored to an application specific deployment which makes it quite static over its lifetime [13,36]. FaultBit addresses this problem supporting multiple applications at run-time through continuous data collection and model training.

7 Conclusion and Future Work

This paper proposed FaultBit, an application independent toolkit for fault detection using IoT devices and machine learning. Faultbit enables deep reconfiguration of the IoT system to enable efficient data collection and compression followed by classification over bandwidth limited IoT networks. Evaluation illustrates that FaultBit performs well on two diverse use-cases, achieving accuracy close to raw data, while offering multi-year battery life and significant network bandwidth savings.

In the future, we will extend our evaluation of FaultBit to support more sensors by conducting empirical research in other fault types and datasets with uni-dimensional data sharing a common timeline. FaultBit requires labelling to be performed via a human-in-the-loop approach. In future work, we will explore (i) methods to weakly label the data using unsupervised learning approaches, (ii) automation of model switching based on model performance while preserving the energy consumption and limited bandwidth usage of the current approach and (iii) methods to output a time-window in which the fault is predicted to occur, i.e. predict the RUL of devices. Since FaultBit models are incrementally trained, they are susceptible to over-fitting. We currently address this issue by selecting models which are resistant to over-fitting. In future work, we will consider using more complex model structures or introducing regularization methods and other techniques address the problem of over-fitting.

References

1. Rolling element bearing diagnostics—a tutorial. Mech. Syst. Signal Process. **25**(2), 485–520 (2011)
2. LSM6DS3: iNEMO inertial module always-on 3D accelerometer and 3D gyroscope (2016). ST Microelectronics. https://www.st.com/resource/en/datasheet/lsm6dsl.pdf. Accessed 20 Oct 2019
3. Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J., Watteyne, T.: Understanding the limits of LoRaWAN. IEEE Commun. Mag. **55**(9), 34–40 (2017)
4. Afanasov, M., et al.: Battery-less zero-maintenance embedded sensing at the Mithræum of circus maximus. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems, pp. 368–381 (2020)
5. Baert, M., Rossey, J., Shahid, A., Hoebeke, J.: The Bluetooth mesh standard: an overview and experimental evaluation. Sensors **18**(8), 2409 (2018)
6. CWRU Bearing Data Center: CWRU vibration dataset with faulty bearings. https://engineering.case.edu/bearingdatacenter/download-data-file
7. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
8. Chesnes, J., Kolodziej, J.: An application based comparison of statistical versus deep learning approaches to reciprocating compressor valve condition monitoring. In: Annual Conference of the PHM Society, vol. 13 (2021)
9. Chi, Y., Dong, Y., Wang, J., Yu, F.R., Leung, V.C.: Knowledge-based fault diagnosis in industrial Internet of Things: a survey. IEEE Internet Things J. (2022)
10. Chowdhury, T.: Towards reducing labeling efforts in IoT-based machine learning systems: Ph.d. forum abstract. In: Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021), pp. 416–417 (2021)
11. Dolui, K., Thangarajan, A.S., Claes, T., Michiels, S., Hughes, D.: Towards on-board learning for harvested energy prediction. In: Proceedings of the 6th International Workshop on Embedded and Mobile Deep Learning, pp. 7–12 (2022)
12. Dujovne, D., Watteyne, T., Vilajosana, X., Thubert, P.: 6TiSCH: deterministic IP-enabled industrial Internet (of Things). IEEE Commun. Mag. **52**(12), 36–41 (2014)

13. Dutta, P., Grimmer, M., Arora, A., Bibyk, S., Culler, D.: Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In: Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, pp. 497–502. IEEE (2005)
14. Entezami, M., Roberts, C., Weston, P., Stewart, E., Amini, A., Papaelias, M.: Perspectives on railway axle bearing condition monitoring. Proc. Inst. Mech. Eng. Part F: J. Rail Rapid Transit **234**(1), 17–31 (2020)
15. Gardner, W.A., Napolitano, A., Paura, L.: Cyclostationarity: half a century of research. Signal Process. **86**(4), 639–697 (2006)
16. Hoang, D.T., Kang, H.J.: A survey on deep learning based bearing fault diagnosis. Neurocomputing **335**, 327–335 (2019)
17. Hou, L., Bergmann, N.W.: Novel industrial wireless sensor networks for machine condition monitoring and fault diagnosis. IEEE Trans. Instrum. Meas. **61**(10), 2787–2798 (2012)
18. Jia, F., Lei, Y., Lin, J., Zhou, X., Lu, N.: Deep neural networks: a promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. Mech. Syst. Signal Process. **72**, 303–315 (2016)
19. Kionix: KX132-1211 tri-axis digital accelerometer specifications. https://kionixfs.azureedge.net/en/datasheet/kx132-1211-e.pdf
20. LEM: Current Transducer CTSR series. https://www.lem.com/sites/default/files/products_datasheets/ctsr_0_3-p_ctsr_0_6-p_v8.pdf
21. Li, H.X., Yang, J.L., Zhang, G., Fan, B.: Probabilistic support vector machines for classification of noise affected data. Inf. Sci. **221**, 60–71 (2013)
22. Li, J., Liu, L., Liu, J., Green, R.: Building diversified multiple trees for classification in high dimensional noisy biomedical data. Health Inf. Sci. Syst. **5**, 1–10 (2017)
23. Li, L., Ota, K., Dong, M., Borjigin, W.: Eyes in the dark: distributed scene understanding for disaster management. IEEE Trans. Parallel Distrib. Syst. **28**(12), 3458–3471 (2017)
24. Mao, W., Liu, Y., Ding, L., Safian, A., Liang, X.: A new structured domain adversarial neural network for transfer fault diagnosis of rolling bearings under different working conditions. IEEE Trans. Instrum. Meas. **70**, 1–13 (2021). https://doi.org/10.1109/TIM.2020.3038596
25. Mazaev, G., Ompusunggu, A.P., Tod, G., Crevecoeur, G., Van Hoecke, S.: Data-driven prognostics of alternating current solenoid valves. In: 2020 Prognostics and Health Management Conference (PHM-Besançon), pp. 109–115. IEEE (2020)
26. ST Microelectronics: LSM6DSOX datasheet. https://www.st.com/resource/en/datasheet/lsm6dsox.pdf
27. Prieto, M.D., Cirrincione, G., Espinosa, A.G., Ortega, J.A., Henao, H.: Bearing fault detection by a novel condition-monitoring scheme based on statistical-time features and neural networks. IEEE Trans. Ind. Electron. **60**(8), 3398–3407 (2012)
28. Ran, Y., Zhou, X., Lin, P., Wen, Y., Deng, R.: A survey of predictive maintenance: systems, purposes and approaches. arXiv preprint arXiv:1912.07383 (2019)
29. Roshanmanesh, S., Hayati, F., Papaelias, M.: Utilisation of ensemble empirical mode decomposition in conjunction with cyclostationary technique for wind turbine gearbox fault detection. Appl. Sci. **10**(9), 3334 (2020)
30. Sadler, C.M., Martonosi, M.: Data compression algorithms for energy-constrained devices in delay tolerant networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys 2006, pp. 265–278. Association for Computing Machinery (2006)
31. N Semiconductor: NRF52840 product specification v1.1. https://infocenter.nordicsemi.com/

32. Shao, C., Nirjon, S.: ImageBeacon: broadcasting color images over connectionless Bluetooth LE packets. In: Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, IoTDI 2017, pp. 121–132. Association for Computing Machinery (2017)
33. Soother, D.K., Ujjan, S.M., Dev, K., Khowaja, S.A., Bhatti, N.A., Hussain, T.: Towards soft real-time fault diagnosis for edge devices in industrial IoT using deep domain adaptation training strategy. J. Parallel Distrib. Comput. **160**, 90–99 (2022)
34. Suresh, V.M., Sidhu, R., Karkare, P., Patil, A., Lei, Z., Basu, A.: Powering the IoT through embedded machine learning and LoRa. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 349–354. IEEE (2018)
35. Tang, S., Yuan, S., Zhu, Y.: Cyclostationary analysis towards fault diagnosis of rotating machinery. Processes **8**(10), 1217 (2020)
36. Thangarajan, A.S., Yang, F., Joosen, W., Hughes, D.: Deterministic 40 year battery lifetime through a hybrid perpetual sensing platform (hyper). In: Proceedings of the 10th International Conference on the Internet of Things, pp. 1–8 (2020)
37. Thangarajan, A.S., Yang, F., Joosen, W., Michiels, S., Hughes, D.: ReFrAEN: a reconfigurable vibration analysis framework for constrained sensor nodes. In: 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 124–131 (2021). https://doi.org/10.1109/DCOSS52077.2021.00033
38. Tod, G., Mazaev, G., Eryilmaz, K., Ompusunggu, A.P., Hostens, E., Hoecke, S.V.: A convolutional neural network aided physical model improvement for AC solenoid valves diagnosis. In: 2019 Prognostics and System Health Management Conference (PHM-Paris), pp. 223–227 (2019). https://doi.org/10.1109/PHM-Paris.2019.00044
39. Wang, X., Lu, S., Huang, W., Wang, Q., Zhang, S., Xia, M.: Efficient data reduction at the edge of industrial internet of things for PMSM bearing fault diagnosis. IEEE Trans. Instrum. Meas. **70**, 1–12 (2021)
40. Zhang, B., Sconyers, C., Orchard, M., Patrick, R., Vachtsevanos, G.: Fault progression modeling: an application to bearing diagnosis and prognosis. In: Proceedings of the 2010 American Control Conference, pp. 6993–6998. IEEE (2010)
41. Zhang, S., Zhang, S., Wang, B., Habetler, T.G.: Deep learning algorithms for bearing fault diagnostics-a comprehensive review. IEEE Access **8**, 29857–29881 (2020)
42. Zhang, W., Peng, G., Li, C., Chen, Y., Zhang, Z.: A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. Sensors **17**(2) (2017). https://doi.org/10.3390/s17020425. https://www.mdpi.com/1424-8220/17/2/425
43. Zhao, Y., et al.: Towards battery-free machine learning and inference in underwater environments. In: Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications, pp. 29–34 (2022)
44. Zheng, H., Yang, Y., Yin, J., Li, Y., Wang, R., Xu, M.: Deep domain generalization combining a priori diagnosis knowledge toward cross-domain fault diagnosis of rolling bearing. IEEE Trans. Instrum. Meas. **70**, 1–11 (2021). https://doi.org/10.1109/TIM.2020.3016068



DeepHeteroIoT: Deep Local and Global Learning over Heterogeneous IoT Sensor Data

Muhammad Sakib Khan Inan[1], Kewen Liao[1(✉)], Haifeng Shen[1], Prem Prakash Jayaraman[2], Dimitrios Georgakopoulos[2], and Ming Jian Tang[3]

[1] Australian Catholic University, North Sydney, NSW, Australia
muhammadsakibkhan.inan@myacu.edu.au, {kewen.liao,haifeng.shen}@acu.edu.au
[2] Swinburne University of Technology, Hawthorn, VIC, Australia
{pjayaraman,dgeorgakopoulos}@swin.edu.au
[3] Atlassian, Sydney, NSW, Australia

Abstract. Internet of Things (IoT) sensor data or readings evince variations in timestamp range, sampling frequency, geographical location, unit of measurement, etc. Such presented sequence data heterogeneity makes it difficult for traditional time series classification algorithms to perform well. Therefore, addressing the heterogeneity challenge demands learning not only the sub-patterns (local features) but also the overall pattern (global feature). To address the challenge of classifying heterogeneous IoT sensor data (e.g., categorizing sensor data types like temperature and humidity), we propose a novel deep learning model that incorporates both Convolutional Neural Network and Bi-directional Gated Recurrent Unit to learn local and global features respectively, in an end-to-end manner. Through rigorous experimentation on heterogeneous IoT sensor datasets, we validate the effectiveness of our proposed model, which outperforms recent state-of-the-art classification methods as well as several machine learning and deep learning baselines. In particular, the model achieves an average absolute improvement of 3.37% in Accuracy and 2.85% in F1-Score across datasets.

Keywords: IoT Sensor Data · Data Heterogeneity · Deep Learning · Time Series Classification

1 Introduction

IoT sensor data[1] or readings exhibit time series-like sequences. The classification of such sequence data has emerged as an in-demand research area, given the ubiquitous presence of IoT sensors in our daily lives in accordance with the growing dominance of mobile computing [10,20,35]. The next decade is predicted

[1] Aligning with the works of [5,21,22], we define IoT sensor data as sequences of raw numerical readings/observations obtained from IoT sensors over time.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 119–135, 2024.
https://doi.org/10.1007/978-3-031-63989-0_6

to see IoT devices creating $14.4 trillion worth of value across different industries [3]. IoT sensor devices are widely adopted in the sectors like healthcare, environmental monitoring, traffic management, and energy management, in a diverse range of settings [5,21,22]. This naturally causes the IoT data to vary in timestamp range, observation configurations, sampling frequency, unit of measurement, and geo-location [30], which leads to the production of a plethora of IoT sensor data with inherent heterogeneity [2,9,21]. The inherent heterogeneity, where sensors of the same type (e.g., temperature) record data in different units of measurement (Celsius or Fahrenheit) or with varying sampling frequencies, results in inconsistent and non-repetitive complex discriminatory patterns in time-series readings from IoT sensors. In such scenarios, time-series sensor readings/sequences display a mixture of local sub-patterns occurring at irregular intervals, forming a global pattern of heterogeneous nature at times.

The growing use of IoT sensors, integrated with a variety of mobile computing devices, highlights the importance of accurate classification, such as distinguishing between temperature and humidity data, to ensure the reliability of dynamic systems like smart homes [7]. The heterogeneous nature of IoT sensor data also affects the data quality [10], making the classification tasks challenging even for the existing machine-learning techniques. It also hinders the effective re-purposing of these data for interoperability [20]. For instance, the traditional time series classification methods often underperform [1,22] when it comes to classifying IoT sensor data in a multi-class setting [1,2].

Recent research has witnessed the development of machine learning (ML) algorithms that leverage textual metadata (e.g., sensor names and descriptions) to ameliorate the classification of IoT sensor data [19,21,22]. However, such textual metadata are often not available or presented inaccurately in IoT sensor datasets [2]. The expansion of IoT systems also increases the chance of significant metadata loss due to disrupted or corrupted connections to configuration metadata, making collected IoT sensor data unusable [5]. Moreover, depending on the availability of accurate metadata limits the capability and feasibility of learning where only numeric sensor readings are available [2,4]. Though optimized ensemble machine-learning algorithms [21,22] and feature transformation-based strategies [2] recently achieved notable successes in heterogeneous IoT sensor data classification, those approaches lack generalizability and scalability and they are also not deeply learned end-to-end processes [14]. Along with the rapid advancement of artificial intelligence, deep learning (DL) algorithms have demonstrated massive potential in time series classification [14] due to their advantage of representation learning. Some studies [6,26] developed DL-based solutions for IoT sensor data classification but only considered a homogeneous data domain. There also have been studies that incorporated deep learning-based strategies for IoT sensor data forecasting [24,28]. However, there were no DL models specifically designed for heterogeneous IoT sensor data classification. In search of such a DL solution, we first rigorously investigated the state-of-the-art deep time series classification algorithms including InceptionTime [15] and TapNet [34], along with other traditional machine learning and deep learning based algorithms. From our empirical study, we found that in terms of the heterogeneous

IoT sensor datasets, existing solutions do not perform well which was similarly highlighted in previous studies [1,2]. Though there exist effective ensemble learning models like MACE [21] and TKSE [22], these models are highly complex, easily overfit, and do not learn the underlying artifacts of heterogeneous IoT sensors with a contextual-learned approach.

To this end, we propose a novel end-to-end deep learning model with the aim of better capturing the heterogeneous patterns of IoT sensor data that would lead to enhanced classification performance. The model learns ensembled *local* neighborhood-based features with Convolutional Neural Network (CNN) convolution layers of varying kernel sizes, and *global* whole series sequential features with a stack of Bi-directional Gated Recurrent Units (GRUs). We deliberately name the model as DeepHeteroIoT for ease of reference. The main contributions of the work are summarized as follows:

- We propose the first end-to-end DL model for heterogeneous IoT sensor data classification. The model facilitates a novel combination of CNN and Bi-directional GRU modules to respectively learn local and global patterns of heterogeneous IoT sensor data.
- We provide rigorous empirical comparisons across multiple IoT datasets along with the development of a new heterogeneous IoT sensor dataset. Experimental results demonstrate that our deep learning model achieves state-of-the-art classification performance in both accuracy and F1-score.

2 Related Works

IoT Sensor Data Classification. IoT sensors typically generate numerical continuous data, forming numerical sequences when collected at regular or irregular intervals over a specific time period. The task of classifying these continuous numerical series is akin to problems such as Time Series Classification (TSC) or Sequence Classification. To deal with challenging heterogeneous IoT sensor data, Montori et al. [22] proposed a two-layer sequential ensemble approach of classifiers where individually trained classifiers are stacked during the prediction pipeline in a sequential manner to filter out classes with fewer prediction probabilities. In their extended version [21], an improved multi-layer sequential ensemble model was proposed with novel heuristics for selecting and ordering classifiers and filtering classes between classifiers. Borges et al. [2] proposed a transformation-based classification strategy that converts raw sensor data to an ordinal pattern with improved feature representation and class separability. Postol et al. [25] proposed a topological data analysis-based strategy for noisy IoT sensor data classification. A random forest-based strategy was recently proposed to classify a signal type being read from an IoT sensor utilizing raw IoT sequence data [5]. Moreover, a probabilistic data mining approach [4] incorporating slope distribution computation via linear approximation of time series sequence was first developed for heterogeneous IoT sensor data classification. Although previous approaches achieved notable classification performance, they have only sought global patterns from different perspectives. Several studies have

explored the potential of deep learning for classifying IoT sensor data for tasks such as intelligent waste management [26] and human activity recognition [6]. However, the sensors used in these domains are predominantly homogeneous and share similar characteristics. To the best of our knowledge, the development of end-to-end deep learning algorithms to better capture rich low-dimensional temporal semantics of heterogeneous IoT sensor data (that comes from diverse environments) is still missing in the literature.

Time Series Classification. The time series classification domain has seen many developments over the years such as the classical methods of Dynamic Time Warping (DTW) and Weighted Dynamic Time Warping [1] that rely on whole-series distance-based similarity measures. Additionally, methods that work with the approach of subsequence extraction, coined as "shapelets" based strategies, have been popular in data mining [27,33]. However, to deal with the dimensionality issue present in the distance measure-based approaches, a lower-bound symbolic representation of time series sequence named Symbolic Aggregation approXimation (SAX) [17] was introduced to first transform data via Piecewise Aggregate Approximation (PAA) and then symbolize the PAA representation into a discrete string. Lin et al. [18] proposed a histogram-based high-level feature extraction strategy utilizing PAA through SAX representation for time series classification, taking inspiration from the popular "bag of words" strategy in the domain of text mining. Nevertheless, the majority of conventional techniques concentrate on individual time series attributes like shape or frequency. These methods lack scalability and efficacy in handling the diverse characteristics inherent to the IoT data [21]. In recent years, learning-based methods that incorporate state-of-the-art deep learning algorithms have been developed for time series classification due to their capability of learning low-dimensional feature representations automatically without domain knowledge [14]. Convolutional Neural Network (CNN) based model InceptionTime [15] and Attention-mechanism incorporated hybrid approach TapNet [34] are two prominent examples of such effective deep learning-based solutions for time series classification. Additionally, ensemble or hybridization of deep learning algorithms including CNN and recurrent neural networks (RNN) have shown great potential in extracting rich semantics of time series in terms of sensor data classification [13] and forecasting challenges in various domains such as water level prediction [24] and short-term residential load forecasting [28]. While current state-of-the-art time series classification algorithms exhibit strong performance on general time series data, they often lack the specialized design required to effectively capture the inherent heterogeneity within IoT sequence data. As a consequence, there is a critical need to develop novel methodologies that can accommodate the varying characteristics of IoT sequence data and extract meaningful insights from these distinct and heterogeneous sources.

3 DeepHeteroIoT: Our Proposed Deep Learning Model

In this section, we delineate a brief description of the key components of our proposed end-to-end deep learning model. As represented in Fig. 1, our proposed deep learning model, DeepHeteroIoT, extracts learned feature representation of global patterns using a stack of bi-directional GRU and local patterns using an ensemble of decoupled CNN stacks with varying kernel sizes. Learned local and global feature vectors are then combined through a concatenation layer which connects to a multilayer perceptron (MLP) head for final classification.

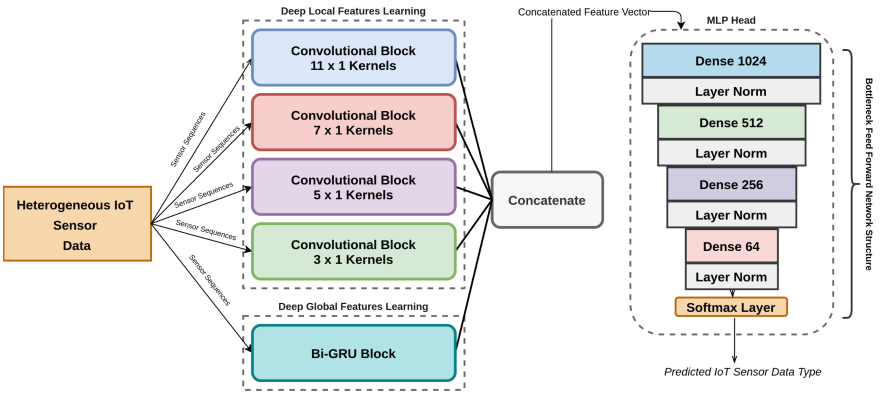


Fig. 1. An overview of our proposed deep learning model with learnable local and global learnable patterns for heterogeneous IoT sequence data classification

Deep Local Features. Initially developed to tackle complex issues within Computer Vision, Convolutional Neural Networks (CNNs) leverage local neighborhood-based features to enhance the distinction between different classes by shaping decision boundaries. As the realm of deep learning continues its rapid evolution, CNN architectures have showcased significant potential in extracting robust spatial features from input sequences. By employing convolutional operations with small kernel sizes, CNNs excel at capturing local intricacies, making them an excellent fit for intricate time series or sequence analyses. In our proposed deep learning model, we introduce a customized novel CNN architecture tailored to seamlessly integrate into our end-to-end framework for IoT sensor data classification. A single convolutional block incorporated in our proposed model can be expressed mathematically;

$$F_f = ConvBlock(X, f, p) \tag{1}$$

In Eq. 1, X represents a vector containing input sequences such as $[X_0, X_1, ... , X_t]$, distributed over t timestamps, f indicates kernel size for convolutional layers, p indicates the type of padding to be incorporated and F_f is the feature

space computed base on kernel size f. Also, each $ConvBlock$ consists of a stack of convolutional layers with designated kernel size and pooling layers. Each set of convolutional blocks comprises nine consecutive 1D-convolutional layers. Following the 3rd and 6th convolutional layers, a max-pooling layer with a size of 2 is incorporated. Each convolutional block ends with a Global Average Pooling layer [31]. The pooling operations allow the models to downsample feature maps in the way the most significant information has been retained and passed to the next layer while reducing the spatial dimensions. Within each convolutional block, the sequence of stacked convolutional layers is set up with 128 filters for the layers preceding the initial max-pooling stage, and subsequently, 64 filters are employed for all successive convolutional layers.

The size of the kernels used in convolutional layers is of utmost importance for capturing local features, as they function as windows to compute features on sub-patterns extracted from the entire time series like sequences at a lower level. In Algorithm 1, local deep learned features are computed at steps 2,3,4 and 5 incorporating X into Eq. 1 for varying kernel sizes with designated padding strategy. Drawing inspiration from the idea of incorporating kernels (convolutional windows) of different sizes to enhance the extraction of local features from different spatial ranges [29], we propose a decoupled ensemble structure for our convolutional module. In our design, we stack convolutional layers with kernel sizes of 3, 5, 7, and 11, allowing our model to learn sub-patterns using various receptive field sizes in separate blocks. These small kernel sizes were adopted on the basis of their proven computational efficiency in prior state-of-the-art CNN models [29], as well as their effectiveness in capturing local features in complex computer vision problems. The final choice of kernel sizes was also determined from randomized parameter tuning. Additionally, padding plays a critical role in the design of CNNs as it ensures the preservation of spatial dimensions in input sequences. By retaining spatial information, padding enables CNNs to process time series alike sequences with varying lengths consistently. For the convolutional layers, we use 'causal' padding. This type of padding strategy adds zeros at the start of the input sequence after each convolution operation. Importantly, this maintains the temporal sequence order, ensuring that $output_t$ does not depend on $input_{t+1}$ to $input_{t+n}$ [23].

Deep Global Features. Gated Recurrent Unit (GRU), one type of Recurrent Neural Network (RNN), has emerged as a powerful deep learning technique for capturing overall long temporal dependencies or global patterns in sequential data for challenges like time series classification [8,32]. Though GRU possesses a simpler architecture with fewer gates and parameters to train, it still excels at capturing both short-term and long-term dependencies in sequential data compared to Long Short-Term Memory (LSTM) [32]. These unique traits make it computationally more efficient and potentially better suited for our IoT Sensor data classification challenge. A GRU unit maintains a gating mechanism incorporating an Update Gate (u_t) and Reset Gate (r_t) based strategy to selectively update and reset its internal state for capturing long-term dependencies

and global facets or features of sequence data along with solving the vanishing gradient problem. Mathematically,

$$u_t = \sigma(W_u \odot x_t + U_z \odot h_{t-1} + b_u) \tag{2}$$

$$r_t = \sigma(W_r \odot x_t + U_r \odot h_{t-1} + b_r) \tag{3}$$

Here, t denotes the timestamp, x_t, and h_{t-1} accordingly denotes the input in the current timestamp and hidden state in the previous timestamp. W, U and b are respective weight matrices and vectors for update and reset gates. To better capture the global sequential patterns of IoT sensor data, we develop a stacked Bi-directional GRU module for integration into our deep learning model. Bi-directional means capturing the patterns of the sequence data from both directions (forward and backward) to allow it to learn temporal dependencies and patterns that exist in both past and future contexts. A single block of bi-directional GRU unit, incorporated into our deep learning model can be represented mathematically as;

$$gru_i = Bidirectional(GRU(X, D)) \tag{4}$$

Here, X represents a vector containing input sequences such as $[X_0, X_1, ..., X_t]$, distributed over t timestamps, and D represents dimensions of output representation space computed by the GRU function. gru_i is the output sequence returned by one $Bidircetional$ function that concatenates the computed output features by the GRU function from both the forward and backward direction in an appending manner. As described in Fig. 1 and Algorithm 1, for deep global sequential feature extraction we stack 3 Bi-directional GRU units accompanied by Batch Normalization after each unit. The hyperparameter setting of parameter D for each unit is accordingly 128, 128, and 64 in steps 7, 8, and 9 for Algorithm 1 with the incorporation of Eq. 4 for computation of global patterns at each epoch for every sample in X.

Combined Deep Learning Model. To improve the classification performance on challenging IoT data, as delineated in Algorithm 1, we propose a novel deep learning model that incorporates deep learning-based local features by our customized CNN and global features by our Bi-directional GRU. As depicted in Algorithm 1, the learned feature space at every epoch from both of these networks is concatenated through a concatenation layer in step 10 (Algorithm 1) and fed into an MLP Head (Multi-layer perception-based fully-connected layer) in step 11 (Algorithm 1) which consists of a stack of dense feed-forward layers accompanied by layer-normalization after each dense linear projection layer. The stack of dense linear projection layers follows a bottleneck design (accordingly 1024, 512, 256 and 64) in terms of the number of neurons in each layer inspired by the previous state-of-the-art architectures [12]. The key motivation for this type of design is to reduce the number of parameters in subsequent layers to force the network to learn more compact representations that eventually improve generalization reducing the chance of overfitting [12]. The number of *Epochs* was set to 200 using ADAM (Adaptive Moment Estimation) optimizer with a learning rate

of 0.001. The weights of the best epoch on the validation stage were considered for the best model.

Algorithm 1. DeepHeteroIoT: Our Proposed Deep Learning Model

Input: $\mathbb{X}$: 2-D matrix that contains N number of individual sequences represented as x_i where IoT Sensor data sequences is distributed of t time-stamps. $\mathbb{X} = [x_i, x_{i+1}, ..., x_{i+N}]$. Each x_i contains a sequence of data points representing IoT sensor readings.. Every x_i in $\mathbb{X}$ feeds as input to the neural network over the epochs into individual steps.

1: **for** Each *epoch* in *Epochs* **do**
2: $F_3 = ConvBlock(X, 3, "causal")$
3: $F_5 = ConvBlock(X, 5, "causal")$
4: $F_7 = ConvBlock(X, 7, "causal")$
5: $F_{11} = ConvBlock(X, 11, "causal")$
6: Concatenate F_3, F_5, F_7, F_{11} in an appending manner which generates a vector $concat_{local}$, containing a matrix of local features.
7: $gru_1 = Bidirectional(GRU(X, 128))$
8: $gru_2 = Bidirectional(GRU(gru_1, 64))$
9: $gru_3 = Bidirectional(GRU(gru_2, 64))$
10: Concatenate extracted local feature ($concat_{local}$) and global features (gru_3) in an appending manner.
11: Feed the concatenated feature space into the MLP Head.
12: Compute softmax predictions.
13: Compute loss of training epoch.
14: Update weights of each layer based on the learning rate.
15: **end for**

Output: $\mathbb{O}$: predicted class labels defining IoT sensor data

4 Experiments

In this section, we outline the outcomes of our model's experiments using benchmark IoT datasets [21,22]. To evaluate the efficacy of our novel deep learning model, we contrasted its Accuracy and F1 scores with conventional machine learning and deep learning models. We also compared it to time series classification-centric deep learning models like InceptionTime [15] and TapNet [34]. Furthermore, we conducted a comparison between our model and a contemporary state-of-the-art ensemble machine learning model known as MACE (as presented in [21]), which was explicitly tailored for classifying IoT data in a prior research endeavor. We omitted comparisons with feature-transformation or data-mining oriented methodologies [2] for classifying IoT sensor data. Because, these techniques rely on statistical transformations of IoT sequence data and our study emphasizes a comprehensive exploration of the capabilities of ML and DL methods in capturing heterogeneous patterns within IoT sequence data at a more granular level [14]. To ensure a consistent comparison within our domain

and alignment with the methodology in the prior study [21], we adopted the identical experimental design, including the division of datasets into training and testing subsets. We performed a stratified split over the datasets assigning 70 % to the training set and 30 % to the testing using the Scikit-Learn library by setting parameter value "random_state=100". The tested classifiers in this paper are implemented in Python 3.9.16 on top of Scikit-Learn for machine learning models (except XGBoost) and Tensorflow for deep learning models. For XGBoost, we utilize the Python implementation of the XGBoost algorithm. All the experiments were conducted on a Linux server running Ubuntu 20.04 LTS as the OS, equipped with a 24-core Intel CPU (x86_64) and 128 GiB of RAM. In our work, we include the Swiss Experiment dataset and Urban Observatory dataset from MACE [21] paper. We exclude the ThingSpeak dataset which was included in the MACE [21] paper as this dataset contains textual meta-data describing IoT sensors which do not go with the motivation of this study of dealing with IoT sensor classification challenge utilizing only raw numeric sensor observations. Additionally, to validate the generalizability of our proposed solution, we constructed another dataset named IOWA ASOS, inspired by a recent study [2] which will be discussed in detail in the remaining sections.

4.1 IoT Sensor Datasets

In this section, we delineate the three benchmark IoT Sensor datasets that have been utilized in this study including the IOWA ASOS dataset that we have developed specifically for this paper to prove the generalizability of our model performance along with Urban Observatory (UrbObs) and Swiss Experiment (Swiss) datasets from previous studies [4,21].

– **IOWA ASOS (IOWA).** Inspired by the study of [2], we developed this IoT sensor dataset by collecting sensor data of various meteorological sensors from the data archive of Iowa Environmental Mesonet (IEM) prepared by the IOWA State University and made available via API and web interface[2] This dataset contains sensor data from meteorological sensors at airports using Automated Surface Observing Systems (ASOS)[3] These sensors can produce observations every minute or every hour based on the requirements of the airport authority to corroborate aviation operations by using weather forecasting reports. To make our own version of the new IoT dataset for this study, we collect data with 1-h intervals over a 6-month time-span from 5 different stations in the IOWA state of the United States. To make this dataset suitable for an IoT Sensor data classification challenge, we include 1-week data at 1-h intervals for each individual classification sample. The dataset contains 8 class labels and these are: Air Temperature, Dew Point Temperature, Relative Humidity, Wind Direction, Pressure Altimeter, Visibility, Wind Gust,

[2] Iowa Environmental Mesonet ASOS-METAR Data Download: https://mesonet.agron.iastate.edu/ASOS/.

[3] Automated Surface Observing Systems from the U.S. National Weather Service: https://www.weather.gov/asos/asostech..

and Apparent Temperature (Heat Index). The missing values are imputed by the standard imputation strategy [2] of replacing by global average (mean) on that time-stamp across the individual class labels.

- **Urban Observatory (UrbObs).** The version of this dataset we utilized for our experimentation was developed in a previous study [21] of IoT sensor data classification. We have incorporated the exact same version of this dataset to make our experimental results comparable to the literature. This dataset contains sensor data generated through an innovative initiative[4] spearheaded by the University of Newcastle. The program aims to establish an urban sensor network in Newcastle, United Kingdom (UK) and provides publicly accessible real-time environmental data [16]. The dataset has 16 different class labels including NO_2 (Nitrogen Dioxide), Wind Direction, Humidity, Wind Speed, Temperature, Pressure, Wind Gust, Rainfall, Soil Moisture, Average Speed, Congestion, Traffic Flow, Journey time, Sound, CO (Carbon Monoxide) and NO (Nitrogen Monoxide). This dataset contains highly correlated IoT data, primarily due to its city-wide scope and originating from a single source [21].
- **Swiss Experiment (Swiss).** This dataset is one of the few extremely noisy and heterogeneous IoT datasets that consist of data from various microscopic locations in various time-stamps within the Swiss Alps Mountain range [4]. This dataset contains data from sensors having a diverse range of sampling rates making the phase-shift of data series very significant [4,21]. Following the exact same version of this dataset from the previous study [21], we cut each time series data to the length of the shortest stream as the original data contains data of slightly different lengths. It includes class labels such as CO_2 (Carbon Dioxide), Humidity, Lysimeter, Moisture, Pressure, Radiation, Snow Height, Temperature, Voltage, Wind Speed, and Wind Direction.

Table 1. Summary of IoT Datasets

Dataset	Length	Duration	Samples	Labels
Urban Observatory	864	1 day	1065	16
Swiss Experiment	445	Variable	346	11
IOWA ASOS	168	1 week	1000	8

Table 1 provides a summary of 3 IoT datasets highlighting time series sequence length, duration of record per sample, the total number of samples, and the total number of class labels for each dataset. Out of these three IoT datasets, the Swiss Experiment dataset exhibits a notable class imbalance and has a very limited number of training samples, making it a particularly challenging dataset for deep learning methods. On the other hand, the IOWA ASOS dataset is almost completely balanced where each class has almost the same amount of data samples. The Urban Observatory exhibits some degree of class

[4] Urban Observatory Environment Program: http://newcastle.urbanobservatory.ac.uk/.

imbalance, but due to the presence of less noisy IoT sensor data, the imbalance issue does not pose a major problem for this dataset [21]. In the Swiss Experiment, the majority class has 78 samples, while the minority class has only 14 samples in total.

4.2 Experimental Results

In this section, a detailed representation of experimental results along with a rigorous comparison against baseline machine learning and deep learning models have been delineated. Also, performance comparison with previous state-of-the-art models across 3 datasets is presented and validated.

Evaluation Metrics. Before starting, we describe the evaluation metrics that have been computed and compared throughout the experimental validation stage for our proposed model and all other experiments. These are:

- *Accuracy:* It indicates the percentage of IoT Sensor data that have been classified correctly by our model out of total data samples.
- *F1-Score:* It is a harmonic mean of Precision and Recall which gives us a balanced perception of our model's performance and helps to validate DL model's effectiveness properly.

For the evaluation of each experiment, we focus on the Accuracy and weighted average F1-Score throughout the study. Later, we computed the macro-average F1 Score for comparison of the model's effectiveness with a previous state-of-the-art study [21]. The weighted average F1 Score is a more suitable way to compute F1-Scores for scenarios where the class imbalance is present to find the dataset. On the other, the macro average F1-Score does not take into account that class imbalance is present in the dataset.

Ablation Studies. To validate the efficacy of our proposed combination of global and local deep learning-based learned patterns for IoT sequence data, we illustrate a detailed ablation analysis of our proposed model in Table 2 with Accuracy scores and weighted average F-1 scores across three different IoT datasets. From the results of Table 2, it is clearly visible that our proposed novel deep-learning model outperforms individual components including only global features, only local features, and only MLP Head (without any deep feature extraction) based classifiers with a high margin. However, out of individual components, classification with only global features dominates compared to combinations with only local features and only MLP Head across 3 datasets.

It is also noticeable that only MLP Head performs badly, indicating the importance of deep-learning-based feature extraction before training a fully-connected network-based classifier for the classification of heterogeneous IoT sensor data. If we draw attention to the accuracy scores presented in Table 2, we can notice that our proposed deep learning model outperforms the second-best combination in Table 2 by 3.75%, 6.73% and 1.68% accordingly on Urban Observatory, Swiss Experiment and IOWA ASOS datasets. Similarly, in terms

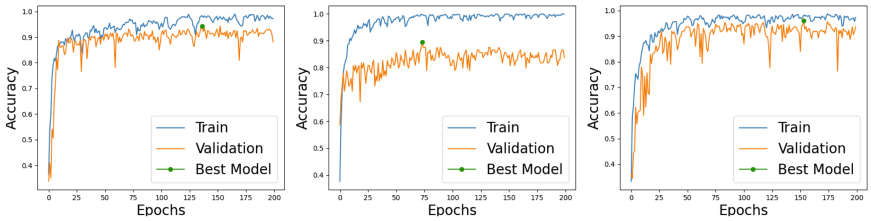
Table 2. Ablation study of our proposed deep learning model across IoT datasets

	UrbObs (Accuracy)	UrbObs (F1-Score)	Swiss (Accuracy)	Swiss (F1-Score)	IOWA (Accuracy)	IOWA (F1-Score)
Global Features	90.63%	89.92%	79.81%	78.5%	94.33%	94.32%
Local Features	88.13%	87.72%	77.88%	76.41%	87.33%	87.50%
MLP Head	77.51%	76.02%	45.19%	42.04%	86.01%	85.80%
DeepHeteroIoT	**94.38%**	**94.27%**	**86.54%**	**85.12%**	**96.01%**	**95.93%**

of weighted average F1-score our proposed model outperforms by 4.35%, 6.62%, and 1.61%. The outperforming scores achieved by our proposed model justify the importance of learning both global and local deep learning-based patterns of IoT sequence data within one combined and end-to-end learning model. As IoT sensor data is heterogeneous, the proposed combination of both local and global learnable deep neural network architecture is a more suitable approach to capture different dynamic facets of IoT sensor data.

Out of the 3 IoT datasets, as described earlier, the Swiss Experiment dataset is highly heterogeneous with a very limited number of available samples including class imbalance issues. In such scenarios, intricate deep-learning architecture like ours tends to suffer from non-convergence issues and poor training generalizability due to data scarcity issues. For this, only the Swiss Experiment dataset was pre-processed in the training stage by incorporating the time series data-augmentation method followed by oversampling by B-SMOTE [11]. Throughout our evaluation process, we also conducted experiments involving the data augmentation module on the other two datasets. However, since these datasets had a sufficient number of samples and exhibited either fewer or no significant class imbalance issues in their original training data, it was obvious that the improvements in accuracy and f1-scores were not as substantial in comparison to the Swiss Experiment dataset after applying data augmentations. And so, for later experimental results only for the Swiss Experiment dataset, we present the results that have been achieved by our model after incorporating the augmented training data. An illustrated representation of the learning curves delineating Accuracy per epoch for the training phase of our proposed deep learning model is outlined in Fig. 2. The learning curves (Fig. 2) certainly validate the training efficacy and robust learning capacity of our proposed model across 3 IoT datasets.

Comparison of Performance Against Other Models. In this stage, we compared our novel DL approach with traditional baseline ML models, including K Nearest Neighbor (KNN), Logistic Regression (LR), Random Forest (RF), Gradient Boosting (GB), Extreme Gradient Boosting (XgBoost), and Support Vector Machine (SVM). To ensure baseline settings for the machine learning algorithms that have been used from Scikit-learn's package, default parameter settings for every algorithm are utilized. For our baseline setting of DL models such baseline RNN and baseline CNN, we accordingly incorporate the Simple



(a) Urban Observatory (b) Swiss Experiment (c) IOWA ASOS

Fig. 2. Learning-curve representation for training and validation (per-epoch) for our proposed model across 3 datasets

RNN API[5] from Tensorflow with a filter size of 32 and Conv1D API[6] by Tensorflow and follow the traditional example[7] of stacking Convolutional units (3 by 1 kernel sizes and "relu" with no padding) followed by a MaxPooling layer. For time series DL models such as InceptionTime [15] and TapNet [34], the sktime implementation was utilized[8]. We trained all the DL models for 200 epochs.

Table 3, depicts a rigorous comparison of our DeepHeteroIoT model against baseline ML and DL methods in terms of accuracy and weighted average F1-Score across 3 IoT datasets. Among all machine learning classifiers, only ensemble tree-based models, such as RF, GB, and XgBoost, seem to achieve competitive performance against our proposed deep learning model. Non-tree-based classifiers, such as LR, SVC, and KNN, demonstrate comparatively poor performance across all three datasets. Traditional CNN outperforms traditional RNN in UrbObs and IOWA datasets but lags behind in terms of the Swiss dataset. Notably, the popular InceptionTime model [15] exhibited very poor performance with a lower accuracy of 29.81% and an F1-score of 20.15% on the Swiss dataset. Another such model, TapNet [34], also failed to outperform the tree-based ensemble ML classifiers. The results shed light on the ineffectiveness of traditional time-series classifiers and traditional DL and ML models in heterogeneous IoT sensor scenarios, as discussed earlier in the literature review. Whereas, the potential of ensemble tree-based ML models is worth highlighting for their effective classification in heterogeneous IoT sensor scenarios. From the results, it is apparent that DeepHeteroIoT outperforms every learning-based baseline classifier by a significant margin, that is, achieving a greater average accuracy by absolute 9.29% and F1-score by 10.07 % across 3 datasets compared to the second-best performing models. Furthermore, RF emerged as the second-best model for both the Swiss and IOWA datasets, while GB outperformed RF by a slight margin on the UrbObs dataset.

[5] https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNN.
[6] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D.
[7] https://www.tensorflow.org/tutorials/images/cnn.
[8] https://www.sktime.net/en/latest/api_reference/classification.html.

Table 3. Comparison of performance against state-of-the-art baseline machine learning and deep learning models across 3 IoT datasets. The best results are in bold and the second best are underlined.

	UrbObs (Accuracy)	UrbObs (F1-Score)	Swiss (Accuracy)	Swiss (F1-Score)	IOWA (Accuracy)	IOWA (F1-Score)
CNN	78.13%	77.71%	47.11%	41.11%	83.01%	82.43%
RNN	50.31%	42.87%	58.65%	51.11%	65.01%	62.27%
InceptionTime [15]	64.98%	53.80%	29.81%	20.15%	87.98%	87.99%
TapNet [34]	53.44%	48.36%	42.31%	33.45%	55.68%	54.52%
LR	55.63%	51.75%	23.08%	15.52%	53.01%	50.41%
KNN	77.19%	73.78%	55.77%	52.36%	75.34%	74.18%
SVC	55.63%	47.98%	45.19%	33.85%	55.67%	46.95%
RF	86.56%	86.63%	73.07%	70.49%	92.00%	91.84%
GB	86.87%	86.74%	65.39%	64.53%	91.33%	91.35%
XgBoost	85.63%	85.27%	64.42%	65.36%	90.68%	90.63%
DeepHeteroIoT	**94.38%**	**94.27%**	**89.42%**	**89.07%**	**96.01%**	**95.93%**

Table 4. Comparison of performance across 3 IoT datasets with previous state-of-the-art study in IoT sensor data classification domain

	UrbObs (Accuracy)	UrbObs (macro F1-Score)	Swiss (Accuracy)	Swiss (macro F1-Score)	IOWA (Accuracy)	IOWA (macro F1-Score)
MACE (Top-k)	90.6%	87.9%	84.6%	83.2%	87.33%	87.40%
MACE (PF)	90.6%	87.9%	83.2%	83.2%	87.33%	87.40%
MACE (SoF)	90.6%	87.9%	82.7%	80.0%	90.33%	90.48%
MACE (brute-force)	92.2 %	88.7%	86.5%	84.0%	91.01%	91.04%
DeepHeteroIoT	**94.38 %**	**92.01%**	**89.42%**	**87.34%**	**96.01%**	**95.93%**

Additionally, the accuracy scores and macro-average F1-scores provided in Table 4 unequivocally demonstrate that DeepHeteroIoT consistently outperforms all alternative combinations involving the previous state-of-the-art machine learning model, MACE [21], across all datasets. Our model eventually outperforms MACE [21] by absolute 2.18 %, 2.92 %, 5 %, in accuracy and 3.31 %, 3.34 %, 4.89 % in macro-average F1-score on the respective Urban Observatory, Swiss Experiment, and IOWA ASOS datasets. It is worth noting that MACE employs an ensemble method alongside a brute-force strategy, which leads to considerable computational costs and undermines its scalability. On the other hand, our proposed deep learning model is a complete end-to-end solution which makes it scalable to the increase of dataset or problem size. For instance, the brute-force approach proposed in the MACE paper [21] took about 2 h and 17 min to train and build on the smallest dataset (Swiss), while our proposed deep learning model only required approximately 23 min and 55 s. Moreover, previous studies employing ensemble machine learning techniques [21,22] did not incorporate the learning of heterogeneous patterns from IoT sensor data during

the training phase. Our proposed deep learning architecture fills this gap by enabling end-to-end learning of complex contextual patterns in challenging IoT sensor data sequences.

5 Conclusion

In this study, we proposed a novel deep learning model that integrates learnable local features using CNN and global features using Bi-GRU in one end-to-end network, leading to significantly improved classification of heterogeneous IoT sensor data. Through rigorous experiments across three IoT datasets, we validated that our proposed deep learning model not only outperforms baseline machine learning and deep learning methods but also achieves state-of-the-art performance in classifying heterogeneous IoT sensor data. In the future, we would like to extend our model to multi-modal IoT datasets that occur from a diverse range of mobile computing environments.

Acknowledgements. This research was funded by the Australian Research Council (ARC) under ARC Discovery Project DP220101420 and partially supported by the Australian Government Research Training Program (RTP) Allowance Scholarship (Relocation).

References

1. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc. **31**, 606–660 (2017)
2. Borges, J.B., Ramos, H.S., Loureiro, A.A.: A classification strategy for Internet of Things data based on the class separability analysis of time series dynamics. ACM Trans. Internet Things **3**(3), 1–30 (2022)
3. Bradley, J., Barbier, J., Handler, D., Bradley, D.H.: The Internet of Everything is Happening Now (2013). https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy.pdf
4. Calbimonte, J.P., Corcho, O., Yan, Z., Jeung, H., Aberer, K.: Deriving semantic sensor metadata from raw measurements (2012)
5. Čulić Gambiroža, J., Mastelić, T., Nižetić Kosović, I., Čagalj, M.: Lost in data: recognizing type of time series sensor data using signal pattern classification. Int. J. Data Sci. Anal. 1–12 (2023)
6. Dua, N., Singh, S.N., Semwal, V.B., Challa, S.K.: Inception inspired CNN-GRU hybrid network for human activity recognition. Multimed. Tools Appl. **82**(4), 5369–5403 (2023)
7. Elazhary, H.: Internet of things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: disambiguation and research directions. J. Netw. Comput. Appl. **128**, 105–140 (2019)
8. Elsayed, N., Maida, A.S., Bayoumi, M.: Gated recurrent neural networks empirical utilization for time series classification. In: 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1207–1210 (2019). https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00202

9. Georgakopoulos, D., Jayaraman, P.P.: Internet of Things: from internet scale sensing to smart services. Computing **98**, 1041–1058 (2016)
10. Georgakopoulos, D., Jayaraman, P.P., Dawod, A.: SenShaMart: a trusted lot marketplace for sensor sharing. In: 2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC), pp. 8–17 (2020). https://doi.org/10.1109/CIC50333.2020.00012
11. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005). https://doi.org/10.1007/11538059_91
12. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
13. Interdonato, R., Ienco, D., Gaetano, R., Ose, K.: DuPLO: a dual view point deep learning architecture for time series classification. ISPRS J. Photogramm. Remote. Sens. **149**, 91–104 (2019)
14. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Min. Knowl. Disc. **33**(4), 917–963 (2019)
15. Ismail Fawaz, H.: InceptionTime: finding alexnet for time series classification. Data Min. Knowl. Disc. **34**(6), 1936–1962 (2020)
16. James, P.M., Dawson, R.J., Harris, N., Joncyzk, J.: Urban observatory environment. Newcastle University (2014). https://doi.org/10.17634/154300-19
17. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Min. Knowl. Disc. **15**, 107–144 (2007)
18. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. J. Intell. Inf. Syst. **39**, 287–315 (2012)
19. Madithiyagasthenna, D., et al.: A solution for annotating sensor data streams - an industrial use case in building management system. In: 2020 21st IEEE International Conference on Mobile Data Management (MDM), pp. 194–201 (2020). https://doi.org/10.1109/MDM48529.2020.00042
20. Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M.: Deep learning for IoT big data and streaming analytics: a survey. IEEE Commun. Surv. Tutor. **20**(4), 2923–2960 (2018)
21. Montori, F., et al.: A metadata-assisted cascading ensemble classification framework for automatic annotation of open IoT data. IEEE Internet Things J. (2023). https://doi.org/10.1109/JIOT.2023.3263213
22. Montori, F., Liao, K., Jayaraman, P.P., Bononi, L., Sellis, T., Georgakopoulos, D.: Classification and annotation of open internet of things datastreams. In: Hacid, H., Cellary, W., Wang, H., Paik, H.-Y., Zhou, R. (eds.) WISE 2018. LNCS, vol. 11234, pp. 209–224. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02925-8_15
23. van den Oord, A., et al.: WaveNet: a generative model for raw audio. In: 9th ISCA Speech Synthesis Workshop, p. 125 (2016)
24. Pan, M., et al.: Water level prediction model based on GRU and CNN. IEEE Access **8**, 60090–60100 (2020)
25. Postol, M., Diaz, C., Simon, R., Wicke, D.: Time-series data analysis for classification of noisy and incomplete Internet-of-Things datasets. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 1543–1550. IEEE (2019)

26. Rahman, M.W., Islam, R., Hasan, A., Bithi, N.I., Hasan, M.M., Rahman, M.M.: Intelligent waste management system using deep learning with IoT. J. King Saud Univ.-Comput. Inf. Sci. **34**(5), 2072–2087 (2022)
27. Rakthanmanon, T., Keogh, E.: Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 668–676. SIAM (2013)
28. Sajjad, M., et al.: A novel CNN-GRU-based hybrid approach for short-term residential load forecasting. IEEE Access **8**, 143759–143768 (2020)
29. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
30. Teh, H.Y., Kempa-Liehr, A.W., Wang, K.I.K.: Sensor data quality: a systematic review. J. Big Data **7**(1), 1–49 (2020)
31. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: a strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1578–1585 (2017). https://doi.org/10.1109/IJCNN.2017.7966039
32. Yamak, P.T., Yujian, L., Gadosey, P.K.: A comparison between Arima, LSTM, and GRU for time series forecasting. In: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence, pp. 49–55 (2019)
33. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 947–956 (2009)
34. Zhang, X., Gao, Y., Lin, J., Lu, C.T.: TapNet: multivariate time series classification with attentional prototypical network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 6845–6852 (2020)
35. Zhang, Y., Suleiman, B., Alibasa, M.J.: FedGroup: a federated learning approach for anomaly detection in IoT environments. In: Longfei, S., Bodhi, P. (eds.) MobiQuitous 2022. LNICST, vol. 492, pp. 121–132. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-34776-4_7

Federated Reinforcement Learning for Automated LoRaWAN Management in Industrial IoT

Ameer Ivoghlian(✉), Zoran Salcic, and Kevin I-Kai Wang

The University of Auckland, Auckland 1010, New Zealand
aivo002@aucklanduni.ac.nz, {z.salcic,kevin.wang}@auckland.ac.nz

Abstract. The scale of wireless networks is growing rapidly to support next generation industry 5.0 applications. LoRaWAN offers features such as low power consumption and long communication range, beneficial in many industrial applications. However, this comes at the cost of low data rates and link margins that rely on coding gain and careful network planning. These qualities make LoRaWAN networks vulnerable to interference and data-loss, which are particularly problematic in large-scale and mobile networks. It is vitally important that LoRaWAN networks are automatically and adaptively managed, to ensure that resources are optimally allocated, while satisfying multiple application requirements. In this paper, a novel multi-agent federated reinforcement learning framework is designed to enable knowledge sharing across multiple agents, and to resolve the issue of limited sampling coverage deep reinforcement learning. The goal is to allow individual node agents to adaptively control their network parameters to satisfy their own application requirements, while ensuring fair access across all nodes. The proposed method is evaluated against a non-federated approach. The results show that the proposed federated reinforcement learning approach can overcome the cold-start problem when new devices join a network. The proposed approach also demonstrates the ability to support mobile nodes travelling through locations, with changing network conditions, while maintaining communication performance.

Keywords: Federated Learning · Federated Reinforcement Learning · LoRaWAN · Industrial Internet of Things

1 Introduction

The trend of intelligent and collaborative manufacturing demands large-scale, geographically distributed networks to support the next generation Industrial Internet of Things (IIoT). LoRaWAN (LoRa Wide Area Network), like other LPWANs (Low Power Wide Area Networks), supports long-range and low power communication, and is becoming increasingly popular. However, the long-range communication comes at a price of low bandwidth and low data rate. In addition,

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 136–155, 2024.
https://doi.org/10.1007/978-3-031-63989-0_7

LoRaWAN operates in the international unlicensed ISM (Industrial, Scientific, and Medical) radio bands, which offer easy access, but can be very crowded [1]. Therefore, it is critical to ensure the network resource is automatically and optimally allocated to the connected devices to meet their application requirements.

The crowded spectrum, large number of nodes, and low data rate bring fundamental challenges to offering fair and smooth access to every connected node. The fact that different nodes may have different application requirements; for example different data delivery rate and power consumption, makes the problem even more complex. Therefore, an automated network management solution is necessary to ensure satisfaction of co-existing applications requirements and fair access to network resources for all connected nodes [2].

While LoRaWAN can be deployed in many industrial applications, in this study, we are targeting the specific scenario of indoor positioning and tracking applications, which is critical for warehouse and logistics management. For example, it is becoming increasingly important to be able to track goods/products, moving machines (e.g. forklifts), and employees, to ensure smooth logistics flow. Once location data are collected, they will need to be transmitted to a central server for higher level planning and management. LoRaWAN, which offers large coverage and low power consumption, is capable of collecting data from sensing devices (nodes) attached to stationery or moving objects.

However, location information has different importance to different objects and applications. For example, human location may be tracked for the purpose of safety and surveillance, and therefore it is critical to ensure regular data delivery so that location information is constantly updated. On the other hand, pallet tracking may only need to happen when a pallet changes its location, to prolong node operating life (i.e., lower power consumption) instead of constantly tracking its movement. Additionally, within a warehouse or logistics centre, due to different layouts, product content, and deployed network infrastructure, network performance can vary drastically at varying locations. Therefore, it is important that devices can adapt quickly to the changing environment to ensure reasonable communication performance.

While LoRaWAN has its own adaptive data rate (ADR) mechanism, it is designed for static network deployment and is not suited to rapid change after the initial configuration. This obviously does not offer good performance in a warehouse setting when the shelving contents are constantly changing and the objects/humans/machines are frequently moving. There are several machine learning based approaches developed to address this issue and offer better performance. However, these approaches also suffer some common issues. First, most machine learning algorithms require a large amount of training data, which is challenging to achieve in a typical IIoT system [3]. Second, the trained model is very often not generalised, and hence network performance may be degraded at certain physical locations within a warehouse, because no prior knowledge was acquired from that location [4].

In this study, we propose a new multi-agent-based federated reinforcement learning approach for automated and adaptive network management, specifically

targeting LoRaWAN. Each individual IoT node has its own agent which interacts with its environment and decides how its network parameters are configured to satisfy its requirements. While individual devices and agents conduct their own learning, periodically all the agent models are aggregated and a global model is redistributed back to each individual agent. The goal is to overcome the challenge of limited sampling coverage through knowledge sharing using federated learning, and therefore achieve a more generalised and robust model at the agent level [5].

The rest of the paper is organised as follows. Section 2 provides an overview of related works on existing federated and reinforcement learning, and their associated challenges. The motivating scenario and the overall framework architecture are presented in Sect. 3, followed by the detailed methodology and implementation explained in Sect. 4. Section 5 discusses the experimental set-up, scenarios, and results. We conclude this study with promising perspectives regarding future research directions in Sect. 6.

2 Related Works

2.1 Reinforcement Learning in Wireless Network Management

With the growing complexity of wireless networks, and the limited resources, research efforts have been focused on optimising the access and allocation of the available resources.

Huang et al. [6] presented a decentralised reinforcement learning approach using the policy hill climbing (PHC) algorithm to improve LoRaWAN performance by reducing the collision rate. The results are compared with the slotted ALOHA protocol and demonstrate better communication performance.

Nisioti et al. [7] proposed a different MAC layer design targeting wireless sensor networks in general that use the irregular repetition slotted ALOHA (IRSA) protocol. A mult-agent reinforcement learning approach is used where the agents are deployed on individual nodes. The proposed reinforcement learning is based on tabular Q-learning, with limited state and action spaces to facilitate the building of the Q table. The performance is further enhanced by using a virtual experience approach to achieve accelerated learning. The results demonstrate that the proposed approach is able to find a near-optimal solution for the IRSA protocol.

Yu et al. [8] proposed using multi-agent reinforcement learning to dynamically control the scaling factor and transmission power of individual nodes in LoRaWAN, with the aim to improve communication reliability by reducing the chances of collision, while simultaneously reducing power consumption. The state space is simple and consists of only a binary indicator of connection reliability for each node. The traditional tabular Q-learning is used and the reduced state space dimensionality allows efficient computation and construction of the Q table. The approach is evaluated on a small-scale static network that contains only 5 nodes (and their corresponding agents), and the results show both reduced collision and power consumption.

Fedullo et al. [9] used the SARSA (State-Action-Reward-State-Action) algorithm for LoRaWAN management. Similar to Q-learning, SARSA is a tabular-based method, and hence it is also limited to relatively small problem spaces, to ensure sufficient observation (or sampling) coverage. To address this issue, the training is done by sharing and populating the Q table between all nodes to achieve a more general and robust model. This is likely to be achieved by including the signal-to-noise ratio in the state space and the reward function, in order to generalise the learning with respect to node position. The approach is evaluated with the standard ADR approach of LoRaWAN, under a static network scenario. The results demonstrate improved communication reliability.

Generally, many reinforcement learning methods attempt to simplify the complexity of their state space to allow, for example, low computational complexity and creation of the Q table with limited sampling coverage. While the results demonstrate better performance, the resultant models lack generality and cannot manage network dynamics and node mobility very well. Unlike traditional reinforcement learning, deep reinforcement learning approaches have been developed to accommodate more complex problems and applied to a wider range of wireless network management applications.

Zhu et al. [10] looked at how to find an optimal transmission schedule in a cognitive IoT network scenario using deep reinforcement learning. The goal is to address the issue of poor communication efficiency in modern IoT applications when the spectrum is very crowded. The simulation results demonstrate that the proposed approach achieves similar performance as the strategy iteration algorithm, which can provide the optimal solution, but the computation complexity is not acceptable for wireless network applications.

Ilahi et al. [11] proposed using deep reinforcement learning (specifically Double DQN) to configure the communication parameters in LoRaWAN at runtime. The state space consists of several adjustable parameters and the distance between node and the base station. The action space consists of two configuration parameters, the scaling factor and the transmission power. The reward function considers packet delivery ratio (PDR), airtime, and min-max normalised transmission power (TXP). The approach was evaluated using a simulated network with 100 nodes and with mobile node scenarios. The simulation results demonstrate an improved PDR performance compared with the approach implemented in the LoRa-MAB simulation framework [12].

In [13], the authors proposed a single-agent reinforcement learning approach for automatically managing a large-scale LoRaWAN by considering application awareness and fairness in allocating network resources through a reward function. The application awareness is reflected in the reward function. Experimental results showed that the proposed reinforcement learning approach offers better network performance than standard ADR when interference is present.

In [14], the authors further proposed a multi-agent-based reinforcement learning approach with a reward function that reflects application awareness and network fairness at individual node level to allow quicker responses to changing network dynamics while supporting the overall network access fairness.

2.2 Federated Reinforcement Learning in Wireless Network Management

While deep reinforcement learning approaches are able to handle more complex problem spaces, they still suffer from limited sampling coverage. If certain situations have not been observed (or sampled) before, the model may perform poorly when those situations arise. Different approaches, such as [9], are proposed to share the knowledge of multiple nodes and to enrich one global agent's experience, with the compromise of limited consideration of individual node application requirements. Federated reinforcement learning is becoming a promising direction in solving the issue of sampling coverage. However, there are limited federated reinforcement learning efforts related to automating wireless network management.

One popular application for federated reinforcement learning is in the control of Radio Access Networks (RAN) that offers more interoperable network infrastructure. Cao et al. [15] proposed a federated reinforcement learning approach to find a user access control policy for open RAN, with the aim of optimising the long-term system throughput while preventing frequent user handover. In [16], the authors proposed horizontal federated reinforcement learning for finding the more efficient scheme for RAN slicing that will improve the overall network throughput.

Federated reinforcement learning is also more commonly applied to solving complex resource allocation and task scheduling problems. For example, Zhang et al. [17] proposed a method based on DQN to find the optimal transmission mode selection and resource allocation for cellular-based vehicular networks. Tianqing et al. [18] proposed a new concurrent federated reinforcement learning approach for finding a resource allocation policy that can improve the task completion speed and efficiency across an IoT-based edge computing scenario.

Unlike the existing approaches, this study aims to explore the possibility of using multi-agent federated reinforcement learning in finding the optimal network configuration for individual nodes within a LoRaWAN. The goal is to allow individual agents to automatically and adaptively configure their network parameters in response to their application requirements and changing network conditions, while sharing their knowledge/experience with other agents periodically to achieve more robust agent models across the entire network.

3 Federated Reinforcement Learning Framework for Adaptive Network Management

3.1 Motivating Scenario

In the next generation IIoT, the need for multiple applications to co-exist in an industrial site will be inevitable. These applications (and their associated nodes) are likely to occupy the same radio spectrum. Therefore, in this study, we aim to address two challenges that have been overlooked in modern IIoT systems, which are to tolerate multiple co-existing applications, and drastically changing

network dynamics within an industrial site. These two challenges can be easily illustrated by a real-world application of human/object tracking in a warehouse and logistics management context, as shown in Fig. 1.

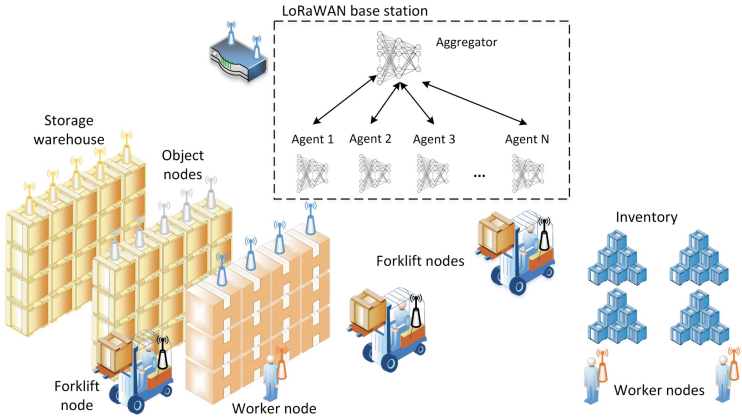


Fig. 1. The federated reinforcement learning framework architecture.

It has been demonstrated previously that location information and location-based services are extremely useful for warehouse/logistics control applications. However, different tracking applications may have different requirements. For example, an application for tracking workers may need to ensure a balanced performance of reliable communication and low energy consumption to ensure a worker's position can be tracked accurately and for a long period of time. In contrast, tracking moving machinery such as forklifts may have lesser energy consumption constraints, and tracking an object (e.g., a pallet) may have lesser real-time constraints. In addition, within a warehouse, depending on the deployed infrastructure, the warehouse layout, and the storage content, the network dynamics may change drastically across space causing variations of network performance.

In this study, three types of nodes are considered, namely worker nodes, forklift nodes, and object nodes, to reflect the following three application requirements:

1. Worker nodes: These nodes are typically wearable devices that have a high energy constraint. The nodes also need to reliably communicate their location data to the base station.
2. Forklift nodes: These nodes are installed on moving machinery, which have lower energy constraints, but require fast and reliable communication to ensure location information is transmitted to the base station.
3. Object nodes: These nodes are typically embedded devices that have very high energy consumption constraints. However, location information is only

needed at certain fixed locations so real-time communication is typically not necessary.

This motivating scenario will be used to establish the proposed federated reinforcement learning framework and will also be used in the evaluation to guide the experiment design.

3.2 Federated Reinforcement Learning Framework for Adaptive LoRaWAN Management (FRL-ALM)

Most existing (deep) reinforcement learning approaches suffer from the issue of sampling coverage. This is mainly due to the fact that individual agents are working in silos. In such cases, each agent can only observe a limited amount of data and hence the agent may perform poorly in an unknown situation. This limitation will be amplified in networks with mobile devices and with varying network conditions, like the ones in our motivating scenario.

Federated learning, as a distributed machine learning approach, is a natural step to be integrated with reinforcement learning to achieve experience/knowledge sharing across multiple nodes (and their corresponding agents). In this study, a unique Federated Reinforcement Learning framework for Adaptive LoRaWAN Management (FRL-ALM) is proposed to allow reinforcement learning to be conducted independently for each node (and agent) based on their own application requirements, while sharing their experience with one another so that the model of each individual agent can become more generalised and robust to changing locations and environmental conditions (i.e. network dynamics).

As mentioned in Sect. 1, LoRaWAN's long-range and low power features come at the price of limited data rate. Therefore, the proposed FRL-ALM framework is designed to incur only a small amount of communication overhead. In ordinary multi-agent reinforcement learning, the agents typically reside on the corresponding nodes, which allows nodes to operate independently according to their own situation. However, this also means that the nodes are running in silos and can only observe and react to what they have observed. In the proposed framework, while each node still has its corresponding agent, all agents are running on the base station.

Each agent still trains its own model based on its own state space (i.e. the environment observed by each node). Periodically, the aggregator will perform an aggregation process to collect and fuse all agent models. Once the aggregation is performed, the global model will be distributed to all agents and the agents will adopt the new model to update their network configurations (i.e. their action spaces) and carry on with their own learning until the next aggregation process. Since LoRaWAN typically utilises a star topology, a typical horizontal federated learning architecture is adopted across all agents and the aggregator.

Although running all agents on the base station will incur a higher workload, the base station typically has higher computational power and is more likely to have a permanent power source. More importantly, this architecture also significantly reduces the communication overhead of sending all the model parameters

of every agent to the base station. The aggregation process allows agents to share their knowledge with each other and hence the learned model will be more generalised and robust against changing network dynamics and node position. This will be evaluated and verified in our experiment with moving node scenarios. In addition, the global model also significantly reduces the training time, by providing any new node with the latest global agent model and is able to achieve a reasonable level of performance within a much shorter time-frame.

4 FRL-ALM Implementation

Extending from the Federated Reinforcement Learning framework presented in Sect. 3, this section discusses further, the actual design and implementation of the framework that can be fully integrated and deployed with the LoRaWAN protocol, in real-world industrial applications.

4.1 Multi-agent Reinforcement Learning

Reinforcement learning offers superior long-term performance for complex questions by interacting with the environment through actions and adjusting its agent's future action based on a reward function. Within a typical IIoT network, each individual node may have its own application requirements and therefore may require different network configurations to satisfy those requirements. In the previous study, multi-agent-based reinforcement learning has demonstrated better performance at both the individual agent and overall network level [14]. Individual agents will also allow their corresponding nodes to respond more quickly to their changing network dynamics and application requirements.

In this study, we continue with a multi-agent-based design. However, the agents are running in the base station instead of individual nodes in order to reduce excessive communication overhead. This is critical for LoRaWAN, which has a low data rate, and in some regions is further constrained by duty cycle limitations. In the proposed framework, each individual agent observes its own environment that is represented by the following state space, S (1):

$$S = \{PDR, ECP, T_{PDR}, T_{ECP}, U, PL\} \tag{1}$$

$$PDR = \frac{Packets\ received}{Packets\ sent} \tag{2}$$

$$ECP = V_{supply} \times I_{supply} \times T_{packet} \tag{3}$$

where the PDR is the observed packet delivery ratio of each node calculated by the base station as in (2); the ECP is the energy consumption per packet calculated by each node as in (3); the T_{PDR} and T_{ECP} are the target PDR and ECP set by the corresponding application requirements of each node; U is the overall network utilisation observed and calculated by the base station; and PL

is the average path loss of the node which gives an indication of the distance between the node and its base station and is calculated by taking the received signal strength and subtracting the transmission power value in the packet.

Referring to Fig. 2, to populate the state table, each node will inform the base station of its T_{PDR} and T_{ECP} during its initialisation phase (i.e. when it is first connected to the network). Each node will also include the ECP in each packet it sent. Other state space variables can be calculated by the base station, which has a better global view of the network. This allows for minimum communication overhead caused by agent learning. In comparison, if the agents are run on individual nodes, PDR, U, and PL will need to be provided to each individual node with more downlink communications, and the model parameters of all agents will need to be transmitted to the base station to enable aggregation, which significantly increases the overheads. The overall implementation and interaction of the base station and each node is shown in Fig. 2.

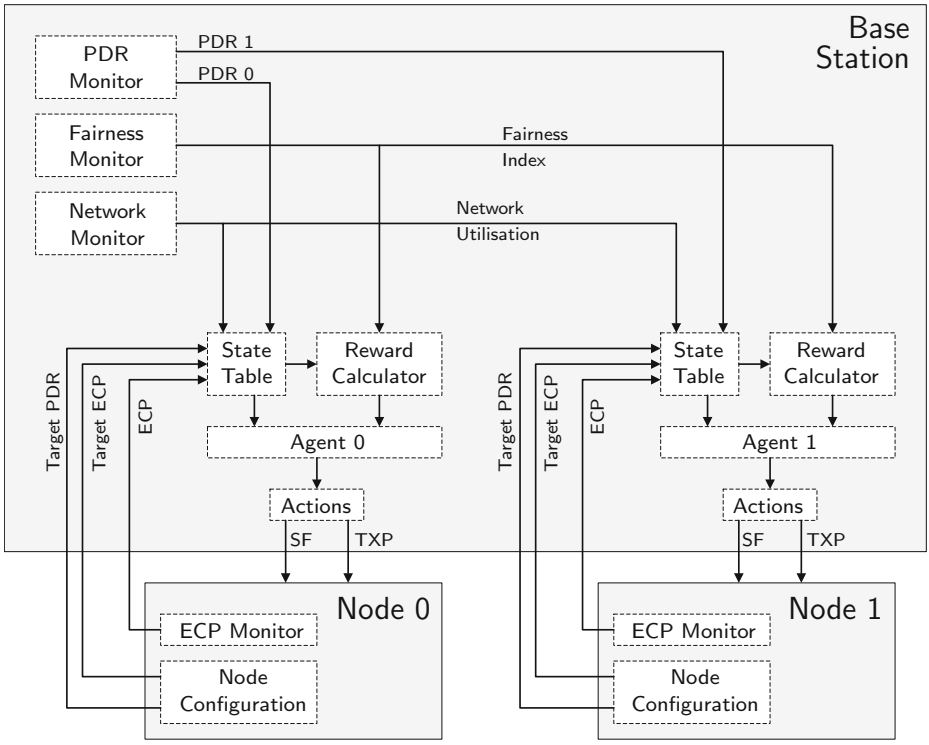


Fig. 2. FRL-ALM implementation architecture.

The purpose of the action space of each agent (and its corresponding node) is to interact with the environment and iteratively adjust its behaviour to satisfy its application requirements. In the proposed framework, the action space, A, is defined as:

$$A = \{SF, TXP\} \tag{4}$$

where SF is the spreading factor, which is a LoRaWAN-specific parameter that can be configured to balance the transmission range, energy consumption, data rate, and resistance to interference; and TXP is the transmission power. These two parameters can easily be configured and have direct impacts on the communication performance of the specific node.

The function of the reward function in reinforcement learning is to determine whether the performed action is beneficial towards the goal. In this study, the goals are to automatically manage and satisfy the application requirements of every individual node and at the same time ensure a fair resource allocation (or access) within the entire network. A unique reward function is shown in (8)–(6), which accommodates both the individual node's requirements and the overall network access fairness. It was first proposed in [14].

$$P_n = \prod_{m=0}^{M-1} Q_m \tag{5}$$

$$Q_m = \begin{cases} \frac{C_m}{T_m}, & C_m < T_m \\ 1, & C_m \geq T_m \end{cases} \tag{6}$$

$$f = \frac{(\sum P_n)^2}{N \sum (P_n^2)} \tag{7}$$

$$R = \frac{1 - 2|P_n - f| + |1 - 2f|}{1 + |1 - 2f|} \tag{8}$$

P_n, as defined in (5), captures how well a node is meeting its application requirements - the first goal. It is the product of the satisfaction Q for each metric m, with M representing the metric count. This research uses metrics PDR and ECP. Q_m is the ratio of the current measured metric value C_m to the configured target metric value T_m, capped at 1.

The second goal, fairness, is captured by f, as defined by (7). It is similar to Jain's Index [19], but has been adapted to accommodate differing application requirements. f represents how uniform each node's P_n is across the network.

Low levels of fairness are indicative of insufficient network capacity. In such cases, a reduction in network utilisation should be encouraged, which will likely result in a reduction of average P_n. To improve fairness, nodes with higher P_n should have a greater incentive to reduce. When there is excess network capacity available - a scenario which naturally results in higher fairness - nodes can increase their P_n without limit. It follows then, that when fairness is moderate, nodes with high P_n should be encouraged to reduce, and nodes with low P_n should be encouraged to increase, effectively motivating all nodes to target a moderate P_n.

To achieve this, the reward function R shown in (8), takes a triangular form, with the peak of the triangle at the current network fairness value f, and the

reward dropping off either side of this value. The $1-2|P_n - f|$ term in the reward function creates this triangular form. The remaining terms apply an offset and scaling factor to constrain the reward value between 0 and 1.

4.2 Federated Multi-agent Reinforcement Learning

In the proposed framework, the deep reinforcement learning approach, DQN (Deep Q Network), is adopted to implement each individual reinforcement learning agent. Different to the traditional Q-learning approach, which aims to derive a tabular result of the state-action function (i.e., the Q function), the DQN uses a neural network to approximate and estimate the Q values. While DQN has an improved ability to handle complex problems by replacing the Q table with a neural network, the method still suffers from the limited sampling space coverage and lack of generality. Since each individual agent only has limited observation of the network environment, the coverage of the sampling space is poor. This leads to poor performance when faced with network dynamics, when nodes are moving, or due to changing environmental conditions. Therefore, the federated learning approach is adopted in the proposed framework to facilitate experience sharing from every agent to ensure the agent model has better generality and is more robust against changing network dynamics.

Since all the agent training and model aggregation are performed in the base station, it is important to ensure a reasonable computational complexity. In the proposed framework, we adopted the typical FedAvg approach to aggregate the neural network parameters (i.e. the weights and biases of the DQN neural network). Figure 3 shows the overall federated learning process. Within the framework, individual agents operate independently, to interact with the network environment and adjust configuration parameters to iteratively attempt to satisfy their own application requirements. Periodically, the aggregator will request that all agents share their model parameters. Once parameters are received from all agents, aggregation will be performed and the aggregated model parameters will be distributed to all agents to update their models and action values. In the proposed framework, the node path loss component within the state space is introduced for the purpose of identifying and differentiating node behaviours incurred due to network dynamics at different distances from the base station. If a new node (and hence its agent) is joining the network, the latest global model parameters will be provided to the agent to allow the new node to achieve reasonable performance without excessive training time.

4.3 Framework Simulation Environment

The proposed approach has been developed in a modelling environment, consisting of a network simulation framework, and interconnected reinforcement learning tools. Building on the work of Bor et al. [20], a LoRaWAN modelling framework has been designed using Python and the SimPy library [21]. The framework builds the foundation for emulating LoRaWAN and evaluating the

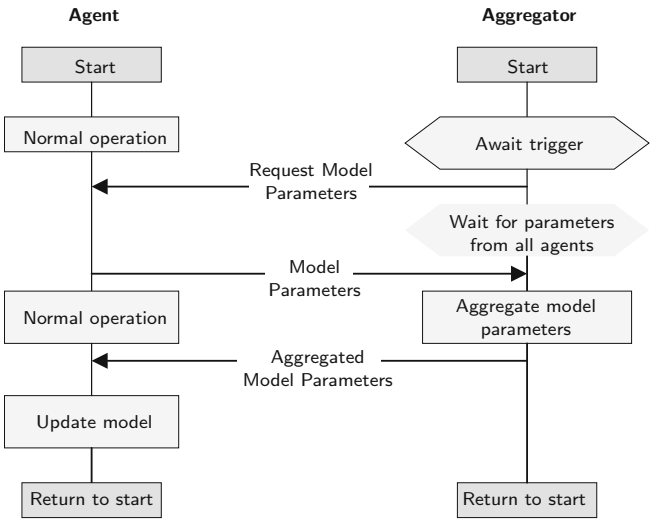


Fig. 3. FRL-ALM federated learning process.

proposed method by enabling the execution and interaction of multiple independent processes. In order to model the propagation environment, Log Distance path loss and shadow fading are used. By adjusting the path loss exponent and the standard deviation of the normally distributed shadow fading component, different transmission environments can be simulated. In this study, the path loss exponent has been configured to emulate an indoor warehouse environment.

The LoRa communication mechanisms are implemented following the processes and equations in Semtech's LoRa Application Notes [22,23], which allow for the calculation of packet link margin and detection of packet collision. The base station model utilised for our implementation contains an eight-channel radio, enabling reception and demodulation on eight channels, while simultaneously transmitting on one channel.

The reinforcement learning agents have been implemented using the Tensorforce library [24], which is built on top of Tensorflow [25]. The Tensorforce library implements a range of reinforcement learning algorithms and provides an interface for application integration. The modular structure of Tensorforce makes it possible to substitute other simulation frameworks, or even a physical LoRaWAN network, without changing the reinforcement learning components. The reinforcement learning algorithm can also be changed, without requiring changes to the simulation framework.

5 Experimental Evaluation

5.1 Experiment Setup

As mentioned in the previous sections, industrial environments are a targeted use case for LoRaWAN and the proposed approach. In this study, the target scenario is specifically the different types of tracking scenarios in a real-world warehouse and logistics management application.

LoRaWAN base stations typically have the capability of demodulating multiple (up to 8) channels simultaneously. From a network capacity perspective, each channel is independent of others. A single channel containing 32 nodes is equivalent to an eight-channel network that consists of 256 nodes. This characteristic allows simulation to be scaled down and accelerated. The scenario is simulated for an indoor environment, containing many absorbing bodies (e.g., stored goods) and reflective surfaces. The average path loss is therefore high, as is the variability. Table 1 lists the specific simulation parameters. The agent training parameters are also listed in Table 2, with the node positions randomly generated at the beginning of each episode.

Table 1. LoRaWAN Simulation Parameters

Parameter	Value
Nodes	32
Cell Radius	100 m
Node Distribution	Random Uniform
Spreading Factors	SF7 .. SF12
Transmission Power Range	−4 dBm .. 14 dBm
Transmission Power Steps	2 dB
Coding Rate	4/5
Frequency Band	915 MHz
Uplink Channel Bandwidth	125 kHz
Uplink Channel Spacing	200 kHz
Uplink Channels	1
Message Period	10 min
Payload Length	48 Bytes
Frame Length	532 bits
Path Loss Exponent (λ)	5
Shadow Fading Std Dev (σ)	9

5.2 Scenario I: Learning Efficiency for New Nodes

The first objective of the proposed federated reinforcement learning is to overcome the limitation of sampling coverage and achieving a more generalised agent

Table 2. Reinforcement Learning Training Parameters

Parameter	Value
Training length	100 episodes
Episode length	10,000 iterations
Memory	100 episodes
Batch Size	4
Discount Factor (γ)	0.1
Exploration (ϵ)	0.1
Learning Rate	e^{-3}
Update Frequency	1

model. A principle motivation for generalising the state space and aggregating the agent parameters is to allow agents to benefit from the experience of others. One advantage of the proposed approach can be demonstrated through the introduction of new nodes to the network, where these new nodes (or their agents) typically experience the cold start problem due to the agent's lack of observations, experiences, and interactions with its environment. A new node with no training, or with generic training that is poorly matched to its deployed situation, will initially perform poorly until it can learn about its environment. This is a very time-consuming process, especially in LoRaWAN where the communication and experience gathering is less frequent (e.g., a few times per hour).

The cold-start problem of new nodes can be significantly improved by taking advantage of the knowledge of other nodes in the network. In this scenario, the PDR target T_{PDR} is set to 0.5, and the target ECP T_{ECP} is set to an inverse normalised value of 0.5 (406 mJ per packet). This represents an application with balanced requirements on network reliability and energy consumption. Immediately after joining the network, the node's agent will request the current aggregated model parameters. The results presented in Fig. 4 and 5 show the PDR and ECP performance of a joining node with and without federated learning.

A node taking advantage of federated learning is compared with one which does not. It is strikingly obvious that, despite having similar levels of ECP, the PDR performance is vastly superior for the node that has the advantage of aggregated knowledge from the rest of the network. The poorly performing node is clearly using completely inappropriate parameters, and it will take significant time to explore the problem space and find appropriate ones.

The results of this scenario demonstrate that, with federated learning, the newly added nodes avoid the cold start problem, attaining high levels of performance immediately. It achieves this by taking advantage of the prior experience of all other nodes in the network.

5.3 Scenario II: Networks with Moving Nodes

The second objective of the proposed approach is to achieve a more robust model that can accommodate drastically changing network dynamics and node mobil-

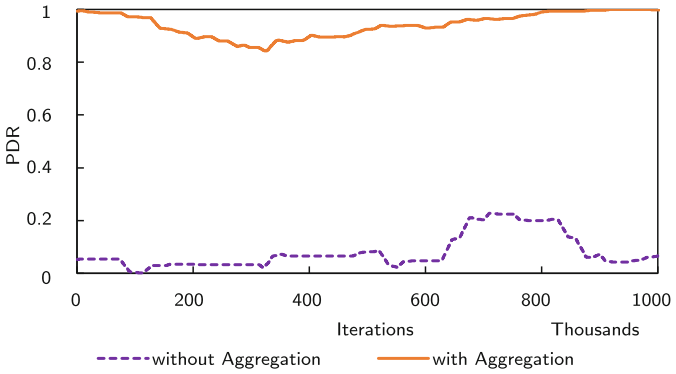


Fig. 4. PDR performance for a newly added node with and without federated learning.

ity. In a simplified scenario where a network consists of only stationary nodes, it is ok for individual nodes (and their agents) to operate independently in silos and only observe their nearby environment to achieve a reasonable performance. This is however rarely the case in real-world industrial applications, especially in the case of warehouse logistics tracking, where nodes are completely mobile with different tracking and communication requirements as discussed in Sect. 3. An agent must be able to respond to this movement without significant learning time. This experiment aims to evaluate if the proposed approach will allow nodes to maintain an acceptable level of performance, despite a significant change in their environment.

In this scenario, two types of mobile node are evaluated. The first represents a person moving in the warehouse on a forklift. As the node is on a person, power consumption is a concern, and so the metric targets are set at 0.4 and 0.6 for PDR and ECP respectively. The second node is attached to a forklift. It follows the same path as the first moving node but is attached to the vehicle. Its energy consumption is therefore less important, and the metric targets are set to 1.0 for PDR and 0.05 for ECP. The experiment is run with and without knowledge aggregation.

Figure 6 illustrates the PDR performance of the node attached to the person. The chart is segmented to show which position the node is in at the specific time. While both nodes show a reduction in performance when moving away from their initial position, when a node utilises federated reinforcement learning, it maintains a better and more consistent performance level and recovers more quickly when returning to its original position. Without federated learning, performance is worse, with more fluctuations and the PDR dropping to zero for some time. Even after returning to its origin, some instability can be observed, and it takes longer for the performance to return to its original level.

Figure 7 illustrates the PDR performance for the mobile node on the forklift. Unlike the pedestrian scenario, this scenario is focusing on PDR because there is

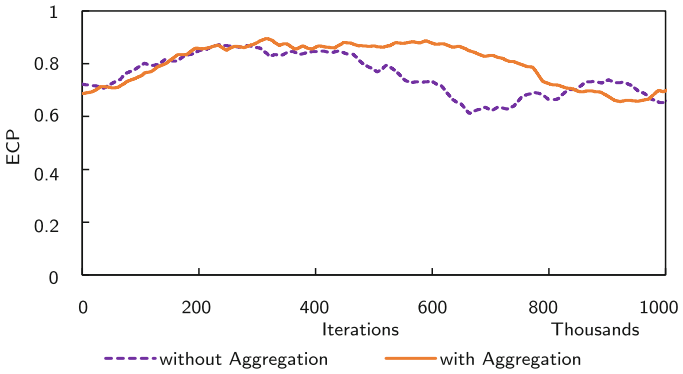


Fig. 5. ECP performance for a newly added node with and without federated learning.

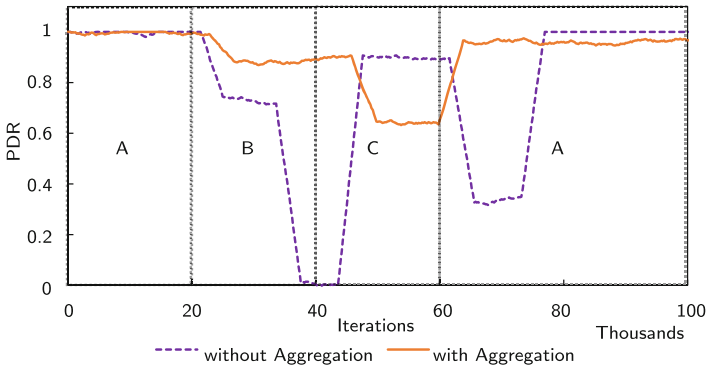


Fig. 6. PDR performance for mobile nodes on a person.

very little energy constraint. The node utilising federated reinforcement learning again offers very consistent performance, showing no impact at all caused by the node's movement. The PDR performance in the case where there is no federated learning clearly suffers more fluctuation. This performance fluctuation is likely to be worse when more complicated real-world factors such as environmental conditions, deployed infrastructures, and storage contents, are considered.

Figure 8 and Fig. 9 illustrate the ECP performance for the same two node types as above. ECP is presented as an inverted min-max normalised value, where values closer to 1 represent low energy consumption per packet, and values closer to 0 represent high energy consumption per packet. Two things can be observed in these results. Firstly, that the node utilising federated learning demonstrates intentional parameter change. It has changed position, and has chosen another set of appropriate parameters, informed by the experience of other nodes. These new parameters necessarily result in a change in ECP performance. A change which is less distinct for the node not utilising reinforcement learning.

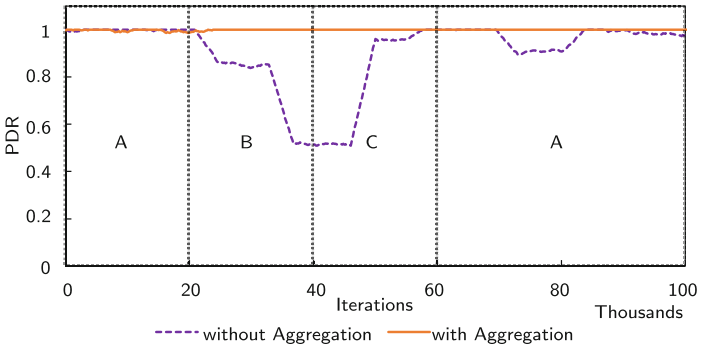


Fig. 7. PDR performance for mobile nodes on a forklift.

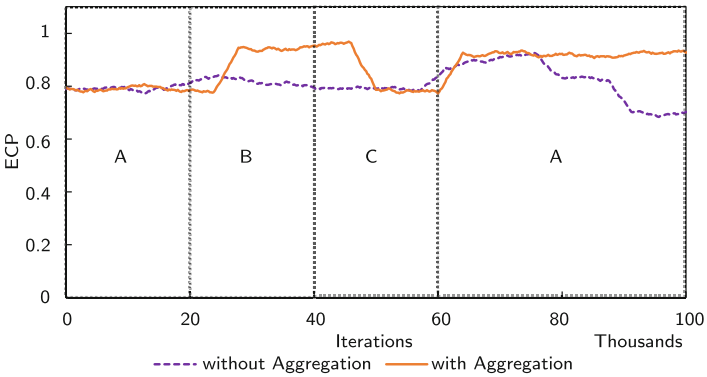


Fig. 8. ECP performance for mobile nodes on a person.

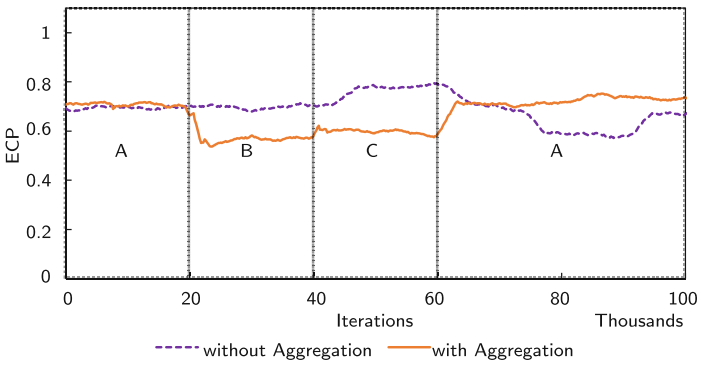


Fig. 9. ECP performance for mobile nodes on a forklift.

The second observation is the alignment with application requirements. ECP performance is clearly lower for the forklift node, as captured in its 1.0 and 0.05 metric targets for PDR and ECP respectively. This is the case regardless of whether reinforcement learning is utilised.

6 Conclusion

Future intelligent and collaborative manufacturing is underpinned by advances in next generation networks. While large-scale, long-range communication can be achieved to support large numbers of IIoT devices, the more pressing issue is how to automatically and adaptively manage the limited network resources to ensure fair access to all connected nodes, such that they can satisfy their application requirements. Many solutions have been proposed, ranging from traditional reinforcement learning to single and multi-agent deep reinforcement learning. While those solutions offer good performance in some environments, one common challenge that existing approaches are facing is the limited sampling coverage. In this study, we proposed a new multi-agent federated reinforcement learning framework specifically targeting LoRaWAN with mobile nodes. The proposed approach allows individual devices to have their own learning agents that control their independent network parameters to satisfy different application requirements. At the same time, agents can share their knowledge through federated learning to improve the agent model with better sampling coverage. The proposed method showed its ability to address the cold-start problem by achieving reasonable communication performance immediately at the start of agent operation. This also shows that the federated reinforcement learning agent model is more generalised. The proposed approach also demonstrated consistent communication performance when nodes are moving within the network with changing network dynamics. A more universal benefit is an improvement in learning efficiency. By regularly sharing experience between nodes, agents can build their models without direct experience, and much more quickly than if they attempted to gain that experience independently.

While the current results are promising, the adopted federated learning architecture and aggregation function are kept simple intentionally to reduce the overall computational complexity for the base station execution. In the future, we will investigate more sophisticated aggregation functions and other potential approaches to selectively aggregate agent models with higher credibility to achieve better performance with reasonable computation complexity.

References

1. Sherazi, H.H.R., Grieco, L.A., Imran, M.A., Boggia, G.: Energy-efficient LoRaWAN for Industry 4.0 applications. IEEE Trans. Ind. Inform. **17**(2), 891–902 (2021). https://doi.org/10.1109/TII.2020.2984549
2. Zhou, X., Yang, X., Ma, J., Wang, K.I.K.: Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT. IEEE Internet Things J. **9**(16), 14988–14997 (2022). https://doi.org/10.1109/JIOT.2021.3077937
3. Liang, W., Hu, Y., Zhou, X., Pan, Y., Wang, K.I.K.: Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT. IEEE Trans. Ind. Inf. **18**(8), 5087–5095 (2022). https://doi.org/10.1109/TII.2021.3116085

4. Lei, L., Tan, Y., Zheng, K., Liu, S., Zhang, K., Shen, X.: Deep reinforcement learning for autonomous internet of things: model, applications and challenges. IEEE Commun. Surv. Tutor. **22**(3), 1722–1760 (2020). https://doi.org/10.1109/COMST.2020.2988367
5. Sun, W., Lei, S., Wang, L., Liu, Z., Zhang, Y.: Adaptive federated learning and digital twin for industrial Internet of Things. IEEE Trans. Ind. Inf. **17**(8), 5605–5614 (2021). https://doi.org/10.1109/TII.2020.3034674
6. Huang, X., Jiang, J., Yang, S.H., Ding, Y.: A reinforcement learning based medium access control method for LoRa networks. In: 2020 IEEE International Conference on Networking, Sensing and Control (ICNSC), pp. 1–6 (2020). https://doi.org/10.1109/ICNSC48988.2020.9238127
7. Nisioti, E., Thomos, N.: Fast Q-learning for improved finite length performance of irregular repetition slotted ALOHA. IEEE Trans. Cogn. Commun. Netw. **6**(2), 844–857 (2020). https://doi.org/10.1109/TCCN.2019.2957224
8. Yu, Y., Mroueh, L., Li, S., Terré, M.: Multi-agent Q-learning algorithm for dynamic power and rate allocation in LoRa networks. In: 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5 (2020). https://doi.org/10.1109/PIMRC48278.2020.9217291. iSSN 2166-9589
9. Fedullo, T., Morato, A., Tramarin, F., Bellagente, P., Ferrari, P., Sisinni, E.: Adaptive LoRaWAN transmission exploiting reinforcement learning: the industrial case. In: 2021 IEEE International Workshop on Metrology for Industry 4.0 IoT (MetroInd4.0 IoT), pp. 671–676 (2021). https://doi.org/10.1109/MetroInd4.0IoT51437.2021.9488498
10. Zhu, J., Song, Y., Jiang, D., Song, H.: A new deep-Q-learning-based transmission scheduling mechanism for the cognitive Internet of Things. IEEE Internet Things J. **5**(4), 2375–2385 (2018). https://doi.org/10.1109/JIOT.2017.2759728
11. Ilahi, I., Usama, M., Farooq, M.O., Umar Janjua, M., Qadir, J.: LoRaDRL: deep reinforcement learning based adaptive PHY layer transmission parameters selection for LoRaWAN. In: 2020 IEEE 45th Conference on Local Computer Networks (LCN), pp. 457–460 (2020). https://doi.org/10.1109/LCN48667.2020.9314772. iSSN 0742-1303
12. Ta, D.T., Khawam, K., Lahoud, S., Adjih, C., Martin, S.: LoRa-MAB: a flexible simulator for decentralized learning resource allocation in IoT networks. In: 2019 12th IFIP Wireless and Mobile Networking Conference (WMNC), pp. 55–62 (2019). https://doi.org/10.23919/WMNC.2019.8881393. iSSN 2473-3644
13. Ivoghlian, A., Wang, K.I.K., Salcic, Z.: Data-driven adaptive network management with deep reinforcement learning. In: Proceedings of the 2021 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE (2021). iSBN 978-1-6654-2174-4
14. Ivoghlian, A., Salcic, Z., Wang, K.I.K.: Adaptive wireless network management with multi-agent reinforcement learning. Sensors **22**(3), 1019 (2022). https://doi.org/10.3390/s22031019. https://www.mdpi.com/1424-8220/22/3/1019
15. Cao, Y., Lien, S.Y., Liang, Y.C., Chen, K.C.: Federated deep reinforcement learning for user access control in open radio access networks. In: IEEE International Conference on Communications, ICC 2021, pp. 1–6 (2021). https://doi.org/10.1109/ICC42927.2021.9500603. ISSN 1938-1883

16. Liu, Y.J., Feng, G., Sun, Y., Qin, S., Liang, Y.C.: Device association for RAN slicing based on hybrid federated deep reinforcement learning. IEEE Trans. Veh. Technol. **69**(12), 15731–15745 (2020). https://doi.org/10.1109/TVT.2020.3033035
17. Zhang, X., Peng, M., Yan, S., Sun, Y.: Deep-reinforcement-learning-based mode selection and resource allocation for cellular v2x communications. IEEE Internet Things J. **7**(7), 6380–6391 (2020). https://doi.org/10.1109/JIOT.2019.2962715
18. Tianqing, Z., Zhou, W., Ye, D., Cheng, Z., Li, J.: Resource allocation in IoT edge computing via concurrent federated reinforcement learning. IEEE Internet Things J. **9**(2), 1414–1426 (2022). https://doi.org/10.1109/JIOT.2021.3086910
19. Jain, R., Chiu, D., Hawe, W.: A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems (1984)
20. Bor, M.C., Roedig, U., Voigt, T., Alonso, J.M.: Do LoRa low-power wide-area networks scale? In: Proceedings of the 19th International Conference on MSWiM, pp. 59–67. ACM (2016). https://doi.org/10.1145/2988287.2989163
21. Simpy - discrete event simulation for Python. https://simpy.readthedocs.io/. Accessed 30 Sept 2010
22. Semtech Corporation: SX1272/3/6/7/8: LoRa Modem Designer's Guide (2013). https://lora-developers.semtech.com/library/product-documents/
23. Semtech Corporation: LoRa Modulation Basics (2015). https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276
24. Kuhnle, A., Schaarschmidt, M., Fricke, K.: Tensorforce: a tensorflow library for applied reinforcement learning. Web page (2017). https://github.com/tensorforce/tensorforce
25. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Software available from tensorflow.org



A Hybrid Approach to Monitor Context Parameters for Optimising Caching for Context-Aware IoT Applications

Ashish Manchanda[1(✉)], Prem Prakash Jayaraman[1], Abhik Banerjee[1], Arkady Zaslavsky[2], Shakthi Weerasinghe[2], and Guang-Li Huang[2]

[1] Swinburne University of Technology, Melbourne, Australia
manchandaa45@gmail.com
[2] Deakin University, Melbourne, Australia

Abstract. Internet of Things (IoT) has seen a prolific rise in recent times and provides the ability to solve several key challenges faced by our societies and environment. Data produced by IoT provides a significant opportunity to infer context that is key for IoT applications to make decisions/actuations. Context Management Platform (CMP) is a middleware to facilitate the exchange and management of such context information among IoT applications. In this paper, we propose a novel approach to monitoring context freshness as a key metric, to improving the CMP's caching performance to support the real-time context needs of IoT applications. Our proposed hybrid algorithm uses Analytic Hierarchy Process (AHP) and Sliding Window technique to ensure the most relevant (as needed by the IoT applications) context information is cached. By continuously monitoring and prioritizing context attributes, the strategy adapts to IoT environment changes, keeping cached context fresh and reliable. Through experimental evaluation and using mock data obtained from a real-world mobile IoT scenario in Sect. 1, we demonstrate that the proposed algorithm can substantially enhance context cache performance, by monitoring the context attributes in real time.

Keywords: Context parameter monitoring · Cached context freshness · Real-Time IoT applications · System efficiency

1 Introduction

IoT applications play a vital role in collecting and processing data from various IoT sensors to provide insights into supporting decision making [1]. The data from IoT sensors is key to infer context that is required to provide relevant services to users. Context as defined by Anind Dey [5] is "Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

Supported by Australian Research Council (ARC) Discovery Project Grant DP200102299.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 156–174, 2024.
https://doi.org/10.1007/978-3-031-63989-0_8

Context Management Platform (CMP) has been proposed in the literature as a middleware to facilitate the exchange of context between context providers and IoT applications [6].

The increasing number of IoT applications and the need to provide context to meet the real-time demands of such application pose several performance challenges for CMPs. Delays in serving context to IoT applications can have severe consequences due to delays or failure in decision-making/actuation. Hence, it is a key challenge to manage the need and serve context to meet the demands.

To mitigate this delay and enhance the efficiency of the system, caching context becomes an essential strategy. By storing context information in the cache minimizes the time-consuming process of context retrieval from context providers, thus allowing for real-time access to context for IoT applications. This adaptation not only optimizes the performance of the CMP but also meets the growing demand for real-time, responsive IoT applications [7].

Caching context for IoT applications to utilize, presents a unique challenge, distinctly different from traditional data caching. The challenges associated with caching context for IoT applications stem from the complex and dynamic characteristics of context. Traditional data, which is typically static, can be cached using well-established metrics like frequency and size [10]. In contrast, context is very dynamic and is continuously changing which demands more nuanced caching strategies. A multitude of factors come into play when determining which context needs to be cached and when to refresh or discard it. As context-aware IoT applications rely on context for their operations, it's imperative to cache it efficiently to ensure quick and accurate retrieval of context. There is a need to support continuous monitoring of key metrics and a critical parameter called "context freshness" which we introduce in this paper. This metric ensures the current state of the context inside the CMP whether the context needs to be cached anew, evicted, or left unchanged in the cache memory. In summary, due to the dynamic nature of context, and the importance of keeping it fresh require specialized strategies designed specifically for IoT applications [2,4]. This guarantees that real-time sensitive IoT applications can make well-informed decisions, even in rapidly changing situations.

Let's consider a scenario of a smart IoT navigation system in Autonomous Vehicles(AVs) in a busy city. Here, an autonomous vehicle moves smoothly using a new IoT system for navigation. This system doesn't only use the vehicle's own sensors; it also uses information from other vehicles. For example, if a car detects a "Road Work" sign or an obstacle, it sends this information to a context management platform. This information is then available for all connected vehicles. So, another car coming to the same area knows about the obstacle before reaching it, making the drive safer and more efficient. By sharing information in this way, vehicles can better handle changing road situations.

Now current autonomous vehicles [8] often face difficulties in detecting situations such as road works, accidents, or heavy traffic congestion, because their internal models need vast amounts of training data to predict and navigate these situations effectively [9]. Contextual awareness of road conditions as dis-

cussed in the scenario above can greatly help in navigation and in some cases turning off auto-pilot. Given the ever-changing nature of such scenarios (road conditions) - with work starting or ending, lanes getting blocked or opened up - the underlying context representing the situations "Road Work" is constant changing (dynamic).

Research in monitoring the parameters of context caching, particularly for context freshness, is in its early stages [10,12]. The monitoring of query logs [11], as a solution has been considered in the past but this approach lacks the dynamism required to keep up with the ever-changing freshness of context that characterizes a situation like road works. This motivating scenario highlights the unique challenges posed by context caching in an IoT ecosystem in CMP, laying the groundwork for our research. In this paper, we propose the Context Freshness Monitoring System (CFMS) to monitor context-parameter/key-metric "context freshness" with the aim of enhancing cache optimization for context-aware IoT applications. Our work also includes comparisons with conventional algorithms such as RU and FIFO in Sect. 4. CFMS uses two algorithms for its function that are:

1. **Decision Supporting Algorithm(DSA)**: The main aim of this algorithm is to determine and prioritize the weights of context attributes using the Analytical Hierarchical Process (AHP) based on their relative significance. By doing so, it ensures that context-aware applications can efficiently process and act upon the most relevant context attributes in changing situations such as road work.
2. **Parameters Freshness Processing Algorithm(PFPA)**: The main aim of this algorithm is to take input of prioritize context attributes from decision supporting algorithm and cache them based on their respective weights. This algorithm maintains the freshness of cached context and makes real-time decisions on updating context attributes when they exceed predefined thresholds using sliding window mechanism. By doing so, it ensures that the most relevant and recent parameters are readily available for context-aware IoT applications, thereby optimizing performance and accuracy.

The CFMS integrates the strengths of its sub-algorithms to determine optimal caching decisions for context-aware IoT applications. The DSA employs the Analytical Hierarchical Process (AHP) to assign weights to context attributes, thereby identifying their relative importance. Subsequently, the PFPA leverages these weights to prioritize which attributes are cached. Using a sliding window mechanism, this algorithm ensures that the cache is consistently refreshed, maintaining only the most relevant and current context attributes. Collectively, the CFMS monitors the key metric "context freshness" which ensures that the caching mechanism is both efficient and contextually relevant.

The remainder of the paper is as follows. Section 2 reviews the Related Work. Section 3 describes the design and implementation of the Context Freshness Monitoring System (CFMS). Section 4 evaluates the CFMS model. Section 5 discusses and concludes the paper.

2 Related Work

Research in the field of IoT has recognized the importance of data caching for improving system performance. Traditionally, data caching strategies have relied on parameters like popularity and logs to enhance cache performance [12]. However, due to the static nature of such data and the rapidly changing IoT environment, these approaches may not always provide the most relevant or updated information [14,17]. Whereas context in IoT refers to the situational, environmental, or operational information that can affect the behavior or interpretation of IoT devices or data. For example, in a context-aware smart home, the context could include whether the resident is home or away, time of the day, day of the week, current weather conditions, etc. Context data involves a lot of factors and attributes that can significantly affect the operation of IoT devices and applications.

In recent advancements concerning context caching within IoT applications, a significant contribution was made in the paper [11]. This work focuses on the optimization of a context management platform (CMP), named CoaaS. Their novel system, ConCaf, utilizes context query logs and machine learning techniques to estimate the demand probability of context information. This approach is fundamentally different from our method, as their primary metric for caching decisions is derived from monitoring query logs and optimizing the context caching probability by adjusting the hyperparameters of the regression models. Although the results of their evaluation showed a significant improvement in the response time of CoaaS, the emphasis was on the proactive placement of context information based on query demand. In contrast, our work specifically zeroes in on the real-time monitoring of context attributes, aiming to maintain a context freshness metric inside the CMP. This distinction underscores our approach's commitment to addressing the transient and dynamic nature of contexts in IoT environments, ensuring that cached data remains relevant and timely.

Context-aware systems consider a multitude of context attributes that provide a high-level interpretation of data and allow systems to make intelligent decisions [18,19]. Nevertheless, the transient nature of these parameters introduces challenges in maintaining updated cache which, in turn, could create potential bottlenecks in system performance.

Several studies have explored adaptive caching strategies for context-aware systems [13], but they often assume that all cached context data is equally relevant. Such approaches do not take into account the freshness of the context or the real-time changes in context parameters. Despite numerous research studies focusing on the need for context-awareness in IoT systems, comprehensive studies emphasizing real-time monitoring of context parameters to maintain context freshness are limited [15,16].

There has been a good amount of literature, that predominantly investigates various techniques and technologies employed in real-time traffic monitoring across diverse contexts. Utilizing ontologies, frameworks like CARE, Wireless Sensor Networks, the fusion of visual perceptions with Convolutional Neural Networks (CNN), and IoT, these studies explore traffic evaluation, collision detec-

tion, and traffic parameter monitoring [23–29]. However, a conspicuous gap lies in these studies' attention to the crucial aspects of context caching and context freshness within IoT applications. Despite the substantial progress made in real-time decision-making for IoT applications, a comprehensive approach towards maintaining the context freshness and effective context caching strategies in IoT systems remains largely unexplored.

The algorithm discussed in the study efficiently incorporates context-specific content popularity to improve cache hits [20]. However, its reliance on content popularity as the main parameter can limit its effectiveness in IoT applications, where a plethora of context attributes and factors define the context. Our research seeks to address this gap by focusing on real-time monitoring of multiple context parameters.

The studies explore content caching strategies in IoT, focusing on balancing energy consumption and content freshness, and optimizing UAV-based network performance considering spatial parameters [21,22]. However, they overlook the potential of continuous real-time monitoring of diverse context parameters. Our approach enhances these strategies by incorporating comprehensive context parameter monitoring, leading to nuanced insights into context caching. This could significantly improve decision-making, system performance, and reliability in IoT applications.

Thus, there is a clear need for research that not only considers context caching but also addresses the dynamic nature of context parameters. This gap in literature justifies the necessity for our research, focusing on monitoring these context attributes/parameters and the implications on metric "context freshness" as a critical parameter in real-time IoT applications.

3 Context Freshness Monitoring System (CFMS)

In the realm of Internet of Things (IoT), the term "context" refers to any information that can be used to understand the situation of an entity. An entity can be a person, place, or object considered relevant to the interaction between a user and an application, including the user and the application themselves [5]. Each building block of this information is termed as a "context attribute".

Taking the IoT ecosystem with autonomous vehicles as an illustrative example, let's delineate the concepts:

1. **Context Attribute**: These are individual pieces of information, or parameters, that can provide insights about a specific aspect of the environment(situation) [30,31]. For contextual information such as "road work" as discussed in the motivating scenario, context attributes could be "speed of vehicle", "weather conditions", "traffic density", or "road quality". Each attribute offers insight into a specific aspect of the driving conditions.
2. **Situation/Context**: This represents a broader, composite understanding that is derived from the combination of various context attributes. For autonomous vehicles, a "road work" situation would be an amalgamation

of multiple context attributes: a decrease in vehicle speed, the presence of roadblocks or diversions, detection of construction machinery, and maybe an increase in dust levels [30,31].

The dynamism of these context attributes, especially in fluctuating scenarios such as road conditions, underscores the importance of "context freshness". Ensuring that the context attributes being considered are the most recent and relevant is crucial for accurate contextual awareness, particularly in real-time environments like autonomous driving. Hence, it becomes paramount to continually monitor and update the context cache, addressing the unique challenges of context freshness in IoT ecosystems.

Figure 1 demonstrates how the CFMS system works which can be incorporated into a Context Management Platform (CMP).

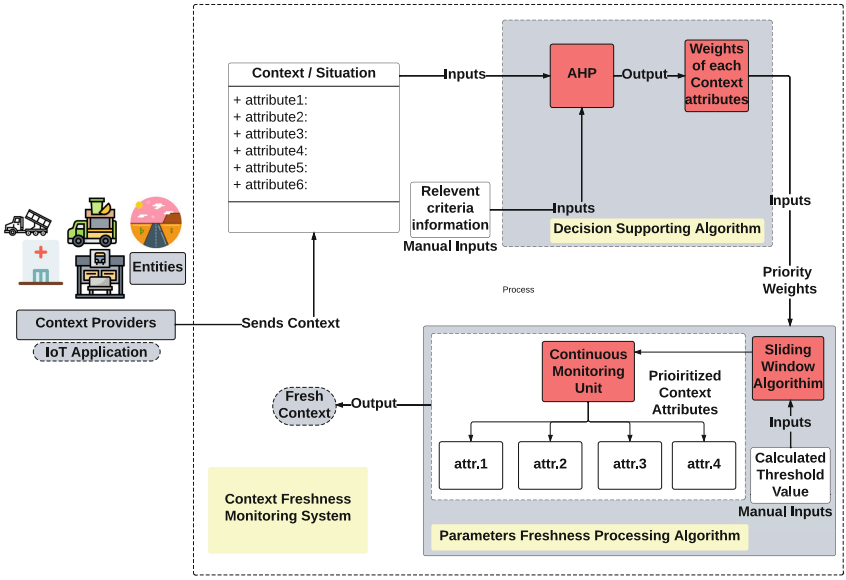


Fig. 1. Context Freshness Monitoring System.

In the proposed system, the contextual information with all the context attributes is received from IoT applications/context providers. The system incorporates two main algorithms. Firstly, the Decision Supporting Algorithm (DSA) explained in Sect. 3.1, extracts context attributes and forwards them to the Analytic Hierarchy Process (AHP) along with the relevant criteria information. Based on this, DSA calculates the weight of each context attribute. Once the weights are determined, the priority of the context attributes is ascertained. This prioritized information is then passed on to the Sliding Window Algorithm, which is an integral part of the Parameter Freshness Processing Algo-

rithm (PFPA) explained in Sect. 3.2. Within PFPA, the Sliding Window Algorithm processes the inputs received from DSA and a predefined threshold value. The output of this is then channeled to a Continuous Monitoring Unit for the cache module, which scrutinizes each parameter or context attribute against the given threshold. Subsequent caching decisions are then formulated based on observations from the continuous monitoring unit.

In managing these operations, the system, labeled as CFMS, directly monitors a key metric called "context freshness". This metric indicates the recency or timeliness of the context information. As a result, the output from the continuous monitoring unit ensures that the context information stored in the cache remains consistently fresh.

3.1 Decision Supporting Algorithm

The aim of this algorithm is to consider the entire context as fresh if the most essential context attributes are updated. To identify these pivotal context attributes, it's necessary to determine the weight (indicating its relevance to the context) of each attribute, achieved using the AHP method.

For the application of the Analytical Hierarchical Process (AHP) [33] to any given situation, the situation/context is perceived as the "goal" or the ultimate object of consideration. The distinct context attributes of the contextual information form the criteria, which are the parameters under examination in the context. These context attributes are defined as a set CA, such that CA = ca1, ca2, ..., can where n represents the total number of context attributes. Each context attribute or criterion cai is then weighed and ranked through AHP.

AHP uses a pairwise comparison matrix to determine the relative importance or "weight" of each context attribute which is then compared with every other context attribute, forming a set of n(n-1)/2 pairwise comparisons. This process results in a weight wi associated with each context attribute ci, forming a weighted set W = w1, w2, ..., wn. Once these weights are determined, a consistency check is performed using the consistency index (CI) and consistency ratio (CR). Provided the CR falls within an acceptable threshold, affirming consistent pairwise comparisons, the global priority weights are calculated.

A ranking is assigned to each context attribute based on its global priority weight, forming a ranked set R = r1, r2, ..., rn, where r1, r2, ..., rn represent the ranks of the context attributes ca1, ca2, ..., can respectively. The ranking ranges from 1 to n, with 1 being assigned to the attribute of highest priority or importance and n to the one with the least. The entire process provides an objective and quantified way to identify the key parameters for any context. It is generalizable and capable of handling 'n' number of context attributes as shown in Algorithm 1.

Algorithm 1. Decision Supporting Algorithm

```
procedure AHP_Decision(Goal, Criteria)
    hierarchy = "Goal": Criteria
    comparison_matrix = pairwise_comparison(Criteria)
    comparison_matrix = build_comparison_matrix(Criteria, comparison_matrix)
    priority_weights = calculate_priority_weights(comparison_matrix)
    CI = calculate_consistency_index(comparison_matrix, priority_weights)
    ACI = calculate_average_consistency_index(CI, num_criteria)
    CR = calculate_consistency_ratio(ACI, RI)
    if CR > threshold then
        return "Inconsistent pairwise comparisons"
    end if
    global_weights = calculate_global_priority_weights(priority_weights, hierarchy)
    sensitivity = perform_sensitivity_analysis(comparison_matrix, global_weights)
    ranking = rank_criteria(global_weights)
    return ranking
end procedure
```

The resulting pairwise comparison matrix is then processed to calculate the weight of each context attribute. This operation is performed for all context attributes as parameters and subsequently selects the top four parameters based on their calculated weights. These chosen parameters are then cached for quick reference and used as triggers for the Sliding Window Algorithm, ensuring the context freshness of cached context. Figure 2 displays the hierarchy of context "road work" with related criteria as parameters and weighing decisions of cache and no cache as alternatives.

3.2 Parameters Freshness Processing Algorithm

The Parameters Freshness Processing Algorithm with Sliding Window Algorithm(SWA) and Continuous Monitoring Unit(CMU) operates in tandem with the outputs from the DSA. The purpose of this algorithm is to ensure that prioratized context attributes remain fresh and updated according to a specified threshold inside the cache.

Inputs and Initializations. The algorithm requires two main inputs:

- **prioritized_attributes**: This is a list of context attributes ranked based on their weights, as determined by the DSA.
- **Threshold** T: A predefined threshold value which is used to determine when a context attribute value should be updated or removed from the sliding window.

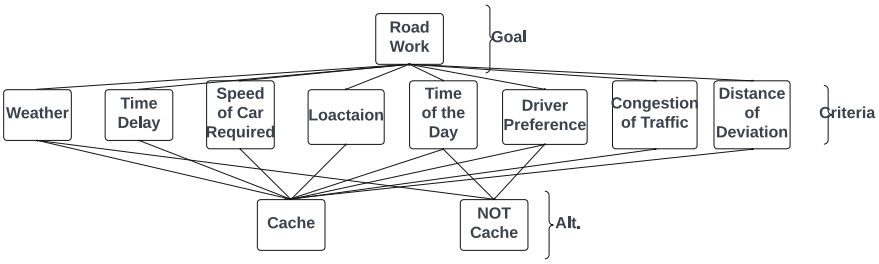


Fig. 2. Hierarchy of context "Road Work" for decision making.

Algorithm 2. Parameters Freshness Processing Algorithm with SWA, CMU, and Cache

Require: prioritized_attributes from DSA ▷ Ranked context attributes
Require: Threshold T
Initialize a deque *window* to hold the sliding window of data with size *window_size*
Initialize a set *cache* to hold cached data points
Initialize Continuous Monitoring Unit (CMU) with prioritized_attributes
procedure SWA(data_point)
    Add data_point to window
    while CMU.check_threshold(data_point) $> T$ **do**
        old_data_point = window.popleft()
        if old_data_point not in window **then**
            CMU.update_context_attribute(old_data_point)
            Remove old_data_point from cache
        end if
    end while
end procedure
for each data_point in prioritized_attributes **do**
    if data_point not in cache **then**
        Add data_point to cache
    end if
    SWA(data_point)
    if CMU.check_threshold(data_point) $> T$ **then**
        Update cache with the fresh value of data_point
    end if
end for

Upon initiation:

1. A deque, named **window**, is initialized to act as the sliding window, which holds context attributes. The size of this window is governed by the variable **window_size**.
2. The CMU is initialized with the prioritized context attributes obtained from DSA.

Sliding Window Algorithm (SWA) Procedure This procedure processes individual data points by:

1. Adding the context attribute value to the window.
2. Checking if the current value's freshness or value surpasses the threshold T as per the CMU's evaluation. If it does:
   (a) The oldest value is removed from the window.
   (b) If this removed value is no longer present in the current window, the CMU updates its records to reflect the removal of this context attribute.

Main Loop. For each value in the prioritized list of context attributes, the algorithm calls the SWA procedure. This ensures all attributes are checked and the sliding window is adjusted accordingly to maintain context freshness.

This algorithm thereby provides a systematic approach to handle and update context attributes based on their weights and freshness, ensuring that the most significant attributes are always up-to-date and maintained.

4 Experimental Evaluation

This section introduces an experimental evaluation designed to ascertain the efficiency of the caching mechanism in maintaining the context freshness by monitoring the context attributes identified through the Analytical Hierarchical Process (AHP). The effectiveness of the CFMS is measured using **cache hit** and **cache miss** ratios as key performance indicators, illuminating the efficiency of the caching system. Furthermore, a separate metric accounting for expired items in the cache - **cache expired** ratio - is introduced to specifically measure the context freshness aspect. The experimental setup is being tweaked in three primary ways:

1. Varying the threshold for the Sliding Window Algorithm: This manipulation allows the determination of the point at which data crosses the line from being fresh to being "stale". The identification of this threshold is critical for context freshness preservation.
2. Varying the volume of entries/contextual information entering the system from various IoT applications: This variation enables an evaluation of how effectively the system can handle different levels of demand, and whether or not this impacts context freshness and scalability.
3. Altering the cache capacity: By adjusting the scaling up or down the context memory as in [13], the influence of storage capacity on the system's performance and the context freshness as key metric of the cached context was tested.

Expanding upon the Analytical Hierarchical Process (AHP) methodology described in Sect. 3.1, the computed weights are illustrated in Fig. 3. These weights mark the priority of the different criteria related to the context "road work", indicating which context attributes are most important and thus guiding the caching process within the PFPA. This process, coupled with continuous monitoring, ensures the maintenance of context freshness.

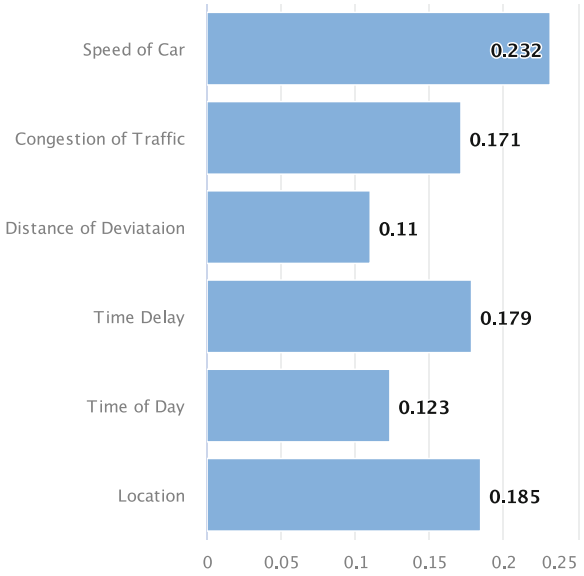


Fig. 3. Weights of context attributes of context "Road Work" after DSA.

The outputs generated from the DSA are subsequently used as inputs to monitor the context freshness of context attributes(parameters) via the Algorithm 2. This algorithm primarily focuses on maintaining a sliding window of parameters and tracking the performance of caching in terms of cache hit-and-miss ratios.

4.1 Varying the Threshold for the Sliding Window Algorithm

In this subsection, the performance of the caching system is evaluated in terms of cache hit and cache miss ratios, with the key variable being the threshold value set for the "sliding window algorithm", which means after the threshold is reached, the IoT data corresponding to the context attribute will be considered as stale and evicted from cache. The threshold is systematically varied from 10 min to 25 min, in increments of 5 min, as indicated in Table 1.

Table 1. Cache hit and cache miss ratios at different thresholds.

Threshold	Value			
	10	15	20	25
Cache hit	174	179	184	186
Cache miss	26	21	16	14
Ratio	6.7	8.5	11.5	13.3

The results of this variation, visualized in Fig. 4, suggest a trend of increasing cache hits as the threshold value rises. After analysis, a 20-min threshold has been selected for the experiments conducted in the subsequent sections. It's important to note that post a threshold of 22 min, no significant impact or changes were observed in the system's performance. This threshold selection ensures an optimal balance between cached context freshness and computational efficiency.

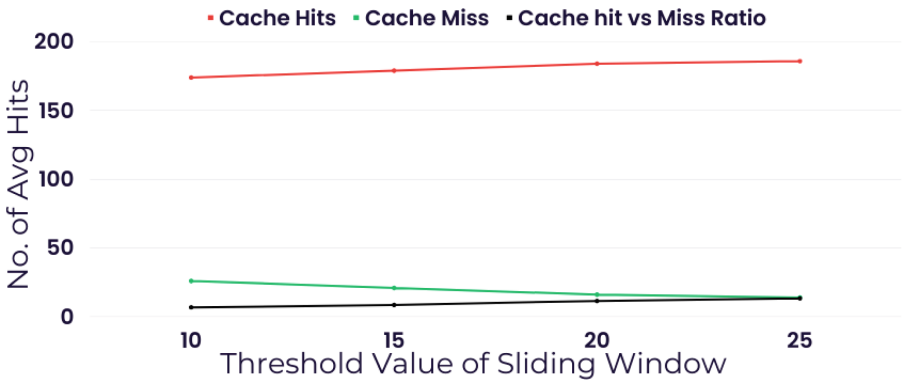


Fig. 4. Comparison of Cache hit with a threshold value of Sliding Window.

4.2 Changing the Size of the Incoming Entries (Contextual Data)

In this subsection, the size of the incoming entries load was systematically varied, testing with 150, 250, 350, and 500 for each threshold from 10 to 25 min in increments of 5 min as shown in Table 2. The findings reveal a consistent pattern across all test cases. With an increasing number of entries, both cache hit and cache miss counts increase, but the cache hit ratio remains relatively consistent indicating that the "hybrid approach" also supports scalability.

Table 2. Cache hit and cache miss ratios at different number of entries.

No. of queries	Threshold	Cache Hit	Cache miss	Cache Hit Ratio
150	10	528	72	7.33
	15	542	58	9.34
	20	555	45	12.33
	25	561	39	14.38
250	10	880	120	7.33
	15	904	96	9.41
	20	925	75	12.33
	25	934	66	14.15
350	10	1232	168	7.33
	15	1266	134	9.44
	20	1296	104	12.46
	25	1309	91	14.38
500	10	1761	239	7.36
	15	1809	191	9.47
	20	1851	149	12.42
	25	1870	130	14.38

From the 'Cache Hit Ratio' heatmap (Fig. 5), we can observe a pattern of increasing cache hit ratio with an increasing threshold for all entry sets. For a threshold of 10, the cache hit ratio remains relatively steady around 7.33 to 7.36 across all entries. As the threshold increases to 15, there is a notable improvement in the ratio, reaching up to 9.47 for 500 entries. When the threshold is increased further to 20 min, the ratio experiences an additional boost to a range of approximately 12.33 to 12.46. Interestingly, upon reaching a 25-min threshold, the ratio increases to around 14.38 for all query sets, except for 250 queries where it marginally drops to 14.15. This discrepancy could be attributed to various factors including caching policies, size of the cache, or variability in the access patterns.

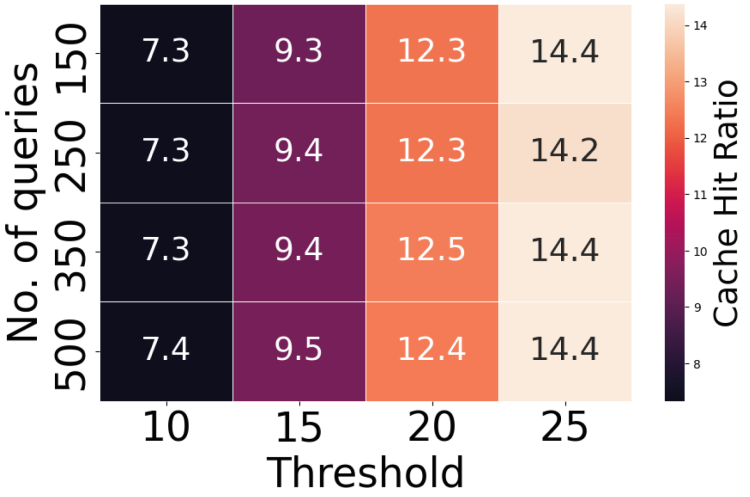


Fig. 5. Comparison of Cache hit ratio with Number of Entries and Thresholds.

These findings, illustrated in the heatmap, affirm the choice of a 20-min threshold as a suitable point. While the cache hit ratio generally improves with an increase in threshold, the gains beyond the 20-min mark are relatively minor. This confirms the trade-off between context freshness and computational efficiency, and indicates the diminishing returns of increasing the threshold beyond 20 min. Therefore, a 20-min threshold appears to be the optimal point for maintaining an efficient cache system, given the current configuration and workload.

4.3 Modifying the Cache Capacity

In this subsection, the cache capacity is adjusted to varying capacity - 20%, 60%, and 80%. This test keeps the number of incoming entries constant at 500 and sets the threshold at 20 min. The experiment aims to compare the efficiency of using DSA & PFPA in caching with other caching algorithms, namely LFU (Least Frequently Used) and RU (Recently Used). The results of this comparison are displayed in Fig. 6.

A careful analysis of the results reveals that as the cache size increases from 20% to 80%, the use of DSA & PFPA experiences a slight increase in average cache hits, from 91 to 95. Comparatively, the LFU algorithm exhibits a more substantial increase in average cache hits, growing from 23 to 76 with the increase in cache size. Similarly, the RU algorithm demonstrates a significant rise in average cache hits, from 18 to 78, as the cache size increases.

These results suggest that while increasing cache capacity does enhance average cache hits for all algorithms, the use of DSA & PFPA appears less sensitive to changes in cache capacity. This indicates more efficient utilization of cache

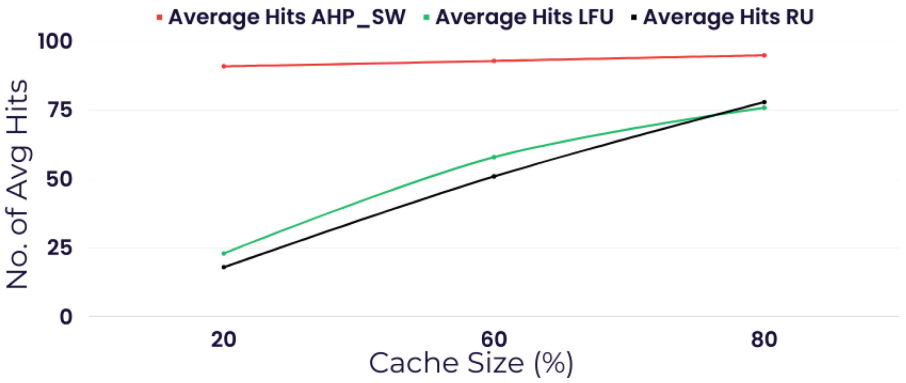


Fig. 6. Comparison of the average number of cache hits for different cache storage limit.

space by using DSA & PFPA in caching which takes into account for monitoring context attributes and maintaining the context freshness, thereby reinforcing its suitability and advantage in real-time IoT applications, where memory resources may be limited.

4.4 Evaluating the Cache Expired Ratio

Figure 7 provides a comparative view of the cache expired ratio - a measure of context freshness metric - with two different caching algorithms: DSA & PFPA used in caching, Recently Used (RU), and First In, First Out (FIFO). An essential observation from the figure is the distinct capability of the monitoring ability of DSA & PFPA to perform efficiently even when the cache size is as low as 20. This significant feature underscores its potential applicability in scenarios like network edge or fog computing, where memory constraints are prevalent. As more systems aim to achieve data/process localization and real-time operations, the DSA & PFPA's "context freshness" monitoring proficiency at low cache sizes becomes a vital contribution of this work.

As the cache size increases from 20 to 80, the cache expired ratio calculated using DSA & PFPA remains consistently low, highlighting its superior ability to maintain context freshness. Even with increasing cache size, this monitoring algorithm ensures storage of only the most recent and relevant context, indicating effective cache management.

Conversely, the RU algorithm, starting with a high cache expired ratio of 0.322 at a cache size of 20, shows a decrease to 0.195 as the cache size expands to 80. While this indicates some improvement in context freshness with a growing cache size, it is still less efficient than DSA & PFPA. FIFO, which starts with a cache expired ratio of 0.2 at a cache size of 20, observes a significant drop to 0.0095 at a cache size of 80. This sharp decrease, however, may not necessarily signify high context freshness metric, especially given its initially high ratio. DSA

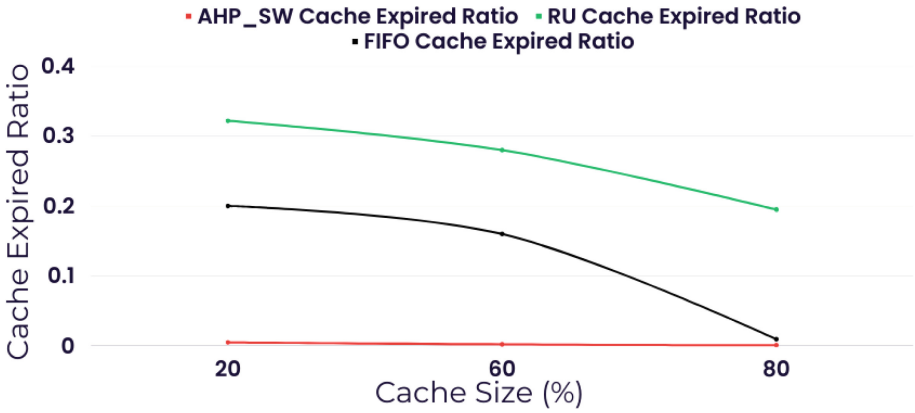


Fig. 7. Comparison of the cache expired ratio for different cache storage limit.

& PFPA establishes its robustness and efficiency by continuously monitoring parameters and maintaining the context freshness as a preferred mechanism for caching algorithm for real-time IoT applications, especially in environments with memory constraints.

5 Discussion

The evaluation has presented significant findings. Using DSA and PFPA to monitor parameters and maintaining the metric of "context freshness" has improved context caching. One important observation was made during the threshold experiment. The cache hit rate increased as the threshold value went up. However, after reaching the 20-min point, there was only a small improvement. This highlights the importance of finding a balance between maintaining context freshness and optimizing computational efficiency. It's essential to choose a threshold that not only increases hit rates but also preserves context freshness, especially for real-time IoT applications. Based on the evaluation, a 20-min threshold appears to provide this balance.

Additionally, altering the amount or number of contextual information entries into the CFMS demonstrated the strong scalability of the proposed hybrid approach. As the number of entries increases, the proportion of cache hits stayed steady because increasing the entries didn't affect the CFMS. The system continued to monitor the parameters linked to the new entries. This consistency, even with changing entries, supports the system's reliability, making it appropriate for various IoT applications. Yet, the most notable observation came from the changes in cache capacity. While all other algorithms (RU, LFU) performed better with larger cache sizes, but the performance of caching context using DSA & PFPA maintaining the context freshness was less dependent on increasing the cache size. This adaptability to size variations can be very useful in time-sensitive

IoT applications, particularly in edge computing environments where memory may be limited.

Finally, the cache expired ratio, which monitors the metric "context freshness", showed that DSA & PFPA performed notably better. Even with smaller cache sizes, the CFMS effectively kept the cache expired ratio low. This capability is important for real-time IoT applications, especially in environments with limited memory restrictions. This observation highlights the potential of DSA & PFPA as a promising approach, suggesting more research and testing in different IoT environments.

6 Conclusion and Future Work

The study indicates that by monitoring parameters by maintaining metric "context freshness", which can keep the context fresh can significantly improve the performance of caching in real-time context-aware IoT applications. The system, which uses DSA & PFPA, monitors the prioritized parameters to ensure the context remains fresh, leading to a stable cache hit rate. This hybrid approach provides a dynamic response to environmental changes, ensuring that both the freshness of the cached context and the system's overall efficiency are enhanced.

Looking forward, there are several exciting possibilities for further enhancing of CFMS. One such improvement could be the integration of fuzzy AHP, which would provide a more nuanced understanding of the relative importance of each criterion/parameter. The incorporation of decision trees could further refine this approach, enabling more complex and layered decision-making. Plans are also underway to incorporate this enhanced system into a broader context management platform like CoaaS.

Acknowledgement. Support for this publication from the Australian Research Council (ARC) Discovery Project Grant DP200102299 is thankfully acknowledged.

References

1. Manchanda, A., Lee, K., Poznanski, G.D., Hassani, A.: Automated adjustment of PPE masks using IoT sensor fusion. Sensors **23**(3), Article no. 1711 (2023)
2. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **39**, 164–176 (2015)
3. Forkan, A.R.M., et al.: Mobile IoT-RoadBot: an AI-powered mobile IoT solution for real-time roadside asset management. In: Proceedings of the 28th Annual International Conference on Mobile Computing and Networking, pp. 883–885. Association for Computing Machinery (2022)
4. Ryan, N., Pascoe, J.: From context-aware to context-driven: a paradigm shift. In: Proceedings of the Workshop on Situated Interaction in Ubiquitous Computing (2000)
5. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**, 4–7 (2001)

6. Hassani, A., et al.: Context-as-a-service platform: exchange and share context in an IoT ecosystem. In: Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 385–390 (2018). https://doi.org/10.1109/PERCOMW.2018.8480240
7. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the Internet of Things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)
8. Krasniqi, X., Hajrizi, E.: Use of IoT technology to drive the automotive industry from connected to full autonomous vehicles. IFAC-PapersOnLine **49**(29), 269–274 (2016)
9. Iyer, L.S.: AI enabled applications towards intelligent transportation. Transp. Eng. **5**, 100083 (2021). ISSN 2666-691X
10. Al-Ward, H., Tan, C.K., Lim, W.H.: Caching transient data in Information-Centric Internet-of-Things (IC-IoT) networks: a survey. J. Netw. Comput. Appl. **206**, 103491 (2022). ISSN 1084-8045
11. Khargharia, H.: Probabilistic analysis of context caching in the Internet of Things applications. IEEE Xplore (2022). https://ieeexplore-ieee-org.ezproxy-b.deakin.edu.au/document/9860212
12. Weerasinghe, S., Zaslavsky, A., Loke, S.W., Hassani, A., Abken, A., Medvedev, A.: From traditional adaptive data caching to adaptive context caching: a survey. arXiv preprint arXiv:2211.11259 (2023)
13. Weerasinghe, S., Zaslavsky, A., Loke, S.W., Medvedev, A., Abken, A.: Estimating the dynamic lifetime of transient context in near real-time for cost-efficient adaptive caching. SIGAPP Appl. Comput. Rev. **22**(2), 44–58 (2022)
14. Bettini, C., et al.: A survey of context modelling and reasoning techniques. arXiv preprint arXiv:1611.01800 (2010)
15. Stankovic, J.A.: Research directions for the Internet of Things. arXiv preprint arXiv:1401.0003 (2014)
16. Gu, T., Pung, H.K., Zhang, D.Q.: A middleware for building context-aware mobile services. arXiv preprint arXiv:0512.2656 (2005)
17. Rizwan, P., Suresh, K., Babu, M.R.: Real-time smart traffic management system for smart cities by using Internet of Things and big data. In: Proceedings of the 2016 International Conference on Emerging Technological Trends (ICETT), pp. 1–7 (2016). https://doi.org/10.1109/ICETT.2016.7873660
18. Shekade, A., Mahale, R., Shetage, R., Singh, A., Gadakh, P.: Vehicle classification in traffic surveillance system using YOLOv3 model. In: Proceedings of the 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 1015–1019 (2020). https://doi.org/10.1109/ICESC48915.2020.9155702
19. Fernández-Sanjurjo, M., Bosquet, B., Mucientes, M., Brea, V.M.: Real-time visual detection and tracking system for traffic monitoring. Eng. Appl. Artif. Intell. (2019). https://doi.org/10.1016/j.engappai.2019.07.005
20. Müller, S., Atan, O., van der Schaar, M., Klein, A.: Context-aware proactive content caching with service differentiation in wireless networks. IEEE Trans. Wirel. Commun. **16**(2), 1024–1036 (2017)
21. Xu, C., Wang, X.: Transient content caching and updating with modified harmony search for Internet of Things. Digit. Commun. Netw. **5**(1), 24–33 (2019)
22. Fang, T., Tian, H., Zhang, X., Chen, X., Shao, X., Zhang, Y.: Context-aware caching distribution and UAV deployment: a game-theoretic approach. Appl. Sci. **8**(10), Article no. 1959 (2018)

23. Mcheick, H., Khreis, M., Al-Kalla, M., Sweidan, H.: CARE framework: context-aware reliable engine health focus on traffic monitoring system. In: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, pp. 411–416 (2014). https://doi.org/10.1109/UKSim.2014.76
24. Goel, D., Pahal, N., Jain, P., Chaudhury, S.: An ontology-driven context aware framework for smart traffic monitoring. In: 2017 IEEE Region 10 Symposium (TENSYMP), pp. 1–5 (2017). https://doi.org/10.1109/TENCONSpring.2017.8070059
25. Serdaroglu, K.C., Baydere, S.: On the data freshness for IoT traffic modelling in real-time emergency observation systems. In: 2018 23rd Conference of Open Innovations Association (FRUCT), pp. 335–340 (2018). https://doi.org/10.23919/FRUCT.2018.8588029
26. Faro, A., Giordano, D., Spampinato, C.: Evaluation of the traffic parameters in a metropolitan area by fusing visual perceptions and CNN processing of webcam images. IEEE Trans. Neural Netw. **19**(6), 1108–1129 (2008). https://doi.org/10.1109/TNN.2008.2000392
27. Pande, A., Abdel-Aty, M.: Assessment of freeway traffic parameters leading to lane-change related collisions. Accid. Anal. Prev. **38**(5), 936–948 (2006). https://doi.org/10.1016/j.aap.2006.03.004
28. Beymer, D., McLauchlan, P., Coifman, B., Malik, J.: A real-time computer vision system for measuring traffic parameters. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 495–501 (1997). https://doi.org/10.1109/CVPR.1997.609371
29. Thakur, T.T., Naik, A., Vatari, S., Gogate, M.: Real-time traffic management using Internet of Things. In: 2016 International Conference on Communication and Signal Processing (ICCSP), pp. 1950–1953 (2016). https://doi.org/10.1109/ICCSP.2016.7754512
30. Padovitz, A., Loke, S.W., Zaslavsky, A.: Towards a theory of context spaces. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 38–42 (2004). https://doi.org/10.1109/PERCOMW.2004.1276902
31. Boytsov, A., Zaslavsky, A.: From sensory data to situation awareness: enhanced context spaces theory approach. In: Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pp. 207–214 (2011). https://doi.org/10.1109/DASC.2011.55
32. Hassani, A., Medvedev, A., Delir Haghighi, P., Ling, S., Zaslavsky, A., Prakash Jayaraman, P.: Context definition and query language: conceptual specification, implementation, and evaluation. Sensors (Basel) **19**(6), 1478 (2019). PMID: 30917602; PMCID: PMC6470624
33. Afshari, A., Mojahed, M., Yusuff, R.: Simple additive weighting approach to personnel selection problem. Int. J. Innov. Manag. Technol. **1**, 511–515 (2010). https://doi.org/10.7763/IJIMT.2010.V1.89

LOADHoC: Towards the Automatic Local Distribution of Computation Using Existing IoT Devices

Shaine Christmas[1(✉)], Kevin Lee[1], and Jean-Guy Schneider[2]

[1] Deakin University, Geelong, VIC 3220, Australia
schristmas@deakin.edu.au
[2] Monash University, Clayton, VIC 3800, Australia

Abstract. Mobile devices often use offloading nodes to reduce the amount of power used locally, or because of the low computational power of the device. Mobile devices can take advantage of offloading to reduce the power usage and increase the battery life of the device. Cloud offloading architectures are largely out of the users control and have higher latency than running the computation on the local device. Cloud computing architectures use specialised nodes, which need to be designed for the service provider. As more Internet of Things (IoT) devices are created and deployed, the number of idle devices also increases. These devices use power to stay in an idle state, which constitutes an inefficient usage of computational resources; both as network resources and physical materials used to make the device. To address an alternative to cloud offloading architectures, this work proposes an architecture for taking advantage of idle computing power of IoT devices for offloading computations from devices locally to improve privacy, latency, and sustainability. Testing of the proposed architecture demonstrates that computations can be successfully offloaded, with acceptable latency, and minimal increase to the individual power use of the connected IoT devices.

Keywords: Internet of Things · IoT · Offloading · Distribution

1 Introduction

Mobile devices often require aid in completing computations, due to stringent power and computation needs [21]. Offloading uses existing computational infrastructure to enhance the capability of resource-constrained (mobile) devices. Depending on connection speeds and data privacy, using cloud-situated resources for offloading is not always desirable or possible. Local offloading methods can be used to address these concerns. This requires an additional device for the offloading of computation, which is not always required for smaller offloading applications. By taking advantage of locally deployed devices, idle computation can be used to effectively complete offloading requests within a timely manner, and with no data exposure to the wider Internet.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 175–200, 2024.
https://doi.org/10.1007/978-3-031-63989-0_9

Within the wider area of IoT, many reduced functionality devices resort to offloading techniques and technologies to reduce local power consumption, or to circumvent local computation limitations. A common technique [31] of cloud offloading systems allows for devices to send computations to external services, and receive results given a short latency and run time. Whilst this works for applications where devices are connected to the wider internet, localised offloading techniques are still being explored for efficiencies in latency, reduction of communication complexity, and practical applications.

In addition to optimisation of offloading techniques for local applications, IoT devices are often produced with more computational power than strictly required for average operation [29]. In the context of Smart Home devices, many do not constantly use their full computational power, as their computation requirements vary largely. As the number of these devices increases, and become more complex [12], materials will become more scarce and difficult to manufacture. Use of local existing IoT devices would reduce the material cost of creating local offloading solutions.

Single device offloading systems can be a single point of failure for physical and digital vulnerabilities. Edge Computing [6] is the practice of keeping resources available to client devices as close to the client network as possible; often placing devices within the clients network for local access. Taking advantage of Edge Computing, a distributed architecture of devices to handle offloading on the edge would reduce latency, and increase privacy.

The aim of this work is to justify and propose an architecture for localised offloading of computations from reduced functionality devices to a network of low power devices. The contributions of this work are: (i) an architecture for dynamic offloading using pre-deployed IoT devices; (ii) a method for offloading with dynamic network sizes and computation sizes; (iii) an approach for negotiation of resources for multiple simultaneous clients; (iv) an evaluation of the architecture, focusing on scalability of devices, computation and offloading techniques.

The remainder of this paper is as follows: Sect. 2 looks at relevant past works. Section 3 presents the research design and methodology. Section 4 explores the areas of the proposed architecture to evaluate, and outlines the results of the experiment evaluation, to be further evaluated within Sect. 5, with Sect. 6 concluding this work.

2 Background

The Internet of Things (IoT) is an area of increasing importance, especially with more devices and technologies being used in technical, industrial, and private sectors. These devices can be used to perform simple tasks, and gather data from the world around them. IoT devices are becoming increasingly popular for consumers, and the rise of smart-technology enabled homes has lead to an increase in the usage of low power devices.

Additionally, as the number of devices globally increases [2], the longevity of device manufacturers and developers relies on the use and reuse of previously

or currently deployed devices. This would reduce overall production costs for new distributed systems, as well as increasing the material efficiency of in-place devices.

Distributed offloading systems cover a variety of different areas in existing literature. These areas include offloading technologies, automatic resource discovery, dynamic code deployment and execution, and resource negotiation.

2.1 Offloading Technologies

Offloading is an area that has been explored with increasing interest. Most of this focus is on the architectures and algorithms used when offloading computations. Mainly, methods that are used revolve around a Mobile Edge Computing (MEC) architecture, as explored in [16]. MEC architectures constitute an edge node communicating with the mobile client device, which can then communicate further with external cloud services. The edge node can communicate with other IoT devices, as well as cloud based servers. The main goal of this architecture is to offload from the mobile edge device, to reduce local power and computation requirements.

In the example presented in [11], applications for connecting multiple different mobile devices to the same network, This is a requirement for scalable dynamic MEC systems to be justified over the alternative of having one high powered device to complete otherwise offloaded computations. Additionally, [11] shows that such a system can complete complex tasks, such as Machine Learning algorithms, further justifying the use of a MEC system as a common place architecture.

[28] further looks at the joining of Small Cell Network (SCN) and MEC systems to allow for Non-Orthogonal Multiple Access (NOMA) algorithm to work on a similar architecture. In part, this is approaching the problem of bandwidth resource management; as more and more devices start using wireless communications, the usable range of network bands start to have a higher demand. Whilst this is a problem that is present for having a lot of devices working on the same SCN, the underlying architecture is focused on the application of MEC systems in the modern world. Whilst focusing on the bandwidth efficiency of this algorithm, the energy usage of the individual mobile device is taken into account within the bandwidth minimisation algorithm used.

In [25] and [14], new algorithms and architectures are proposed to solve highly specific problems. [25] explored the use of an edge device to compute energy harvesting techniques for IoT devices in a Fog architecture. This use of a edge device allows for IoT devices to interact with the network in a obfuscated manner: adding a layer of separation between the IoT device and the cloud system. This reduces the overall load on the cloud network, but as a trade-off, requires a MEC device in the end system. For this application, whilst each of the IoT devices exists in isolation, the application of a MEC device allows for computation to be done over the whole system, rather than on a single device. Similarly, [14] looked at computational offloading for IoT devices using a MEC

system. However, this work begins to look at decentralised distributed systems as a mechanism for Offloading.

As seen in the explored literature, offloading has been a major focus when using IoT devices for data collection and decision making. However, at this stage, mobile devices are always offloading to a high powered device, in the form of a MEC device, or cloud services. Distribution of computations to local devices, following distributed programming techniques, is not applied to MEC architectures to create offloading systems.

2.2 Resource Discovery

In IoT networks, having flexibility in the deployment and usage of resources is important to allow for up-scaling the network. Mobile devices specifically have started to interact more with the other devices around them, especially when approaching IoT integrated mobile systems. As discussed in [27], some challenges with using dynamic resource discovery are recognising the existence of new nodes, allowing for the sharing and acquisition of knowledge, and predicting when new nodes will join the network for scheduling optimisation. Each of these challenges is important to plan for when creating a dynamic node network, to ensure that the requirements of timely communication and QoS are met.

The usage of low power devices in a dynamic node network also presents challenges, especially the energy usage of the communication method and protocol used between new and old devices. Bluetooth Low Energy (BLE) can be a useful connection strategy for IoT devices to avoid using a WiFi based connection protocol. [7] proposed sDiscovery; a discovery technology for use on low power devices using BLE communications. This method more efficiently uses energy than traditional device discovery, mainly through the use of BLE as the communication method. This focused on the energy usage of the system, which is an important aspect when using deployed IoT devices. Additionally, sDiscovery used an optimisation to the search protocol, allowing for the overall system to discover more effectively and efficiently.

Another method of reducing the discovery time and the energy usage can be achieved by clustering devices into "neighbourhoods". [10] discussed how this method could be used to achieve fast discovery in Device-to-Device (D2D) architectures. This method would allow for dynamic sizing of smaller "neighbourhoods" of devices, promoting a scalable device discovery architecture. Additionally, use of a decentralised system promotes the use of low power devices, without the requirement of a high power device to act as a routing node. However, testing is deployed using a simulation, without physically creating a network of low power devices.

Literature focusing on device discovery outlines the requirements of an IoT network discovery system; mainly the energy efficiency of the system, and the speed of the discovery and subsequent connection. Most work focuses on simulation results, rather than physical implementation with low power devices.

2.3 Dynamic Execution and Deployment

Distributed programming techniques are another method of Offloading computations from a Host device. By distributing computations out to multiple different devices, local energy can be conserved. QoS and Latency are still issues with this type of programming technique, however depending on the architecture and method, these can also be minimised. One approach that aides in distributed programming techniques is dynamic deployment. By dynamically allocating operations to individual devices, operations can be completed in a distributed manner whilst using each device effectively.

As proposed by [5], dynamic deployment of code using containers can optimise the usage of resources to complete a computation. In this case, Docker was used as a containerisation technology for a virtual cluster. Whilst this type of deployment can help to limit the run-time of certain operations, it does also require a higher level of computation to run multiple virtual machines in a cluster. Additionally, it further discusses creating sub-clusters for further breaking down operations into workable components.

Further, the concept of elastic clusters can be used, allowing devices to be dynamically allocated tasks, rather than either being in the cluster, or not in the cluster. As [8] proposed, containerisation of Virtual Elastic Clusters (VEC) can be used as a management system for virtual machines, in replacement of a Hypervisor. This use of elastic clusters focuses on reducing the overhead for individual virtual devices, allowing for more efficient VEC's to be used.

To demonstrate the applications of elastic clusters, [15] discusses using elastic clusters to coordinate wireless sensor networks using a Fireworks algorithm. This application of a distributed algorithm allows for stages of the algorithm to be completed with a faster convergence rate. However, the algorithm is still being deployed using VEC systems.

To further optimise elastic clusters, [20] proposed dynamic deployment of components for embedded systems. By outlining the existing components of a cluster, the components can be migrated to physical embedded devices. This allows for less computation on a host machine, and for embedded systems to do some calculations locally rather than only focusing computational power on data collection. This begins to approach the focus of this paper, as allowing for more computations to take place on low power devices would reduce the computational requirements for client devices.

At this stage, some literature is focused on the clustering and dynamic deployment of virtual machines. This still requires a higher powered network of devices to run the virtual machines. By using running computations on Idle low power devices, the computational requirement of all devices in the cluster can be reduced. QoS can be maintained through different distributed programming techniques.

2.4 Resource Negotiation

In addition to offloading, technologies for prioritising computations need to ensure that operation run-time and QoS meet given requirements through the

process of offloading. Device negotiation approaches this problem by having shared agreements between the host (handling offloading resources) and client (generating offloading requests) devices. As [30] discussed, using a fog architecture to complete offloading requires low latency for operations such as augmented reality. To aid in this type of computation, edge devices may need incentives to complete the computation. As a result of the required motivation and low latency requirements, fast negotiation practices are required to allow for distributed offloading using the Fog network. Whilst this focus works well for decentralised High Powered Devices, for low powered devices such a system may have too much overhead, as incentive systems and negotiation practices require more communication and processing power.

Mesh networks, a type of fog architecture, can be used for individual Device-to-Device (D2D) negotiations. [17] took advantage of individual D2D negotiations for implementation of an energy monitoring and management system for an IoT network. This type of network allows for devices to reach an agreement based on their own computational load and specifications. A distributed network of IoT devices can make an agreement as to which devices will complete tasks required of the network. However, this method does take advantage of centralised task creation method. Having negotiations take place between individual devices, whilst a task list is created in a centralised manner leads to a semi-centralised system, requiring a control hub in addition to the smaller Low Powered Devices.

Further, QoS requirements and capability may not always match between the host network and the client device. As presented by [26], QoS can be negotiated between an end system and the offloading device. In the presented example [26], a simulation is used to model the communication between a mobile device with a IoT device. However, with a network of distributed devices, the overall specifications of the network may not be easily queried, and would need to be recalculated every time a device leaves or joins the network. This may not be suited for dynamic distributed systems, especially for low power devices.

2.5 Gap Analysis

This paper proposes an architecture to use idle low power devices for offloading distributed computations within a deployed IoT network. Exploration of offloading processes onto low power devices is a new area of research. Dynamic programming and deployment strategies can be used to aid in the creation of this architecture. The QoS, latency, run-time, energy usage, and materials usage are all concerns that must be taken into consideration when designing and testing the proposed architecture.

As can be seen from Table 1, the explored works do not cover a system which uses all areas of investigation. The aim of this paper is to outline how such a system would be designed and implemented.

The state-of-the-art analysis identifies the following areas that should be further investigated:

Table 1. Areas covered by existing literature

Literature	Sustainablity	Discovery	Offloading	Negotiation	Dynamic Deployment
[9]	✓		✓		
[10]		✓		✓	✓
[25]	✓		✓	✓	
[17]				✓	✓
[8]					✓
[1]	✓		✓		
[23]			✓		
[18]	✓		✓		
[24]			✓		✓
[19]	✓	✓			
[22]			✓	✓	✓

- Usage of idle computational resources to complete complex computations.
- Sustainability (material cost, energy use) of existing devices within a network.
- Offloading of computations onto a network of low power devices.
- Resource negotiation within offloading computations onto a network.
- Dynamic deployment of offloading computation.
- Low power devices with service discovery.

An IoT system that incorporates resource discovery, offloading of computations, and resource negotiations with dynamic deployment have not been identified or closely studied. Additionally, the area of sustainability within IoT networks and devices has not been thoroughly investigated, specifically with managing old or overpowered devices.

3 Localised Offloading Architecture for Distributed Horizontal Computations (LOADHoC)

The Localised Offloading Architecture for Distributed Horizontal Computations (LOADHoC) is the proposed solution for distributing computations among low power devices. This section aims to identify the core aims and requirements for the development of a prototype using the LOADHoC.

3.1 System Aims

To meet the aims of this work, an architecture is proposed for enabling the use of previously deployed devices for computing local offloading requests. The following aims have been defined to guide evaluations of localised distributed offloading with LOADHoC using the areas identified within Sect. 2.

- Aim 1: Creation of a dynamic system for decentralised offloading of computation.
  Using device discovery allows for ad hoc networks to be created per computation, meaning that devices can be added and removed from the network dynamically. Further, an important aspect of this architecture is to ensure that the latency and QoS are comparable to other offloading solutions.
- Aim 2: Designed for using low power devices for offloading.
  Understanding how networks of low power devices can be used for offloading is important for understanding whether such a system is achievable and useful.
- Aim 3: The system should be energy and resource efficient. Understanding the resources used for the architecture is important for identifying the sustainability and viability of the proposed architecture. This would involve analysing the energy use, latency, and physical material use.

3.2 System Requirements

Given the preceding System Aims, the final prototype design must adhere to the following functional requirements:

- Req 1: The proposed system must use low powered devices.
  Most offloading systems [13] currently use higher power devices. The aim of this paper is to take advantage of pre-existing low power devices within a network. As such, the prototype must use similar devices to emulate the practical application of the research.
- Req 2: The system must be able to perform offload-able computations.
  Offloading computations onto Low Powered devices allows for the prototype to be used to be useful manner, aligning with the aim of reuse of low power devices, and is key to meeting Aim 1.
- Req 3: The system must be decentralised for system requests and queries.
  In a practical situation, devices may be added or removed from the system, with the elimination of high power devices within the offloading process. As a result, the system must be decentralised to allow for each device to be removed from the network when needed.
- Req 4: The system must have measurable latency and QoS.
  Measuring the latency and QoS allows for the comparison of the proposed offloading architecture against performing the computation locally, and offloading to a cloud system.
- Req 5: The energy usage of the system must be able to be monitored, when completing operations.
  The energy usage must be monitored to ensure that the overall system is sustainable not only in terms of computational output per material used, but also in regard to energy usage.

Further, some non-functional requirements must be observed during the development of the LOADHoC.

- Req 6: The offloading computations must have comparable run-times to other offloading solutions.
  The run-time for offloading computations is important as the typical use of offloading is in real-time application. As a result, if the offloading is must slower, than the use of offloading on low power devices is not viable in a commercial application.
- Req 7: The devices must be able to dynamically disconnect and reconnect to the network.
  The system is designed for the use of pre-existing devices. However, when new devices join the network, or old devices leave the network, the system should be robust so that users do not need to manually add or remove devices from the offloading system.
- Req 8: The devices can be queried to display their computation.
  For diagnostic purposes, devices should be able to make available their current computation for testing. In a real world application, this decreases the security of the network, but making clear what computation is being completed allows for more careful and informative analysis of how effective the computation is, as well as analysis of the viability of the proposed system.

3.3 System Design

To meet these requirements, the LOADHoC is designed as a horizontally scaling system, with dynamic resource discovery. As can be seen in Fig. 1, each of the devices in the proposed network are of equal importance; with a distinction only being drawn when a client (the device requiring offloading) connects to the system. By having one of the devices act as a leader (the device handling the clients computation), this means that all of the devices in the network can receive instruction, and allows for no distribution of computation on the client device.

Each of the devices within the network requires two processes to function: these processes are the discovery and advertisement process, and the computation and request process.

The discovery and advertisement process takes advantage of mDNS addresses to advertise the device as a serviceable device. As can be seen in Fig. 2, this allows for other devices in the network to identify this device within the LOADHoC system. The process is then able to discover all other devices within the system, to keep track for when the new device is considered a Leader device.

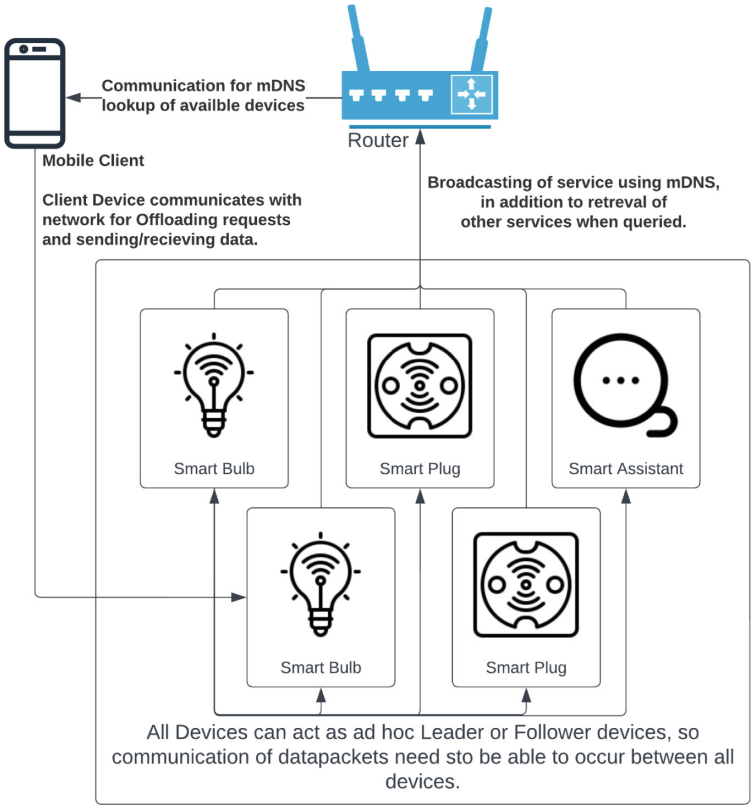


Fig. 1. Architecture Design of the LOADHoC

The computation and request process handles all requests for computation or information from the device, in addition to completing the computations that it receives. Importantly, this process must be able to act both as a leader or a follower device, as computations received directly from client devices will need to be split among all of the follower devices in the system, including the leader device.

The information from each individual device must be able to be queried by the system, as this allows for the leader device to check if a device is able to receive computation, as well as gain detailed information about the rest of the network that can then be used for allocation of computations. This, in addition to the previously outlined computations and dual functionality of the devices, requires the following methods must be present on all of the devices within the architecture:

– Info Request: Request the information of all of the devices within the network.

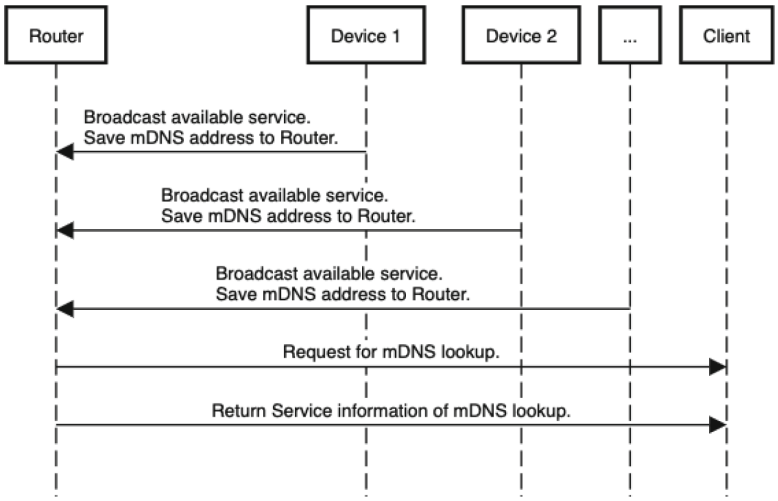


Fig. 2. Sequence for device advertisement method

This request would come from a Client device, and must be propagated through the network using the Info method.

- Info: Sends the info of the local device to the requesting device.
  This method is a direct response to the Info Request method. This endpoint will only be used by leader devices in the network requesting the device information.
- Ready Request: Request the ready status of all of the follower devices in the network.
  This method is similar to the Info Request method, and must return the current status of all of the follower devices in the network. This method would be requested by client device interacting with the system, to be further propagated through the remainder of the network.
- Ready: Sends the status of the local device to the requesting device.
  This method is a direct response to the Ready Request method. This endpoint should only be called by the leader devices, to compile information on the status of all devices for the Ready Request method.
- Computation Request: Receives the requested computation from the client device.
  This method is required on all devices as all devices must be able to act as a leader: able to accept computations from potential clients. This method focuses on filtering the type of computation requested, in addition to allocation of computation to other devices within the network.
- Computation: Receives computation from the leader device, completes and returns the result to the leader device.
  This endpoint will only be requested by the leader device in the context of the computation. This method receives portions of the overall computation, and returns the processed values.

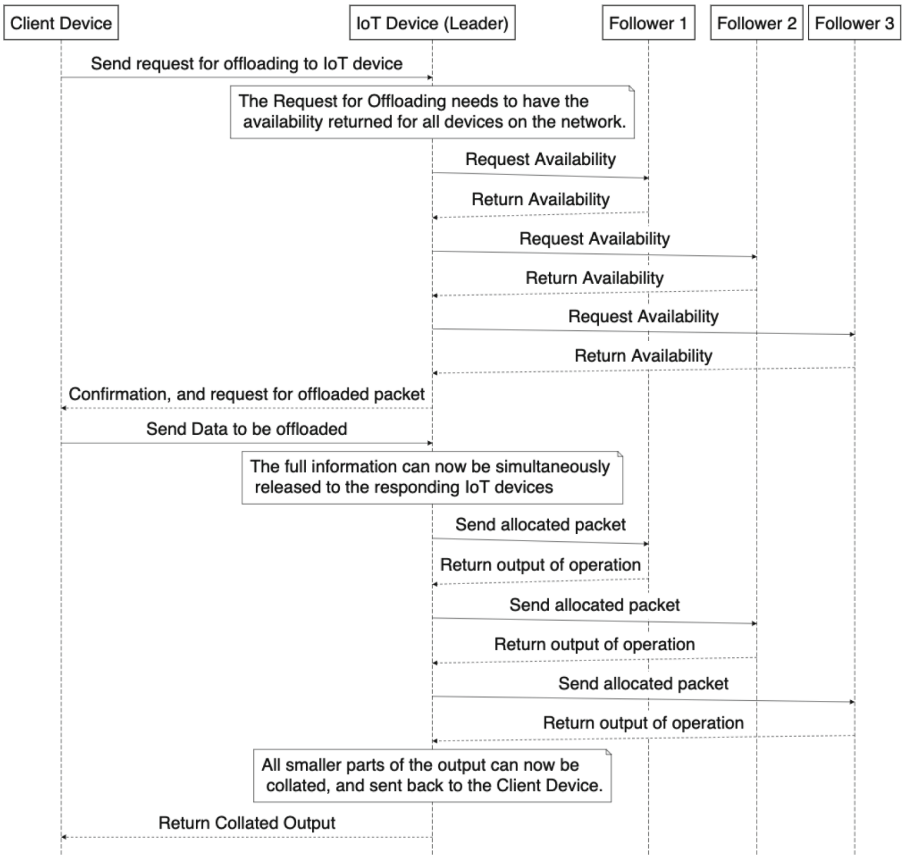


Fig. 3. Sequence Diagram for checking network status and completing computation.

The sequence of events for the status request system and computation request system are shown in Fig. 3. The status of the network, such as number of devices and processing ability of the devices, does not need to be retrieved before sending a request. The leader device should not wait for a portion of the computation to be completed before requesting another portion. This happens asynchronously, to allow for multiple devices to be working on computations simultaneously.

4 Experimental Evaluation

4.1 Experimental Setup

The experimental setup for all of the experiments use the LOADHoC prototype. The LOADHoC prototype is developed using a Restful API, taking advantage of Wireless Access Points (WAP) pre-existing on smart home devices. The use of devices already in place demonstrates the ability for the LOADHoC to operate

without any additional components being added to the network. Smart home systems contain many of these types of devices, and would not require a dedicated device for functional offloading. Communication between the individual devices is conducted through API requests, with each individual device operating as an API.

The LOADHoC prototype was deployed to a cluster of 9 Raspberry Pi Zero Wireless devices (shown in Fig. 4) in addition to a Raspberry Pi 3b+. The Zero devices utilises a 1 GHz Broadcom BCM2835 single-core processor, with 512MB of RAM. Each board communicates with the rest of the network wirelessly using a Mediatek RT5370 (2.4 GHz B/G/N) chipset. The Zero boards run a Raspbian Lite OS, and the 3B+ a full Raspbian OS installation. The singular Raspberry Pi 3b+ includes a Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC processor with a 1.4 GHz clockspeed. This board additionally contains a wireless Cypress CYW43455 chipset. All devices were loaded with the heterogenous LOADHoC prototype, operating as leader and follower devices depending on which device receives the clients initial request.

When a client is interacting with the LOADHoCcluster, each device is able to operate as a routing device (splitting and sending computations to others within the system), and a worker device (completing requested computations). This means that if a request is sent to any of the devices, the computation will be split between the existing devices within the architecture.

Further, each of the devices within the LOADHoC is discovered automatically, with new devices able to participate in the load sharing protocol. This discovery is achieved using ZeroConf [3], an implementation of mDNS addresses designed for service and resource discovery. Devices are able to join the network at any time.

The system can support any offloaded computation, with testing occurring with matrix multiplication problems as the main measure, and accuracy of π also able to be completed with this implementation. Given the design, the system will be able to complete any offloaded computation given enough computational resources across all devices.

4.2 Experiment 1: System Validation

This experiment aims to assess the effectiveness of the proposed architecture. To analyse the effectiveness, the latency, run time, and QoS will be measured, and tested against non-distributed offloading, in addition to non-offloaded computation. This will be completed using the API implementation, to test the validity of the proposed architecture.

The setup for this experiment is focusing on the architecture, rather than the deployment to low power devices. In response to Aim 1, the latency, run time, and QoS will be individually measured.

The LOADHoC can be tested easily by deploying the API to three external devices, and creating a client to test the required metrics. The expected results from this experiment are that the run-time and latency are lowest when performing the computation locally, as the client device will be a high powered

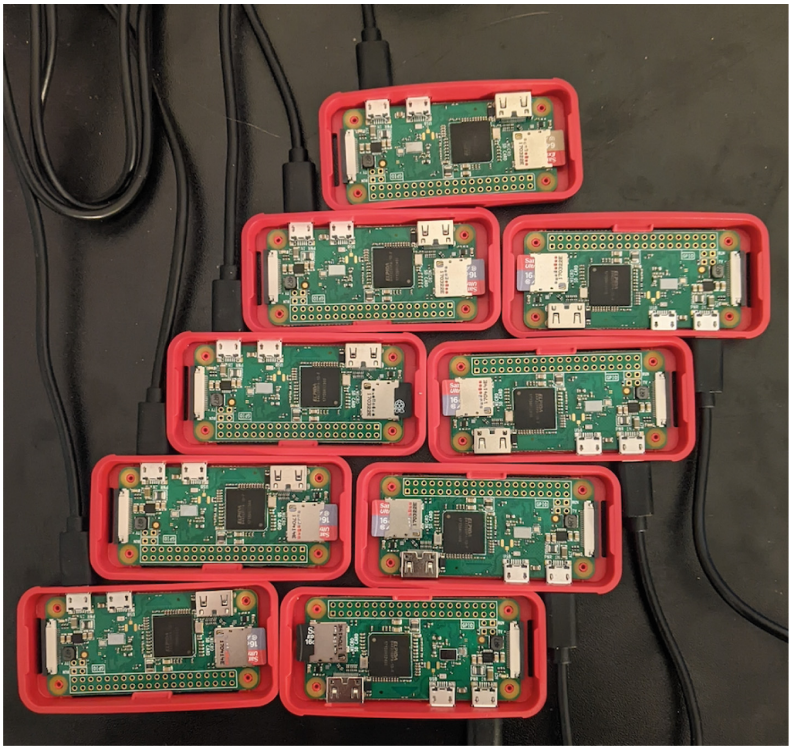


Fig. 4. 9 Raspberry Pi Zero W devices loaded with the LOADHoC Prototype

device. The non-distributed offloading and the distributed offloading systems are expected to be comparable in terms of latency and run-time, as the run-time and latency increase through the use of multiple devices will balance the reduction of sending the computation to multiple devices. However, as the number of devices increases, this balance will change, and the distributed offloading system is expected to outperform the non-distributed offloading method. Finally, the QoS should be the same among all three test-beds, which can be measured by testing the result matrix from each system against the other systems.

Figure 5 shows the time for each system to complete a matrix multiplication problem of the listed size. The tested methods are completion of the computation on the client device (Time locally), offloaded to one device (Time offloaded), and the time offloaded to the LOADHoC (Time offloaded with Distribution). The lowest run-time was achieved by completing the computation on the high-powered client device. This is expected, however this time may differ for mobile clients. Offloading the computation onto one local device produced acceptable run-times, but increases geometrically given the size of the matrices. This time may decrease or increase given the computational resources available to the single offloading device. The distributed system performed similarly to the single offloading system, demonstrating its viability when compared to traditional offloading systems.

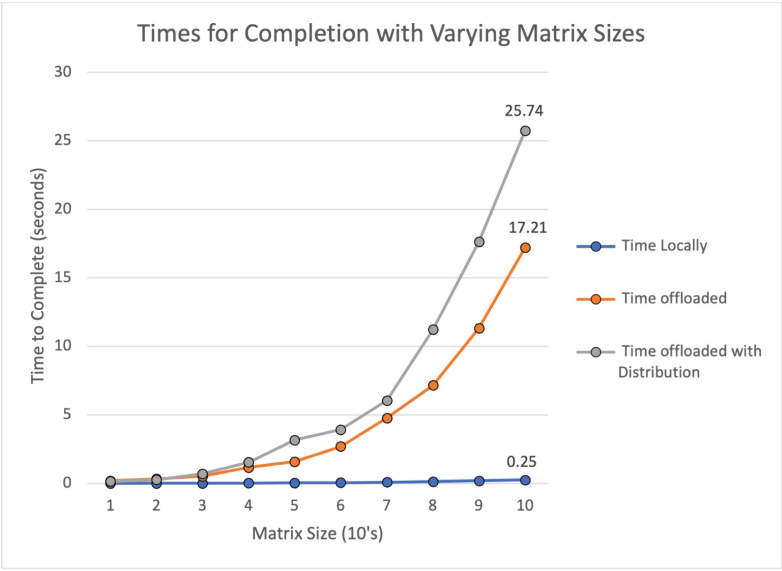


Fig. 5. Time-to-Complete for tested methods

Originally, the response time was much higher due to an implementation error. Each row of the second matrix was sent to devices using individual web requests, meaning the delay for each row was compounded. This inefficiency was fixed by removing all logging of data to the console, as well as changing the method to send multiple columns in the same request.

Additionally, it was shown that the distributed and non-distributed systems were comparable as the size of the matrices increased, with the distributed method taking longer. Experiment 2 tests the sustained efficiency of the system as more devices are added to the network. The non-distributed method performed better as the size of the matrices increased, however as the number of devices is increased, the response time should decrease for larger matrices.

When running computations, the individual power draw for one of the nodes in the LOADHoC can be measured, as shown in Fig. 6. This allows measuring the instantaneous power draw per device, which can give an energy usage rating to all of the devices in the network. As shown, the power draw does change when computations are being completed, by a constant amount. This method can also be used to visualise the computations being completed by the individual device.

Further, all three methods returned the same result matrix when presented with equivalent inputs for all tests. As the LOADHoC is producing an accurate result Matrix, the QoS has been preserved through the use of the API, with matrix sizes up to 100×100.

4.3 Experiment 2: System Scalability

This experiment aims to test the scalability of the LOADHoC. To analyse the architecture, the latency, run time, energy usage, and QoS is be measured for

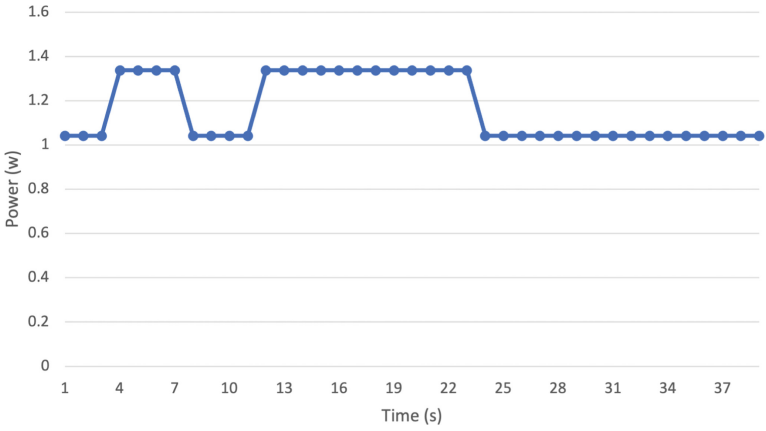


Fig. 6. Power usage of one Raspberry Pi within the system

computations that use up to 9 devices. These computations are distributed through the LOADHoC, using a standard 100 by 100 matrix.

The results of this experiment outline the scalability of the LOADHoC prototype for small scale calculations on medium size networks. The LOADHoC implementation can handle a large number of devices within the network, limited only by the unique number of IP addresses on the network. Testing the run-time of requests within the system as the number of devices increases allows for understanding the minimum number of devices required to complete computations faster than other traditional offloading and local calculation methods.

The LOADHoC prototype has been developed to allow for the number of devices within the system to dynamically change. For this experiment, 9 devices, including the Leader device, are connected within the system. In testing the LOADHoC, two 100×100 matrices are multiplied together, as requested by the client device.

The latency and QoS are measured by the Client device, using the amount of time that it takes to get the result from sending the computation. Additionally, the energy usage of the overall system will be monitored, using one energy collection device to measure the usage of the entire system, and one to measure the usage of an individual device.

This experiment allows the system to be tested on larger networks; testing its run-time and energy usage, compared against performing the computation on a smaller number of devices. Additionally, the method that the LOADHoC uses to handle distributed computations can also be evaluated.

Figure 7 details the time-to-complete the multiplication of two 80×80 matrices. As can be seen, this time increases as the number of devices increases. The lowest run time occurs with one device, as communication only needs to occur between the client and leader device. As the number of devices increases, the run-time also increases due to the number of computation requests that must be sent to each of the follower devices.

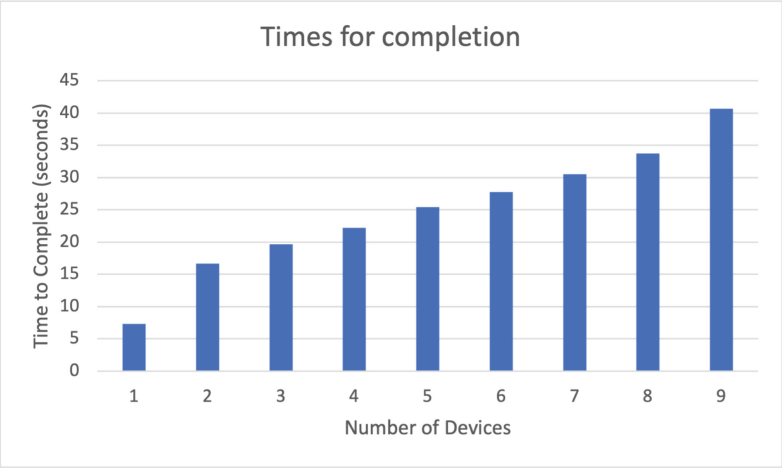


Fig. 7. Time to complete with given number of devices

Figure 8 displays the energy usage of the network when offloading the multiplication of two 100×100 matrices across 9 devices. As more devices receive their computational load, the energy demand of the system increases to accommodate the extra processing power required by the worker devices. The baseline energy usage of these devices remains unchanged before being added to the LOADHoC. This means only the increase of energy use during a computation must be considered when analysing the energy usage of the system.

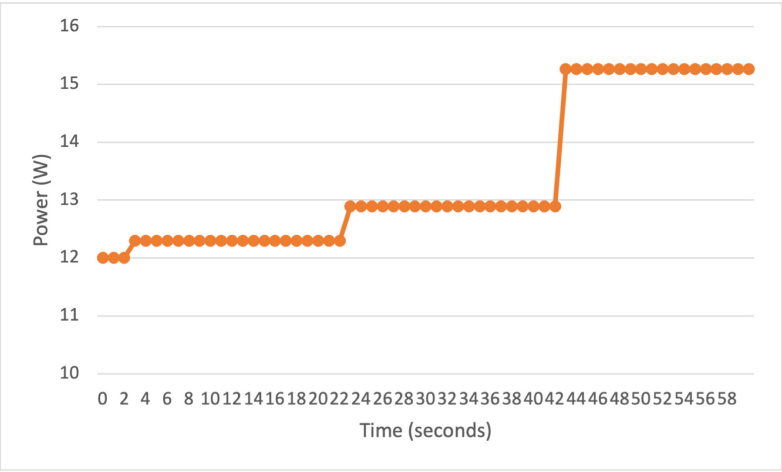


Fig. 8. Energy for 100×100 multiplication across 9 devices

In this case, offloading to one device is seen to be the most optimal solution for completion time and energy use. However, the LOADHoC implementation is currently implemented on high power devices, and as such, are able to complete more complex operations by themselves. These devices are able to complete computations effectively by themselves. The main time-sink is the sending of messages between the leader and follower devices.

4.4 Experiment 3: Computation Flexibility

The aim of this experiment is to demonstrate and test the system using another distributed computation. Implementing a second computation algorithm demonstrates the suitability of the LOADHoCfor dynamic deployment methods. This flexibility in the computations that can be completed is integral for the system having wider applications.

For this experiment, a estimation of π will be implemented as a distributed computation across the system. The accuracy will be tested to ensure that the result that is achieved by the system matches a locally calculated value.

The LOADHoC system has been enhanced to include the calculation of Pi. In order to implement the π calculation, the system must have the distributed computation method on all of the devices. To demonstrate the LOADHoC's flexibility, the overall system architecture needed to remain functional, as well as maintaining the matrix multiplication functionality.

The π estimation method used within this experiment is the Nilakantha infinite series [4]. The Nilakantha series uses the following equation:

$$\pi \approx 3 + 4 \sum_{n=0}^{\infty} \frac{-1^n}{(2n+2)(2n+3)(2n+4)} \tag{1}$$

As implemented within the prototype, the maximum value of n is the accuracy of the computation, which can be specified by the client device when retrieving the value. The values of n can then be distributed to each of the devices, the calculation of each section distributed to the follower devices, then added together on the leader device.

Listing 1.1. Code for estimating Pi locally

```
function calcPi(accuracy):
    result = 3
    for( i in range(0, accuracy / 2)):
        n = i * 2
        m = n + 1
        result+= (1/((2n+2)(2n+3)(2n+4))) - (1/((2m+2)(2m
            +3)(2m+4)))
    return result
```

Listing 1.1 highlights how this code can be used for estimating π on a local device. By giving individual devices differing values of i, the function can be distributed across the for-loop.

This experiment outlines the ability for the LOADHoC to be adapted for more distributed programming applications. The adaptation the prototype requires identification of the requested computation, distribution and collection of the computation, and calculation on the follower devices.

The testing value of π for this experiment is as follows:

$$\pi \approx 3.14159265358979323846264338327 \tag{2}$$

Table 2. Values retrieved from distributed calculation of π

Value of n	Value of π Obtained	No. of accurate decimals
$n = 10$	3.222821622821623	0
$n = 100$	3.141620854093511	3
$n = 1000$	3.1415927233510117	6
$n = 10000$	3.141592653593063	10
$n = 100000$	3.1415926535897905	14
$n = 1000000$	3.1415926535897932	16

The values obtained range in accuracy given the depth of the π calculation. The values within Table 2 were retrieved by requests to the LOADHoC implementation. Further levels of accuracy can be obtained given larger values of n. However, a limitation of the current system is the implementation, and the number of decimal places available. The LOADHoC implementation is currently developed in JavaScript, and limits the number of decimal places available to be received by the client. This is largely a problem that exists with this calculation and this implementation, so this does not effect the validity of the overall LOADHoC implementation.

This experiment demonstrates the flexibility of the LOADHoC architecture. The prototype is able to handle different distributed computations, which are only limited by the systems programmed on the networked devices.

4.5 Experiment 4: Resource Allocation

This experiment aims to demonstrate the implementation of different types of allocation methods for resources. Given the different calculations achievable, and differing devices in the architecture, systems for allocation different sized loads to different devices.

This experiment will record the timing of each of the different allocation methods (including calculation), as well as the energy use of the system using these methods. The output of the different allocation methods can also be tested, to ensure the QoS of the overall system is maintained with differing resource allocation methods.

The previously outlined LOADHoC prototype is used for this experiment. In addition to the calculations, a resource allocation method is implemented for allocating the resources between those available in the network. Within the LOADHoC implementation, four allocation methods are present. These allocation methods are:

- "none": Default allocation method, which splits the computation over the requested number of devices.
- "even": Allocates the computation to every second node, relative to their position in the leader device list.
- "ready": Similar to the default allocation method, allocates computation only to ready devices. Devices are considered ready if they are not currently completing another operation.
- "geo": Allocates multiple computation regions to the same device; increasing by one portion of the overall computation as each device is allocated its computations.

Each of these methods must be set within the leader device local data structure. Once set, all calculations using the altered device as leader will use the specified allocation method. Importantly, different methods can be defined for differing leader devices. This means multiple users are able to define different allocation methods while using the system simultaneously.

This experiment demonstrates that allocation methods can be created and used for the LOADHoC architecture. This implementation uses four different methods, while showing the effective run-time of the same calculation across all methods. Further, more complex allocation methods can be used effectively with this system, using the data made available by each of the devices.

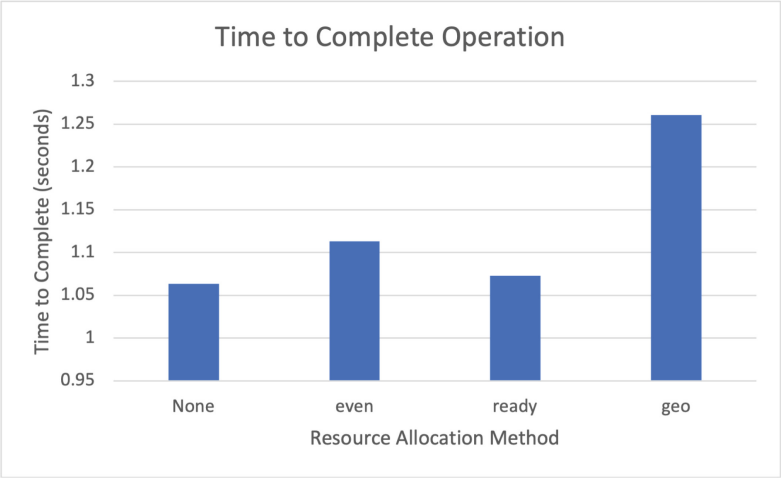


Fig. 9. Time to complete with different allocation methods

As can be seen in Fig. 9, the time to complete the different allocation methods was comparable between all methods, other than the "geo" method. This is mainly due to the number of requests being sent to the same device. Increasing the load on an individual device, when all of the devices are of equal computing and processing power leads to an increase in the total completion time. The Geo method was unable to resolve as the request was taking too long to be fulfilled. This is because the device that was loaded with the most computation was similar in power to the remainder of the devices, and was required to compute many different parts of the computation simultaneously.

This experiment outlines that different allocation methods are possible, in addition to querying information from the devices to inform this allocation. The LOADHoC architecture is able to take advantage of more advanced allocation methods, and only requires the leader device to manage the allocation of computation within the network.

4.6 Experiment 5: Robustness

The aim of this experiment is to evaluate the failure recovery methods for the LOADHoC architecture. The LOADHoC implementation is able to freely add, remove, and drop devices. This allows individual devices to cease normal operation, whilst maintaining the integrity of the overall architecture and network.

The experiment records the QoS of the given computation after a device is forcefully removed from the network. By simulating a device being removed from the system in the middle of a requested computation, the way that the LOADHoC architecture can handle errors can be observed.

This experiment uses the previously outlined LOADHoC implementation. Each of the nodes in the network uses two separate processes: advertisement and discovery, and a computation process. By shutting down the computation process of an individual device, a connection or computation error can be simulated. Importantly, this simulates when a device that is actively on the network is not responding in an expected manner.

When sending a request to a new device, if the request fails, then the same request is sent to a new device listed on the Leader devices device list.

Testing the system in this way tests the LOADHoC's ability to support recovery and retention of computation when individual devices fail. As the client only communicates with the Leader device, the follower devices are able to drop in and out of the network without effecting the ability for the system to function.

Multiple devices can be removed from the network. This will effect the overall run-time to comparable times to limiting the number of devices a computation can use. This test will look at the QoS of the LOADHoC implementation when devices are removed.

This experiment focuses more on how the architecture can be implemented. By only relying on the Leader device to function, a system running the LOADHoC architecture can be scaled horizontally without negatively effecting the

network. Additionally, parts of the network can be removed without effecting the integrity of the remaining network.

Overall, as devices were forcefully removed from the system, the LOADHoC implementation was able to cope with the removal, and reallocated the specific portion of the computation to an active device. This was done by querying all of the devices in the leaders device list, which includes itself. However, this did effect the total run-time of the system, in a similar manner to the results of the experiments presented in Subsect. 4.3. However, when removing the leader device in the middle of a computation, the result will not be returned to the client. The leader device is determined by which device receives the request from the client device. As such, other devices in the network can be queried, so re-sending the computation request from the client side will result in the computation being completed, maintaining the system fidelity. This removal does not effect other computations using the removed device as a worker.

These results demonstrate that the overall architecture is able to cope with individual devices being removed from the network. In applications where network security and stability are not ideal, the system will remain to function in an optimal condition given the client remains connected to the leader device.

5 Discussion

Addressing the aims of the LOADHoC implementation, as defined in Subsect. 3.1, the experimental results can be split into three areas.

5.1 Creation of a Dynamic System for Decentralised Offloading

Through the development of the LOADHoC, the requirements outlined in Subsect. 3.2 were used to allow the overall design to achieve dynamic centralised offloading. Subsection 4.2 outlined the use of the system using matrix multiplication as a test problem. In addition, Subsect. 4.4 focuses on a calculation of π, highlighting the ability for different computations to take place using the system.

By taking advantage of multiple low powered devices, many different types of computations can be completed, limited only by the time constraints placed on the computation. Given the results achieved in Subsect. 4.2, computation runtime could feasibly fall outside the request timeout set by the client device. This can be tweaked, by changing the method of request delivery and sending. Use of a message queue would allow for dynamic Time-To-Live (TTL) for requests and results. The LOADHoC is able to cope with multiple different types of resource management, as demonstrated within Subsect. 4.5. The LOADHoC is designed to be dynamic in both computation and resource allocation method, allowing for further implementation in real-world dynamic applications where local offloading is required.

5.2 Design for Implementation on Low Power Devices

The overall design of the LOADHoC allows for use of low power devices to run and work on the network. This was achieved using a RESTful API system as a demonstration, allowing definition of methods and endpoints, as outlined in Subsect. 3.3. Whilst experiments were run on high powered devices, the network connectivity was the main difference in run-time when compared to a traditional offloading system. As the LOADHoC needed to communicate with several devices, offloading to a dedicated device was substantially faster, as seen in Subsect. 4.2. This time difference was also outlined in Subsect. 4.3, as use of 9 devices produced the largest run-time. In addition to time delay through communication within the network, the use of high powered devices splitting processing power among multiple passive processes can also reduce the amount of available resources for computation. This is highlighted by Subsect. 4.2, where even offloading to a single high powered device significantly increased the run-time of the computation. Distribution of the computation across the network decreased this time inefficiency, while still taking longer to complete the computation. Low power devices can also be easily fault or disconnect from networks in real-world applications. As seen in Subsect. 4.6, the LOADHoC can cope with device reliability issues, allowing for QoS to be maintained for error-sensitive computations. The LOADHoC design lends itself to use on low powered devices, through simple implementation design and robust handling of device and communication errors.

5.3 Energy and Resource Usage and Efficiency

The LOADHoC implementation was tested on run-time, as well as energy and resource usage. The main outcome from the implementation is that the architecture would work on low power devices, as discussed in Subsect. 5.2. Additionally, it can be seen in Subsect. 4.2 that the energy usage of the system is comparable to having the devices passively in the network. Given the implementation of the LOADHoC onto pre-existing devices, the overall energy and resource consumption is minimal compared to having a dedicated device for computation. The energy from each of the devices does add up, as can be seen in Subsect. 4.3, but still remains less than the overall passive power of the devices within the network. The current power readings from the network are a reflection of the power requirements for an implementation of the LOADHoC using low power devices. However, the values may differ, effecting the overall efficiency of the networks energy use. When analysing resource use, taking advantage of pre-existing devices ultimately saves on cost, in addition to resource and material use.

6 Conclusion and Future Work

Through this work, horizontal distribution of computation has proven to be a viable method of distributing computation. Whilst the overall time constraints

and energy usage are higher than the use of a single high powered device, the computations being run on each individual device can be implemented on lower power devices for in-place computation; without the addition of new devices into a network. This decreases the overall price of using a localised offloading network, at the cost of time-to-complete. Whilst the energy efficiency is lower than using a single high powered device, as discussed in Sect. 5, the increased energy usage is minimal when compared to a new high powered device on the network.

Future work focuses on the implementation of the LOADHoC onto low powered devices, testing the run-time and resource usage to determine viability in real-world resource constrained networks. In addition, different methods of communication can be tested, such as a message queue system.

References

1. Agarwal, P., Kumar, G.V.V.R., Agarwal, P.: IoT based framework for smart campus: COVID-19 readiness. In: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 539–542 (2020). https://doi.org/10.1109/WorldS450073.2020.9210382
2. Benazzouz, Y., Munilla, C., Gúnalp, O., Gallissot, M., Gúrgen, L.: Sharing user IoT devices in the cloud. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 373–374 (2014). https://doi.org/10.1109/WF-IoT.2014.6803193
3. Bohnenkamp, H., van der Stok, P., Hermanns, H., Vaandrager, F.: Cost-optimization of the ipv4 zeroconf protocol. In: 2003 International Conference on Dependable Systems and Networks, 2003. Proceedings, pp. 531–540 (2003). https://doi.org/10.1109/DSN.2003.1209963
4. Brink, D.: Nilakantha's accelerated series for pi. Acta Arithmetica **171**, 293–308 (2015). https://doi.org/10.4064/aa171-4-1
5. Cao, Y., Wang, H., He, S.: Dynamic deployment model for large-scale compute-intensive clusters. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 586–591 (2020). https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162887
6. Caprolu, M., Di Pietro, R., Lombardi, F., Raponi, S.: Edge computing perspectives: architectures, technologies, and open security issues. In: 2019 IEEE International Conference on Edge Computing (EDGE), pp. 116–123 (2019). https://doi.org/10.1109/EDGE.2019.00035
7. Chen, B.R., Cheng, S.M., Lin, J.J.: Energy-efficient BLE device discovery for internet of things. In: 2017 Fifth International Symposium on Computing and Networking (CANDAR), pp. 75–79 (2017). https://doi.org/10.1109/CANDAR.2017.95
8. de Alfonso, C., Calatrava, A., Moltó, G.: Container-based virtual elastic clusters. J. Syst. Softw. **127**, 1–11 (2017). https://doi.org/10.1016/j.jss.2017.01.007, https://www.sciencedirect.com/science/article/pii/S0164121217300146
9. Hou, W., Li, W., Guo, L., Sun, Y., Cai, X.: Recycling edge devices in sustainable internet of things networks. IEEE Internet Things J. **4**(5), 1696–1706 (2017). https://doi.org/10.1109/JIOT.2017.2727098
10. Huang, P.K., Qi, E., Park, M., Stephens, A.: Energy efficient and scalable device-to-device discovery protocol with fast discovery. In: 2013 IEEE International Workshop of Internet-of-Things Networking and Control (IoT-NC), pp. 1–9 (2013). https://doi.org/10.1109/IoT-NC.2013.6694047

11. Huang, X., Leng, S., Maharjan, S., Zhang, Y.: Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks. IEEE Trans. Veh. Technol. **70**(9), 9282–9293 (2021). https://doi.org/10.1109/TVT.2021.3096928
12. Janeczek, K.: Composite materials for printed electronics in internet of things applications. Bull. Mater. Sci. **43**(1) (2020). https://doi.org/10.1007/s12034-020-02101-x
13. Jha, S.C., Gupta, M., Koc, A.T., Vannithamby, R.: On the impact of small cell discovery mechanisms on device power consumption over LTE networks. In: 2013 First International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 116–120 (2013). https://doi.org/10.1109/BlackSeaCom.2013.6623393
14. Lei, L., Xu, H., Xiong, X., Zheng, K., Xiang, W.: Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system. IEEE Internet Things J. **6**(3), 5345–5362 (2019). https://doi.org/10.1109/JIOT.2019.2900550
15. Liu, X., Zhang, X., Zhu, Q.: Enhanced fireworks algorithm for dynamic deployment of wireless sensor networks. In: 2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST), pp. 161–165 (2017). https://doi.org/10.1109/ICFST.2017.8210494
16. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. **19**(3), 1628–1656 (2017). https://doi.org/10.1109/COMST.2017.2682318
17. Mannan, A., Bader, M., Hijawi, U., Gastli, A., Hamila, R.: Mesh network-based device-to-device negotiation system for energy management. In: 2021 18th International Multi-Conference on Systems, Signals Devices (SSD), pp. 1091–1098 (2021). https://doi.org/10.1109/SSD52085.2021.9429379
18. Mylonas, G., Amaxilatis, D., Chatzigiannakis, I., Anagnostopoulos, A., Paganelli, F.: Enabling sustainability and energy awareness in schools based on IoT and real-world data. IEEE Pervasive Comput. **17**(4), 53–63 (2018). https://doi.org/10.1109/MPRV.2018.2873855
19. Narottama, B., et al.: Base station energy efficiency of d2d device discovery. In: 2017 International Conference on Control, Automation and Information Sciences (ICCAIS), pp. 257–261 (2017). https://doi.org/10.1109/ICCAIS.2017.8217586
20. Petricic, A.: Predictable dynamic deployment of components in embedded systems. In: 2011 33rd International Conference on Software Engineering (ICSE), pp. 1128–1129 (2011). https://doi.org/10.1145/1985793.1986015
21. Qian, H., Andresen, D.: Extending mobile device's battery life by offloading computation to cloud. In: 2015 2nd ACM International Conference on Mobile Software Engineering and Systems, pp. 150–151 (2015). https://doi.org/10.1109/MobileSoft.2015.39
22. Song, B., Shihong, G., Xiaolong, Q.: Application of blackboard model based multiagent negotiation system in the distributed design of device-layer network. In: The 27th Chinese Control and Decision Conference (2015 CCDC), pp. 6005–6010 (2015). https://doi.org/10.1109/CCDC.2015.7161886
23. Srividhya, S., Sankaranarayanan, S.: IoT-fog enabled framework for forest fire management system. In: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 273–276 (2020). https://doi.org/10.1109/WorldS450073.2020.9210328

24. Sun, H., Wang, S., Zhou, F., Yin, L., Liu, M.: Dynamic deployment and scheduling strategy for dual-service pooling based hierarchical cloud service system in intelligent buildings. IEEE Trans. Cloud Comput. 1 (2021). https://doi.org/10.1109/TCC.2021.3078795
25. Tang, Q., Xie, R., Yu, F.R., Huang, T., Liu, Y.: Decentralized computation offloading in IoT fog computing system with energy harvesting: a dec-POMDP approach. IEEE Internet Things J. **7**(6), 4898–4911 (2020). https://doi.org/10.1109/JIOT.2020.2971323
26. Udoh, I., Kotonya, G.: A dynamic QoS negotiation framework for IoT services. In: 2019 IEEE Global Conference on Internet of Things (GCIoT), pp. 1–7 (2019). https://doi.org/10.1109/GCIoT47977.2019.9058418
27. Valarmathi, M., Sumathi, L., Deepika, G.: A survey on node discovery in mobile internet of things(IoT) scenarios. In: 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 01, pp. 1–5 (2016). https://doi.org/10.1109/ICACCS.2016.7586400
28. Yang, L., Guo, S., Yi, L., Wang, Q., Yang, Y.: NOSCM: a novel offloading strategy for NOMA-enabled hierarchical small cell mobile-edge computing. IEEE Internet Things J. **8**(10), 8107–8118 (2021). https://doi.org/10.1109/JIOT.2020.3042318
29. Yu-Jie, S., Hui, W., Cheng-Xiang, Z.: Balanced computing offloading for selfish IoT devices in fog computing. IEEE Access **10**, 30890–30898 (2022). https://doi.org/10.1109/ACCESS.2022.3160198
30. Zeng, M., Li, Y., Zhang, K., Waqas, M., Jin, D.: Incentive mechanism design for computation offloading in heterogeneous fog computing: a contract-based approach. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6 (2018). https://doi.org/10.1109/ICC.2018.8422684
31. Zhang, Z., Li, S.: A survey of computational offloading in mobile cloud computing. In: 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 81–82 (2016). https://doi.org/10.1109/MobileCloud.2016.15

E-Go Bicycle Intelligent Speed Adaptation System for Catching the Green Light

Khalil Ben Fredj(✉), Akhil Reddy Pallamreddy, Geert Heijenk, Paul Havinga, and Yanqiu Huang

Centre for Networked Systems and Intelligence, University of Twente, Enschede 7522 NB, The Netherlands
{k.benfredj,a.r.pallamreddy,geert.heijenk, p.j.m.havinga,yanqiu.huang}@utwente.nl

Abstract. The expanding growth of electric bikes in recent years underscores their increasing importance as a sustainable and eco-friendly mode of transportation. With zero emissions and the ability to ease urban congestion, e-bikes are becoming a pivotal solution in promoting greener and more efficient commuting habits. However, signalized intersections and frequent stops at traffic lights (TL) are considered uncomfortable for cyclists. This article introduces a personalized and privacy-preserving Intelligent Speed Adaption (ISA) system that helps cyclists adapt to the required speed to catch the green light. In our system design, traffic lights are augmented with Bluetooth Low Energy (BLE) beaconing devices which allow connected e-bikes to get the remaining green light phase duration, estimate the distance to the intersection, and assist the cyclist to catch the green light when necessary. We address the speed adaption problem as a convex optimization problem to ensure smooth and safe acceleration. In addition, a fuzzy logic controller is used to control motor power to reach the recommended speed while considering the human pedal power. We generate different scenarios with various initial velocities, time to red (TTR), slope of the road, and human pedal power to evaluate the system's performance. The results demonstrate that ISA improves the probability of crossing the traffic light by about 77% compared to the absence of speed adaptation.

Keywords: Smart e-bikes · Intelligent Speed Adaptation · Privacy Preserving · Fuzzy Logic Control · Bluetooth Low Energy · RSSI distance estimation · SPaT

1 Introduction

Bicycles are essential in our daily lives as a sustainable, affordable, and healthy mode of transportation [7]. Cycling not only provides a viable alternative to cars, reducing traffic congestion and air pollution, but also helps to improve

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 201–220, 2024.
https://doi.org/10.1007/978-3-031-63989-0_10

public health, reducing the risk of chronic diseases associated with a sedentary lifestyle [11]. However, cycling in urban areas is inconvenient because the traffic infrastructure is designed primarily for cars [15]. Cyclists have prolonged waiting times at traffic intersections due to TL [22]. In addition, they often have difficulties in catching the green light because the time allotted for bicycles to cross an intersection is much shorter than the time allocated to cars. As a result, cyclists must be quick to start pedaling and often have to travel faster than a comfortable pace to get through before the traffic light turns red.

Municipalities, researchers, and industry players have developed innovative approaches to address this problem, such as sensor-based smart traffic lights and smartphone application-based systems [1,4]. Worldwide, functioning smart traffic light systems are available, particularly in The Netherlands, where the government has installed intelligent traffic lights capable of communicating with vehicles and navigation applications [2]. These smart traffic lights use reactive algorithms and data from various sensors (radar, LiDAR, camera, pressing button, and induction loop) to dynamically adjust signal phase and timing (SPaT) based on the volume of vehicles/cyclists at intersections. However, these systems require abundant infrastructure and lack personalization for safety and comfort.

In contrast, more personalized smart systems adapting to traffic lights rely on smartphone Apps like "PrioBike," "Schwung," and "Enschede fietst app" [1,3,4]. These applications use green light time predictions and cyclists' smartphone data (position, direction, and speed) to offer speed recommendations. Third-party developers collect TL data from local agencies and analyze them alongside cyclists' private data (GPS location, directions, speed) using time series prediction methods (e.g., LSTM and 1D-CNN). For example, to give a speed recommendation, the Green Light Optimal Speed Advisory (GLOSA) system estimates the distance between the TL and the bicycle from the GPS location of the bicycle and the location information received from the TL [24]. However, these systems have limited availability based on data-sharing agreements and require continuous user tracking for speed recommendations. While both sensor-based smart traffic lights and smartphone application-based systems have shown promising results, there is a need for a decentralized and anonymous local information-based solution.

In this work, the authors aim to address this limitation by considering dynamic expected velocities based on the distance to the traffic light, human pedal power, road slope, and time-to-red (TTR), ensuring appropriate motor power adjustments for efficient and safe cycling. This study establishes a personalized and privacy-preserving intelligent system that allows cyclists to catch green lights while maintaining comfort and safety. Usually, when cyclists approach the traffic light and see that it is green, they try to accelerate to cross the intersection before the TL turns red. However, they do not have information about the remaining green time. Consequently, either they put an unnecessary effort and cross the green light before it turns to red or the green phase duration is so short that they end up violating the red light. Assisting the cyclist to safely accelerate to the required speed when possible by catching the current green light phase will ensure safety and comfort.

The main contributions of this work include the following:

- This is the first work combining Bluetooth Low Energy (BLE) communication with the fuzzy logic controller-based motor power control system to adapt the bicycle speed for catching the green light.
- It formulates the speed adaptation problem as a convex optimization problem and provides the optimal strategy for controlling motor power to catch the green light while ensuring safety and comfort, which goes beyond the current state-of-the-art.
- The system in this work dynamically updates and enhances the distance estimates along the journey using a Kalman filter and the continuously received beacons from the TL, which is simple and preserves users' privacy compared to GPS-based solutions.

The paper is organized as follows. Section 2 introduces related work on motor power control and distance estimation methods. The system modeling and problem formulation are presented in Sect. 3. In Sect. 4, we provide the proposed speed adaptation system consisting of distance estimation and speed adaptation. The experimental and numerical results are illustrated in Sect. 5 followed by the discussion and future work in Sect. 6 and conclusion in Sect. 7.

2 Related Work

In this section, we present an overview of the present pinnacle of speed adaptation techniques aimed at effectively catching the green light. The manipulation of motor power in the bicycle's mechanics on the one hand, and the accurate estimation of distance between the e-bike and the traffic light on the other, constitute the pivotal components underpinning the functionality of the speed adaptation system. Consequently, we will explore separately the state of the art of the motor power control methods and the distance estimation techniques in the following two subsections.

2.1 Motor Power Control Methods

Thanks to several researchers, including [5,8,12,13,16–20,23,24] it is now possible to control the motor power in the bicycles to adapt their speed. So far, two types of assisted power methods are available for commercial electric-assist bicycles. The first is called constant-assisted power (CAP) [19]. It provides the rider with a consistent and predetermined level of power support, regardless of the pedal power applied. The second is proportional assisted power (PAP) [20]. The idea is to assist motor power proportionally to pedal power until a specific speed limit and then gradually recede. The limitations of these approaches are that they do not consider environmental factors contributing to the resistance (e.g., air drag, road surface friction), leading to tapering off the riding comfort and safety.

To address the problem of dealing with the different environmental circumstances that bicycles encounter a reinforcement learning (RL) based algorithm

is proposed in [13] to improve riding quality considering different road slopes and human pedal power. However, this RL-based solution requires abundant data and high computations, especially in dynamic environments for updating optimal policy to provide sufficient riding quality. To ensure riding quality, [5] proposed a sophisticated algorithm considering wind force and road slopes using a mixed controller to maintain a constant velocity, but its reliance on reliable inputs raises concerns in dynamic environments. To address the limitations posed from the above, a fuzzy logic controller (FLC) has been used to assist the cyclists [8,12,16–18,23,24]. The FLC demonstrates exceptional performance in dealing with vague and uncertain inputs and allows one to model and control complex systems in an intuitive and human-interpreted way. For example, Lee and Jiang [16] have proposed a solution that adapts the bicycle's speed by achieving the urban riding velocity (assuming the expected velocity to be 18 km/h throughout the journey) for ordinary, power-less, and powerful cyclists.

Although the results to date were prominent in maintaining comfort and safety throughout the journey, existing FLC-based methods can only adapt the speed to a predetermined constant velocity. In the real-time application realization, many factors affect the performance of the system, including stops in the TL. In order to catch the green light, the expected velocity should be dynamic along the journey, which depends on the distance to the TL, human pedal power, road slope, and TTR. Obtaining the dynamic expected velocity and adapting motor power accordingly remain overlooked. To the best of our knowledge, this is the first time that anyone has considered the dynamic change in the expected velocity for speed adaptation.

In this work, the authors aim to tackle this limitation by considering dynamic expected velocities based on the distance to the traffic light, human pedal power, road slope, and time-to-red (TTR), ensuring appropriate motor power adjustments for comfort and safe cycling.

2.2 Distance Estimation Methods

In order to find the dynamic expected velocity for crossing the traffic light, the distance between the bicycle and the TL is crucial to determine. The distance can be realized using positional information with Global Positioning System (GPS), sensors like camera, ultrasonic and LiDAR. One of the most used solutions for estimating the distance to the traffic light is GPS location information. For instance, in the Green Light Optimal Speed Advisory (GLOSA) system, the distance between the TL and the bicycle is derived from the GPS location of the bicycle and the location information received from the TL [24]. In addition, it gives an optimal recommended speed for the rider when approaching the traffic light. However, it may overwhelm the rider with excessive information and the rider is responsible for acknowledging and responding to this information, as the proposed system does not assist in adjusting the bicycle's speed.

Although GPS is widely available on all smartphones, sharing the location of cyclists might affect their privacy and make them vulnerable to malicious activities by tracking their movements and mobility history. To improve the

distance estimation accuracy using positional solutions, [21] have combined the Global Navigation Satellite System (GNSS), inertial and magnetic data using an extended Kalman filter to enhance bicycle localization in urban environments. However, this method requires additional computation and hardware, such as GPS equipment and Inertial measurement units on the bicycle. In addition, the TLs should also know their position information that is accessible to the GPS for accurately estimating the distance.

Another wireless solution proposed in [10] has used a combination of a log-normal shadowing model and an adaptive neural fuzzy inference system to reduce the distance estimation error between bicycles and anchor nodes. However, the computation and convergence time depend on a set of parameters, such as the number of membership functions, which is unsuitable for our application when the bicycle has to adapt its speed during a limited time window. Furthermore, several real-time applications use sensors like cameras, LiDAR, and ultrasonic to estimate the distance. However cameras and LiDAR are expensive, they could be affected by the environmental conditions and use high computational power in identifying the objects (in this case TL) that are required to estimate the distance. Meanwhile, [6] enabled cars to detect bicycles using Bluetooth beacons and trigger an alert to the driver when the cyclist is approaching closer to 50 m [6]. The path loss model and the Received Signal Strength Indicator (RSSI) of the received beacons enable the calculation of the distance. Yet, the distance estimation is not accurate enough due to reflection, and scattering, which brings distance estimation error and might affect the possibility of catching the green light in our application.

In this work, we aim to provide a low-cost solution for distance estimation for the long-rage application to 100 m, to avoid privacy issues presented when sharing the bicycles' GPS location, and dynamically enhance the RSSI-based distance estimation along the journey to cross the intersection.

3 System Modeling and Problem Formulation

In this section, we first explain the bicycle dynamics and the parameters used for the fuzzy logic controller. Then we will formalize the adaptation strategy as a convex optimization problem to smoothly adapt the bicycle's speed while preserving the cyclist's comfort and safety. In this paper, safety and comfort refer to maintaining acceleration and speed in a predefined range. This range might change according to the cyclist's preference and capabilities. However, to be able to assess our system we will consider safety by maintaining an acceleration in the range of [–0.4, 0.4 m/s^2] and a speed in the range of (16–20 km/h), as recommended in the literature [13].

3.1 System Modeling of Bicycle Dynamics

A bicycle in motion follows specific physics laws. The power required to move a bicycle with an expected velocity v_{exp}, has to deal with resisting forces. There

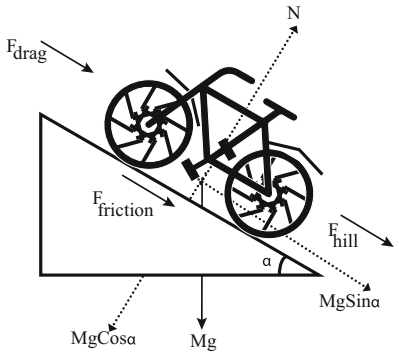


Fig. 1. Free-body diagram of all resisting environmental forces acting on a bicycle when moving uphill.

are three resisting forces as shown in Fig. 1: F_{drag} air drag, F_{hill} gradient drag, and $F_{friction}$ road surface friction at the point of contact on the ground.

The total power required, P_{req}, is the product of resisting force (sum of all resisting forces) and the expected velocity, v_{exp} (since $Power = Force \times Velocity$):

$$P_{req} = (F_{drag} + F_{hill} + F_{friction}) \times v_{exp} \quad . \tag{1}$$

F_{drag} is characterized by the atmospheric drag coefficient C_d, frontal area A_c, atmospheric density ρ, and bicycle's relative velocity to wind $v_{relative}$ (in our simulation the wind's velocity is negligible, hence this $v_{relative}$ eventually becomes v_{curr}):

$$F_{drag} = \frac{C_d\, A_c\, \rho}{2} \, (v_{relative})^2 \quad . \tag{2}$$

F_{hill} is proportional to the object's mass m, gravitational force g, the angle of elevation/depression of the road surface α:

$$F_{hill} = m\, g\, \sin\alpha \quad , \tag{3}$$

and $F_{friction}$ is governed by the friction coefficient μ, the object's mass m, and gravitational force g and the angle of elevation/depression of the road surface α:

$$F_{friction} = m\, g\, \mu\, \cos\alpha \quad . \tag{4}$$

The total power provided to the bicycle's transmission system P_{req} is the sum of the human pedal power and the motor power, which decides the current velocity of the bicycle v_{curr}.

$$P_{req} = P_{human} + P_{motor} \quad . \tag{5}$$

The current velocity of the bicycle v_{curr} should reach the expected velocity v_{exp} as close as possible under the influence of the controller. By substituting all

the resisting forces Eqs. (2) to (4) in Eq. (1) using Eq. (5) and rearranging, we can get

$$v_{curr} = \frac{P_{human} + P_{motor}}{F_{drag} + F_{hill} + F_{friction}} \quad . \tag{6}$$

This equation determines how much motor power should be provided by FLC to reach the expected velocity.

3.2 Problem Formulation of Bicycle Speed Adaptation

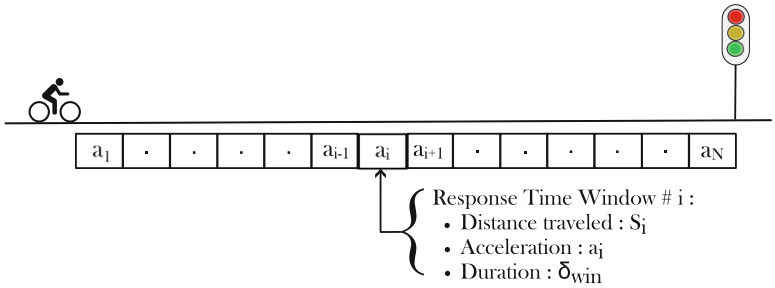


Fig. 2. Scenario representation where Time To Red is divided into N equal Response Time Windows (RTW) of a duration δ_{win} each (Color figure online)

According to the bicycle dynamics in Sect. 3.1, we can control the motor power to reach any expected velocity when the human pedal power and environmental-related forces are known. However, the expected velocity is directly linked to the safety and comfort of cyclists. Therefore, our aim is to smoothly accelerate the bicycle, when the current velocity v_{curr} is not sufficient to catch the green light while maintaining safety and comfort. In theory, deceleration is possible, but this work focuses solely on acceleration due to two reasons: i) We aim to retain human control and avoid complete system takeover. This means no motor assistance if there is no human pedal power. ii) Implementing deceleration alongside acceleration requires additional electronic blocks, which is beyond the scope of this work.

We formulate the problem as a convex optimization problem where we minimize the variance of the accelerations while: i) ensuring the possibility to catch the green light ii) not exceeding preset acceleration and velocity thresholds. The optimization problem can be expressed as follows:

$$\underset{a_i}{\text{argmin}} \quad \sum_{i=1}^{N} (a_i - \frac{1}{N} \sum_{j=1}^{N} a_j)^2 \quad , \tag{7}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} S_i \geq D_{TL} \quad , \tag{8}$$

$$v_i \leq v_{max} \quad for \quad i \in [1....N] \quad , \tag{9}$$

$$0 \leq a_i \leq a_{max} \quad for \quad i \in [1....N] \quad . \tag{10}$$

As seen in Fig. 2, $S_i = v_{i-1}\,\delta_{win} + \frac{1}{2}\,a_i\,\delta_{win}^2$ is the distance traveled at the i^{th} time window of a duration δ_{win}, where $N = \lfloor \frac{TTR}{\delta_{win}} \rfloor$ represents the number of Response Time Windows (RTW) of the system, v_{max} is the maximum allowed velocity, a_{max} is the maximum acceleration that could be tuned by the cyclist according to his comfort and safety range, and D_{TL} is the distance between the bicycle and the TL.

Once a_i is calculated, we can derive $v_{exp,i}$ as follows:

$$v_{exp,i} = v_{exp,i-1} + a_i\,\delta_{win} \quad . \tag{11}$$

After getting the expected velocity, a controller shall supply the motor assistance with the required power to reach this expected velocity.

4 Methodology and System Design

This section introduces our system design and methodologies for solving the speed adaptation problem formulated in Sect. 3.2.

As shown in Fig. 3, the ISA consists of two parts: a Control Unit, and a FLC. The control units takes TTR and the RSSI measurements received from the BLE module as input and estimates the distance to the TL and calculates the expected velocities. Then, the FLC estimates the required motor power to reach the expected velocity v_{exp} to assist the rider in catching the green light. The expected velocity in this case is the required velocity to move the bicycle from one window to the other. One of the inputs to the FLC is the velocity error v_{error}, which is the difference between the expected velocity v_{exp} to catch the green light and the bicycle's current velocity v_{curr}. The other input is the Human pedal power P_{human}. The v_{curr} in practical can simply be achieved using a speed sensor in the bicycle, however for the simulation we realize this using Eq. (6). We will introduce the details of the control unit and the FLC in the following sub-sections.

4.1 Control Unit: Expected Velocity Calculation Based on Dynamic Distance Estimation Combining RSSI Path Loss Model with Kalman Filter

In this subsection, we introduce how we estimate the distance to the TL on the one hand and how we derive the expected velocity to catch the green light on the other hand.

The Control Unit, as shown in Fig. 4, extracts the RSSI measurements and the TTR value from the BLE module and estimates the required acceleration to catch the green light. The distance estimation error is quite significant as the RSSI measurements fluctuate even when the bicycle is at the same distance from

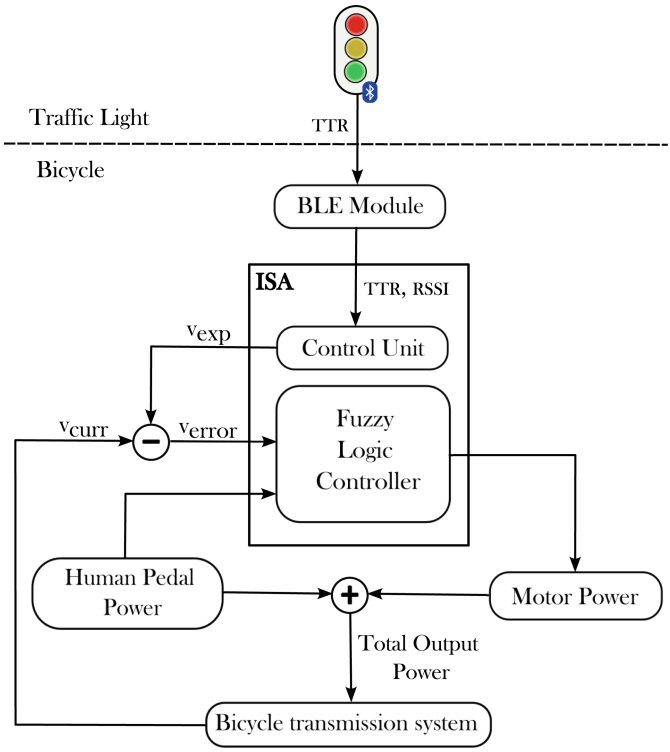


Fig. 3. Proposed communication-based ISA for e-bikes

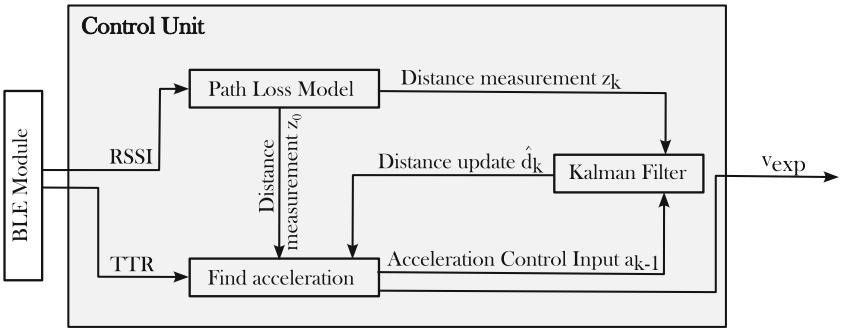


Fig. 4. Control Unit

the TL. Therefore, it affects the cyclist's comfort and the chance to catch the green light. To mitigate this issue, we apply a Kalman filter to enhance distance estimation accuracy while the bicycle is moving. The system state of the bicycle evolves from a previous state to the next state following Newton's law of motion and can be expressed as follows:

$$X_k = A\,X_{k-1} + B\,a_{k-1} + w_{k-1} \quad , \tag{12}$$

where $X_k = \begin{pmatrix} d_k \\ v_k \end{pmatrix}$, d_k and v_k are the distance and speed of the bicycle at time k, respectively. $A = \begin{pmatrix} 1 & -\delta_t \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} -\frac{\delta_t^2}{2} \\ \delta_t \end{pmatrix}$, δ_t is the period of the distance measurement inputs to the Kalman filter, and w_{k-1} is the uncertainty of the model with a covariance Q.

Additionally, the measurement z_k, is the measured distance based on the extracted RSSI values obtained using Eq. (15):

$$z_k = HX_k + u_k \tag{13}$$

Where $H = \begin{pmatrix} 1 & 0 \end{pmatrix}$ and u_k is the measurement noise with variance R. According to Friis formula for free-space transmission [9], the distance can be calculated using the RSSI based on the log-distance model [14], as follows:

$$RSSI[dBm] = C - 10\,n\,\log_{10}(d) \tag{14}$$

Here C is a constant, n is the path loss exponent, and d is the distance in meters between the transmitter and the receiver. Hence, we could derive the distance measurement between the TL and the bicycle from the equation as follows:

$$z_k = 10^{\frac{C - RSSI_k}{10n}} \tag{15}$$

Since both the system model and the measurements have noise, we could obtain the distance estimation using the KF equations in Eq. (16) to find the best trade-off between them.

$$\begin{aligned} \hat{X}_k^- &= A\,\hat{X}_{k-1}^- + B\,a_{k-1} \\ P_k^- &= A\,P_{k-1}\,A^T + Q \\ K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ \hat{X}_k &= \hat{X}_k^- + K_k(z_k - H\hat{X}_k^-) \\ P_k &= (I - K_k H)\,P_k^- \end{aligned} \tag{16}$$

where P_k denotes the a posteriori estimate covariance matrix. Then we find the distance estimate as:

$$\hat{d}_k = H\hat{X}_k \tag{17}$$

The details of dynamically updating the distance estimates through the Kalman filter are listed in Algorithm 1. Since the Kalman filter needs several measurements to converge, we split the duration of the entire journey into Nu windows. Each window is of a period $\Delta_t = \frac{TTR}{N_u}$ (line 8). During the duration of Δ_t and at each δ_t seconds ($\delta_t \ll \Delta_t$), the TL sends a $\delta_t - periodic$ beacon containing a TTR value to the bicycle and the Kalman filter uses the distance measurements of the RSSI path-loss model and the control input acceleration to update

the distance estimation (line 21 to line 23). At the end of each period Δ_t, we decrement the value of TTR by Δ_t and calculate the list of accelerations and the corresponding expected velocities using the last distance update from the Kalman filter (line 24 to line 29). Here we mention that the initial list of accelerations is calculated using only the distance measured through the path-loss model at the reception of the first beacon (line 13 and line 14). Having in hand the list of expected velocities we move to the next section to explain how we use it in order to estimate the required motor power to assist the cyclist to catch the green light.

Algorithm 1. Intelligent Speed Adaptation

1: **Initialization:**
2: Inputs: N_u, δ_{win}, P_{human}, α
3: Output: P_{motor}
4: FLC ← Fuzzy simulation using membership functions
5: Power motor ← MotorPower(fuzzy simulation, V_{error}, P_{human})
6: **Main:**
7: **while** a beacon is received & Traffic Light Phase is Green **do**
8: Calculate $\Delta_t = \frac{TTR}{N_u}$
9: Calculate $i = \lfloor \frac{\Delta_t}{\delta_{win}} \rfloor$
10: Calculate $N = \frac{TTR}{\delta_{win}}$
11: $Timer = \Delta_t$
12: Start $Timer$
13: Calculate the distance measurement input of the Kalman Filter z using Equation (15)
14: Calculate $[a_1, ., a_N]$ and $[v_{exp,1}, ., v_{exp,N}]$ through Equation (7) and Equation (11) using z as the distance to the traffic light ($D_{TL} = z$).
15: **if** $Timer == 0$ **then**
16: $TTR \leftarrow TTR - \Delta_t$
17: $Timer = \Delta_t$
18: Start $Timer$
19: **end if**
20: $k = 1$
21: **while** a beacon is received & Traffic Light Phase is Green **do**
22: Calculate the distance measurement input of the Kalman Filter z through Equation (15) using the received $RSSI$
23: Calculate the Kalman filter distance update $\hat{d}$ through Equation (12) - Equation (17).
24: **if** $Timer == 0$ **then**
25: $TTR \leftarrow TTR - \Delta_t$
26: **if** $TTR <= 0$ **then**
27: break
28: **else**
29: Calculate $[a_{ki}, ., a_N]$ and $[v_{exp,ki}, ., v_{exp,N}]$ through Equation (7) and Equation (11) using the last updated $\hat{d}$ and TTR
30: Calculate v_{curr} using Equations (2) to (4) and (6)
31: Calculate $v_{error} = v_{exp} - v_{curr}$
32: Calculate P_{motor} using inputs P_{human} & v_{error} with FLC
33: $k \leftarrow k + 1$
34: $Timer = \Delta_t$
35: Start $Timer$
36: **end if**
37: **end if**
38: **end while**
39: **end while**

4.2 Fuzzy Logic Controller for Adapting the Speed

This sub-section explains how FLC controls the motor power to reach the expected velocity v_{exp} obtained by the control unit above.

The FLC processes the inputs v_{error} and P_{human} in three stages: a) fuzzification, b) inference, and c) defuzzification, to estimate the motor power P_{motor}. In the fuzzification stage, crisp input values are transformed into fuzzy variables using the membership function (MF) shown in Fig. 5. Next, the fuzzy inference system (FIS) applies fuzzy rules and operators (e.g., AND, NOT, OR) from Table 1 to generate a new set of fuzzy variables. Finally, the FIS's fuzzy variables are converted back to crisp output values using centroid defuzzification. To maintain mathematical simplicity, the triangular membership function is utilized to prepare the MFs and linguistic variables with their abbreviations, all based on heuristic rules.

Table 1. Fuzzy inference rules for velocity error & human pedal power inputs and motor power output

Velocity error (v_{error})	Human pedal power (P_{human})			
	VL	L	M	H
		Motor	Power	(P_{motor})
Z	VL	VL	L	L
VL	VL	L	L	M
L	VL	L	M	M
M	L	M	M	H
MH	L	M	H	H
H	M	M	H	VH
VH	H	H	VH	VH

For the v_{error}, seven linguistic variables are created including **Zero (Z)**, **Very Low (VL)**, **Low (L)**, **Medium (M)**, **Medium High (MH)**, **High (H)**, and **Very High (VH)** Fig. 5 (a). Since regulations in the European Union require turning off the motor assistance when the velocity exceeds 25 km/h, the velocity error MF is thereby in the range of (0–25 km/h).

The human pedal power is represented by four linguistic variables: **Very Low (VL)**, **Low (L)**, **Medium (M)**, and **High (H)**. While Fig. 5 (b) shows the power range as (0–100) W, the MF can be dynamically adapted by adjusting the triangular center points to accommodate different power ranges exerted by cyclists. The human pedal power MF is constructed by adopting the slope gradient and corresponding human pedal power values from [13] as seen in Table 2. The slope gradient is between 0–3%, and their corresponding human pedal power is modeled from 45–125 W with distinct variances.

The motor power is represented by five linguistic variables: **Very Low (VL)**, **Low (L)**, **Medium (M)**, **High (H)**, and **Very High (VH)**, which define the

fuzzy sets of output motor power. As our approach utilizes Mamdani FIS, the output MF is constructed with fuzzy sets using if-then inference rules to handle uncertainties, as depicted in Table 1 [16].

Referring to [16], the construction of the 28 specified fuzzy rules in Table 1 involves a heuristic approach using the "Fuzzy AND" operator. The rules are as follows: First, when pedal power is constant, an increase in velocity error necessitates more motor power. Second, when velocity error is constant, an increase in pedal power also requires additional motor power. Third, the motor power should be reduced to zero when either human pedal power tends to zero or the current velocity exceeds 25 km/h. These rules determine the fuzzy output variable using the fuzzy max-min inference technique.

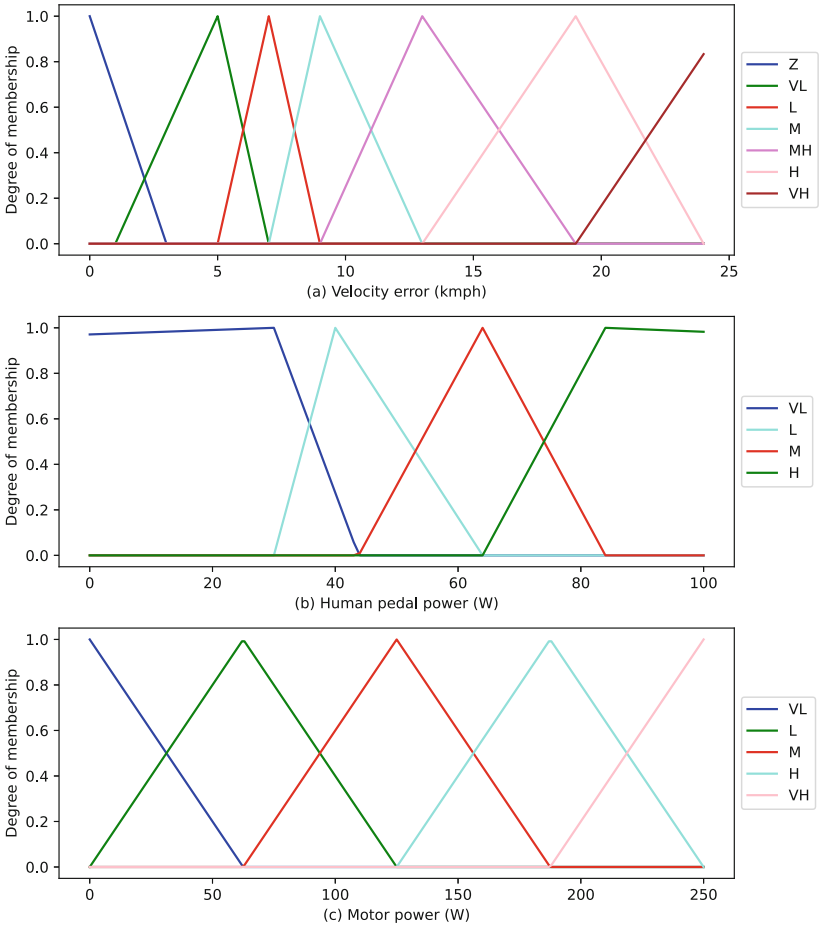


Fig. 5. FLC's linguistic variables and their associated membership functions for inputs a) Velocity error, (b) Human pedal power, and output (c) Motor power.

Table 2. Slope gradient and it's corresponding human pedal power

Parameters	Slope grade			
	0%	1%	2%	3%
Total percent of individual slopes in the entire sampled slope data	70%	15%	10%	5%
Human pedal power mean and variance for different slope values	45,30	65,40	95,50	125,60

5 Results

In this section, we simulate bicycle's dynamics and combine that with the implemented controller and samples taken from a large set of BLE propagation measurements. In addition, we evaluate the performance of ISA in terms of comfort, safety, and the chance to catch the green light. As explained previously, comfort and safety entail the preservation of acceleration and velocity within predetermined ranges. These ranges are guaranteed by the solution design through Eq. (10) and Eq. (9). The constant parameters required for simulation in the equations are given in Table 3.

Table 3. Simulation Parameters

Conditions	Parameters	Value
Environmental	C_d	*0.5*
	A_c	*1* m^2
	ρ	*1.18* kg/m^3
	μ	*0.014*
	α	*(0,1,2,3)%*
	g	*9.8* m/s^2
Rider	***mass***	75 kg
	pedal power	*(0 - max W)*
Journey	$D_{TL,init}$	100 m
	v_{init}	$v_{init} \in [10, 20] Km/h$
	TTR_{init}	$TTR \in [15, 25] s$
	v_{max}	20 Km/h
	a_{max}	$0.4 m/s^2$
	f_b	4 Hz
	δ_{win}	1 s
	N_u	4
Number of simulations	–	2^{15}

5.1 Data Collection and Generation

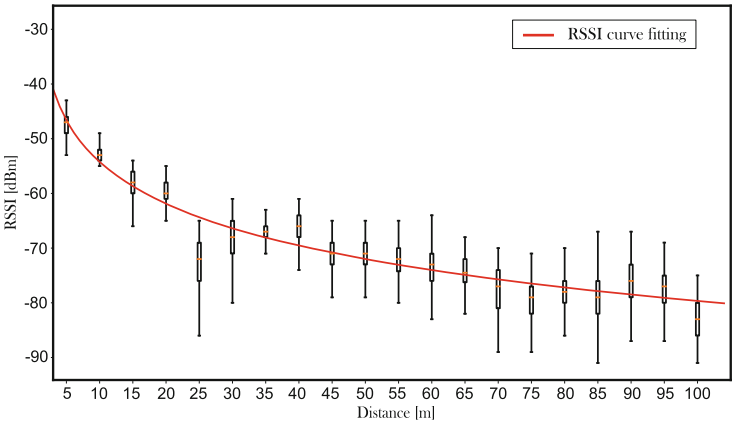


Fig. 6. RSSI Path Loss model with $n = 2.5$ and $C = -28.8$

We have collected RSSI measurements using two Nordic development kits NRF52832, one as a transmitter (Tx) and the other as a receiver (Rx). Both boards were running the BLE v5 protocol stack. Segger-embedded systems and C++ were used to program the two boards; one was configured as the BLE central device (Rx), and the other was configured as a peripheral device (Tx). The transmission power for the experiments has been set to 0 dBm. We have recorded the RSSI on the receiver side at each step of 5 m to a maximum distance of 100 m between the central and peripheral nodes. More than 150 packets containing RSSI values have been recorded at each step. In our setup, we assume that the roadside unit (in this case TL), Tx, will have a Line Of Sight (LOS) with the On-Board Unit (OBU), Rx, on the bicycle. Therefore, we ensured a clear LOS between Tx and Rx in the experimental measurements. One major motivation behind this assumption is that in many European countries, such as the Netherlands, the separation between bicycle lanes and car lanes is widely considered for safety, so the LOS is less to be obstructed by obstacles such as cars and trucks. After fitting the data curve, as shown in Fig. 6, we obtain from Eq. (15): $C = -28.8$ and the path loss exponent $n = 2.5$. These values depend on the measured environment and we assume that during equipping the TLs with BLE modules calibrations have to be done.

Furthermore, we have synthetically generated the slope and human pedal power data, for the simulation, provided in [13]. The slope values are in the range (0–3%). This means increase in the road's gradient, in meters, over a horizontal run of 100 m. For example, a 2% slope constitutes a rise of 2 m over a distance of 100 m. The slope generation involves uniform distribution for respective slope values as shown in Table 2 along with an additive white Gaussian noise of mean

and standard deviation 0 and 0.2 to resemble the real world road profile. The generation of human pedal power corresponding to the slope involves using a normal distribution with mean and variance values listed in e.g., a slope of 0% will have 70% of the total distribution Table 2.

5.2 Impacts of Distance Estimation Error on ISA

To evaluate the distance estimation accuracy, we have generated many scenarios of journeys. Each journey starts when the bicycle receives the first beacon and ends when it reaches the TL. At the start of each journey, we randomly generate an initial velocity v_{init} and TTR_{init} value as mentioned in Table 3, where *init* refers to the value at the reception of the first beacon.

To calculate the distance estimation error, we first obtain the real distance of the bicycle to TL along the journey. It is calculated by first solving the optimization problem in Eq. (7), to obtain a list of accelerations assuming that the initial distance to the traffic light is known; then we get the real distance of the bicycles at every second through Newton's law of motion. Further, at each distance of 5 meters, we compare the estimated distances of the Kalman Filter and the Path Loss model to the real distance between the bicycle and the TL. The RSSI values are fed to the Control Unit at a frequency of $f_b = 4\,\text{Hz}$. As we collected the RSSI values only at discrete points of every 5 m, we randomly select one RSSI value from the set of collected data that is closest to the current position of the bicycle.

In Fig. 7, we show the Mean Absolute Error (MAE) of the estimated distance of Kalman filter and path-loss model. The mean is calculated over 2^{15} combinations of initial velocities, TTR, road slopes, and human pedal power. As expected, the Kalman filter has less MAE than the Path-loss model, as we are reducing the noise coming from the RSSI measurements. Due to the fact that our RSSI collected data is only done at each distance of 5 m, in Fig. 6, we are losing accuracy of at least 2.5 m for the distance estimation. Therefore, collecting RSSI data at several different distances will give us better distance estimation accuracy.

To give a more straightforward illustration of the distance estimation, we show in Fig. 8 one scenario where $v_{init} = 15.47\,\text{km/h}$ and $TTR_{init} = 21.58\,\text{s}$. We have simulated this scenario for multiple runs considering the random selection of the RSSI measurements from the corresponding distances. At each second, we average all the estimated distances and compare those mean values of Kalman Filter and Path Loss model with the real distances. The results are consistent with Fig. 7 that ISA method is closer to the optimal solution and less fluctuating than using only Path Loss model for distance estimation.

5.3 ISA Performance Evaluation Based on Success Percentage of Catching the Green Light

We have evaluated our system based on the percentage of scenarios where we successfully catch the green light. We have generated random values of initial

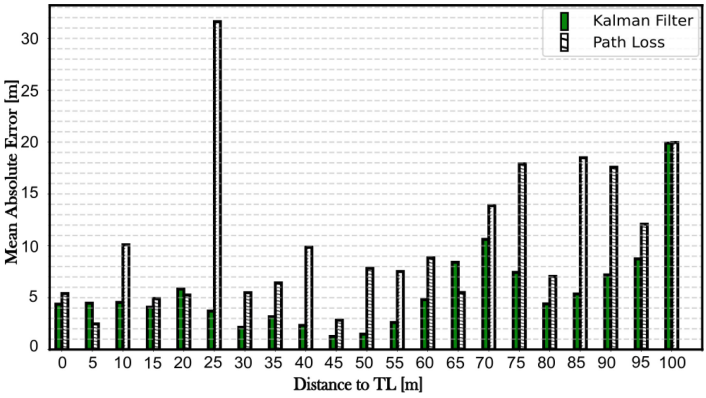


Fig. 7. Mean Absolute Error of Distance Estimation.

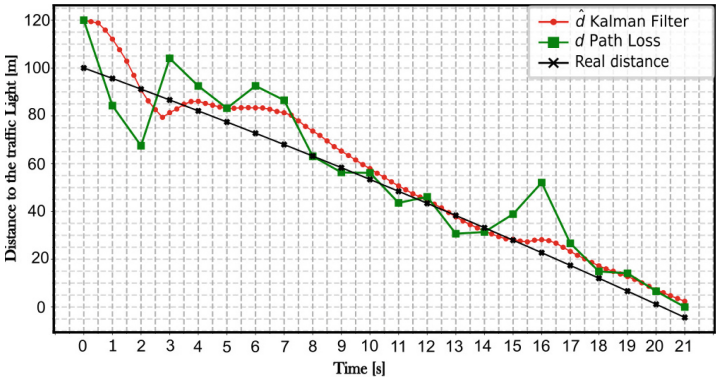


Fig. 8. Average Estimated distances using Kalman Filter and Path Loss model compared to real distances to the TL through one journey with $v_{init} = 15.47\,\text{km/h}$ and $TTR_{init} = 21.58\,\text{s}$

velocities in the range of [10, 20] km/h and initial TTR_{init} in the range of [15, 25] s. Then we defined 2^{15} combinations of initial velocities, initial TTRs, road slopes and human pedal power. Each combination defines one scenario. The rest of the simulation parameters can be found in Table 3. Furthermore, due to the formulation of convex optimization problem the control unit ensures the safety and comfort metrics. It is prepared to satisfy the upper and lower inequality constraints presented in Eq. (7).

In Table 4, we illustrate a range of techniques alongside their respective success rates in capturing the green light. It is essential to note that the success rates in this analysis are derived from a subset comprising 57.80% of the 2^{15} simulation scenarios wherein catching the green light is physically feasible. Regarding the initial scenario, where the bicycle maintains a constant speed without any

Table 4. The percentage of success in catching green light

Method	Success percentage
Without speed adaptation	37.54%
Speed adaptation with Path Loss model	60%
ISA	66.40%

adjustments, the success rate is recorded at 37.54%, indicating the lowest success rate among the techniques considered.

Comparatively, the Intelligent Speed Adaptation (ISA) method demonstrates a notable 66.40% probability of catching the green light, marking a significant enhancement of nearly 77% compared to the non-adaptive approach. Moreover, this improvement is approximately 10% relative to that achieved through the implementation of the path-loss model, attributable to the more accurate distance estimations provided by the ISA methodology. Attaining a flawless success rate hinges on the system's ability to obtain precise distance estimates to the traffic light and promptly adjust the speed upon receiving the first beacon. However, our system relies on obtaining additional beacons during the journey to refine the distance estimation to the traffic light, enabling the calculation of the necessary acceleration for intercepting the green light.

6 Discussion and Future Work

This work serves as a starting point in the design of smart bicycles for self-speed adaptation to assist cyclists in catching the green light. There are, however, several aspects that need to be investigated in the future. In particular, we anticipate continuing our research work along the following lines.

- ***Deceleration and large-scale evaluation.*** As a proof-of-concept, ISA is evaluated when there is only one bicycle that needs to accelerate to catch a traffic light. We need to further investigate how to decelerate and how to quantify the impacts of other road users on the speed adaptation. To include both acceleration and deceleration in one system will increase the complexity, and hence we first focused only on acceleration. In addition, physical implementation with embedded systems is needed to test the system in the real traffic scenarios. Time and battery consumption saved will be further used to evaluate the system when there are multiple traffic lights along the journey.
- ***Interaction with smart traffic lights.*** The results showed in this paper are based on a normal traffic light with fixed signal phase and timing (SPaT). Smart traffic lights are able to change the SPaT according to the traffic situations. Nevertheless, we believe that the basic building blocks of ISA are well posed to tackle the dynamical SPaT because of two reasons. First, ISA can obtain the new SPaT information based on the periodically received BLE beacons; and second, ISA dynamically estimates the distance to the traffic

light along the journey and updates the control policy accordingly. We need to generate different scenarios to test this functionality and evaluate the effectiveness in the next step.

7 Conclusion

This paper proposes an intelligent speed adaptation system at traffic lights for cyclists to increase the chances of catching the green light. The system combines BLE communication with FLC based motor power control and can be fully implemented on the bicycle. It overcomes the limitations of existing methods in two ways: first, compared with smart traffic lights that adapt signal phase to prioritize a group of cyclists conditionally, ISA provides personalized assistance considering the cyclist's velocity, pedal power, distance to the traffic light, time to red, and slopes of roads; second, compared with smartphone application systems that adapt to traffic lights, ISA does not leak users' private information to third parties. Extensive simulation results show that our system improves the probability of catching a green light by 77% compared to no speed adaptation while maintaining the safety and comfort of the cyclist.

References

1. Enschede Cycling App - Enschede Cycling City. https://enschedefietsstad.nl/enschede-fietst-app/
2. Intelligent Traffic Light Installations — Rijkswaterstaat. https://www.rijkswaterstaat.nl/en/mobility/smart-mobility/talking-traffic/intelligent-traffic-light-installations
3. PrioBike-HH: Enhancing cycling comfort and safety in Hamburg, Interreg VB North Sea Region Programme. https://northsearegion.eu/bits/news/priobike-hh-enhancing-cycling-comfort-and-safety-in-hamburg/
4. Schwung - Lekker snel door groen. https://schwung.nu/
5. Abagnale, C., Cardone, M., Iodice, P., Strano, S., Terzo, M., Vorraro, G.: Model-based control for an innovative power-assisted bicycle. Energy Procedia **81**, 606–617 (2015). https://doi.org/10.1016/J.EGYPRO.2015.12.045
6. Anaya, J.J., Talavera, E., Gimenez, D., Gomez, N., Felipe, J., Naranjo, J.E.: Vulnerable road users detection using V2X communications. In: IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, vol. 2015-Octob, May 2020, pp. 107–112 (2015). https://doi.org/10.1109/ITSC.2015.26
7. Bopp, M., Sims, D., Piatkowski, D.: Bicycling for transportation: an evidence-base for communities. Bicycling for Transportation: An Evidence-Base for Communities, pp. 1–227 (2018). https://doi.org/10.1016/C2016-0-03936-0
8. Chen, P.H.: Intelligence application: elebike fuzzy control Part I. In: Proceedings of the 7th International Conference on Machine Learning and Cybernetics, ICMLC, vol. 6, pp. 3581–3585 (2008). https://doi.org/10.1109/ICMLC.2008.4621025
9. Friis, H.T.: A note on a simple transmission formula. Proc. IRE **34**(5), 254–256 (1946). https://doi.org/10.1109/JRPROC.1946.234568

10. Gharghan, S.K., Nordin, R., Jawad, A.M., Jawad, H.M., Ismail, M.: Adaptive neural fuzzy inference system for accurate localization of wireless sensor network in outdoor and indoor cycling applications. IEEE Access **6**, 38475–38489 (2018). https://doi.org/10.1109/ACCESS.2018.2853996
11. Gojanovic, B., Welker, J., Iglesias, K., Daucourt, C., Gremion, G.: Electric bicycles as a new active transportation modality to promote health. Med. Sci. Sports Exerc. **43**(11), 2204–2210 (2011). https://doi.org/10.1249/MSS.0B013E31821CBDC8
12. Guarisco, M., Gao, F., Paire, D.: Autonomy and user experience enhancement control of an electrically assisted bicycle with dual-wheel drive. In: IEEE Industry Application Society - 51st Annual Meeting, IAS 2015, Conference Record, December 2015. https://doi.org/10.1109/IAS.2015.7356842
13. Hsu, R.C., Liu, C.T., Chan, D.Y.: A reinforcement-learning-based assisted power management with QoR provisioning for human-electric hybrid bicycle. IEEE Trans. Ind. Electron. **59**(8), 3350–3359 (2012). https://doi.org/10.1109/TIE.2011.2141092
14. Ivanic, M., Mezei, I.: Distance estimation based on RSSI improvements of orientation aware nodes. In: 2018 Zooming Innovation in Consumer Technologies Conference, ZINC 2018, pp. 140–143, August 2018. https://doi.org/10.1109/ZINC.2018.8448660
15. Latham, A., Wood, P.R.: Inhabiting infrastructure: exploring the interactional spaces of urban cycling. Environ Plan A **47**(2), 300–319 (2015). https://doi.org/10.1068/A140049P
16. Lee, J.S., Jiang, J.W.: Enhanced fuzzy-logic-based power-assisted control with user-adaptive systems for human-electric bikes. IET Intell. Transp. Syst. **13**(10), 1492–1498 (2019). https://doi.org/10.1049/IET-ITS.2019.0092
17. Lee, J.S., Jiang, J.W., Sun, Y.H.: Design and simulation of control systems for electric-assist bikes. In: Proceedings of the 2016 IEEE 11th Conference on Industrial Electronics and Applications, ICIEA 2016, pp. 1736–1740, October 2016. https://doi.org/10.1109/ICIEA.2016.7603866
18. Liang, C.Y., Lin, W.H., Chang, B.: Applying fuzzy logic control to an electric bicycle. In: First International Conference on Innovative Computing, Information and Control 2006, ICICIC'06, pp. 513–516 (2006). https://doi.org/10.1109/ICICIC.2006.54
19. Lomonova, E.A., Vandenput, A.J., Rubáček, J., D'Herripon, B., Roovers, G.: Development of an improved electrically assisted bicycle. In: Conference Record - IAS Annual Meeting (IEEE Industry Applications Society), vol. 1, pp. 384–389 (2002). https://doi.org/10.1109/IAS.2002.1044116
20. Masatoku, K.: Regulations on electric bicycles in Japan. In: Proceedings of the Conference of Safety Popularization Electric Assist Bicycles, pp. 1–5 (2001)
21. Perul, J., Renaudin, V.: BIKES: bicycle itinerancy Kalman filter with embedded sensors for challenging urban environment. IEEE Sens. J. **22**(6), 5270–5277 (2022). https://doi.org/10.1109/JSEN.2021.3086004
22. Romanillos, G., Gutiérrez, J.: Cyclists do better. Analyzing urban cycling operating speeds and accessibility. Int. J. Sustain. Transp. **14**(6), 448–464 (2020). https://doi.org/10.1080/15568318.2019.1575493
23. Indulal, S., Chandramohanan Nair, P.S.: A novel approach in automatic control of a Hybrid Bicycle — IET Conference Publication — IEEE Xplore (2007)
24. Tal, I., Ciubotaru, B., Muntean, G.M.: Vehicular-communications-based speed advisory system for electric bicycles. IEEE Trans. Veh. Technol. **65**(6), 4129–4143 (2016). https://doi.org/10.1109/TVT.2015.2442338

Federated Learning



FedGCS: Addressing Class Imbalance in Long-Tail Federated Learning

Guozheng Liu[1,2], Wei Zhang[1,2(✉)], Huiling Shi[1,2], Lizhuang Tan[1,2], Chang Tang[1,2], and Meihong Yang[1,2(✉)]

[1] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China
10431210755@stu.qlu.edu.cn, {wzhang,shihl,tanlzh,yangmh}@sdas.org
[2] Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan, China
tangchang@cug.edu.cn

Abstract. Federated learning is a privacy-preserving distributed machine learning method, which facilitates clients to cooperate in training a shared model while safeguarding original data. The different distribution and quantity of training data between clients can pose significant challenges, such as data heterogeneity and class imbalance, which can greatly influence the performance of the shared model. Although many methods have been proposed to eliminate the deleterious influence of non-IID data, existing solutions usually do not perform well on tail-classes owing to the absence of attention for the long-tail distribution. We present a long-tail federated learning framework FedGCS, which can solve the global and local class imbalance problem via generic to compensate for specific. Specifically, clients separate features from the training data based on the class activation map and selectively fuse the separated class-specific features and class-generic features to restore the distribution of tail-classes. We also design a loss function—TailDistillation Loss to lessen the bias of the classifier towards head-classes. To appraise the effectiveness of FedGCS, we adapted multiple benchmark datasets to the long-tail federated learning setting. Experiments indicate that the FedGCS is an useful method, and is superior to previous approaches.

Keywords: Federated learning · Long-tail learning · non-IID

1 Introduction

Federated learning (FL) makes clients train a shared model together, abstaining from uploading private data to server. It can settle the issue of data security and privacy protection, while reducing resource consumption on the central server. FL has a good application prospect in numerous areas, such as medical care [30],

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 223–238, 2024.
https://doi.org/10.1007/978-3-031-63989-0_11

financial services [38], Internet of Things [23] and network resource scheduling [6,7].

In standard FL, for each iteration, server will arbitrarily choose a subgroup of all participants and distribute current model. These chosen clients will then train the received model using a private dataset and send their gradient adjustments to server for aggregation. Finally, the server will return updated shared model to all participants. But during FL training, a severe reality problem is heterogeneity arising from non-IID data between varied clients [18]. In the FL situation, the data owned by various clients have different sources and may contain their preferences, which leads to distinct distributions and quantities between clients.

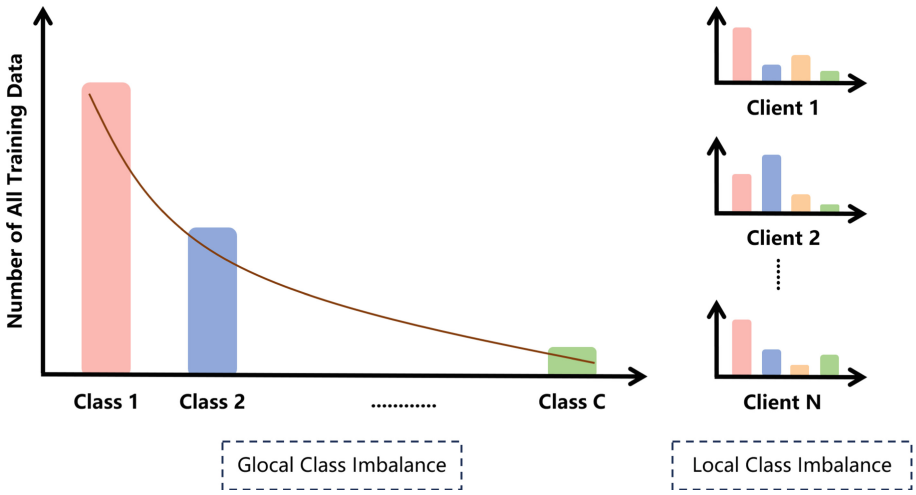


Fig. 1. Global class imbalance and local class imbalance in FL. Research purpose is to alleviate the imbalance problem in training data below the long-tail federated learning setting.

Additionally, actual world data usually exhibit long-tail distribution characterized by severe class imbalance, i.e., the samples for certain classes is much larger than others, for example, the number of patients with different diseases in medical diagnoses is significantly different, and the number of fraudulent transactions in credit card transactions is much lower than normal transactions. In the FL scenarios, class imbalance may exist globally and locally [31], as illustrated in Fig. 1. Among them, global class imbalance means that the amalgamation of clients local data follows a long-tail distribution, and local class imbalance denotes that each class samples on every client is extremely uneven. One radical scenario of local class imbalance is class missing, which means that some classes do not exist, and this problem is likely to occur in those global minority classes. There may also be situations where certain classes are head-classes on some clients and tail-classes on other clients, resulting in data heterogeneity between

clients. When class imbalance exist, the local model may be biased in the process of learning and generalization, which is further inclined towards learning head-classes and neglects tail-classes importance, bringing about the dramatic diminishment in the recognition and classification accuracy of tail-classes in the model. However, in many practical cases, tail-classes may play an important role that far exceeds their proportion in the sample, for example, health monitoring equipment needs to be more sensitive to abnormal conditions than normal [20], and the auto drive system is more important to accurately predict the status of traffic lights than to predict the names of surrounding buildings [12].

Recently, lots of approaches have been presented to address the long-tail issue. One common method is class rebalancing, which aims to decrease the effect of imbalance through resampling or reweighting technique. In addition, although the overall performance can be improved, this type of method usually sacrifices head-classes performance for tail-classes performance, and cannot fundamentally settle the state of lacking data information in the long-tail problem, so this may not be desirable in practical applications. Moreover, most of these methods are based on the premise that the global data distribution is observable, which does not apply to FL where data cannot be exported locally. At present, there are less studies on long-tail federated learning, and most of them do not notice the unified treatment of global class imbalance and local class imbalance.

Within this research, we present a long-tail federated learning framework FedGCS, which can solve the class imbalance problem via generic to compensate for specific. Specifically, first, the confusing class relative to the tail class is obtained locally on each client through similarity between classes. Then, based on the class activation map (CAM) [28,39], the features of the training data are partitioned into class-generic features with little difference between classes and class-specific features with significant difference between classes. Finally, over the FL training phase, the class-specific features of tail-class are combined with the class-generic features of the corresponding confusing classes to obtain novel data features of tail-classes. Using this method, the local model performance on the client can be improved, thereby improving the final aggregated global model performance. Furthermore, when the data present a long-tail distribution, it will cause the classifier to lean more towards identifying head-classes, thus ignoring tail-classes. We introduce TailDistillation Loss, in order that the client local model can further retain the knowledge about tail-classes in shared model over the training phase, decreasing the classifier's inclination for head-classes. We evaluated the proposed FedGCS on CIFAR-10/100-LT [16]. A considerable amount of experiments indicate that FedGCS outperforms the existing FL approaches in image classification tasks with the long-tail distribution. Subsequently, we provide an overview of the contributions made in this study.

- We present a long-tail FL framework FedGCS, which does not need to know the information of global data distribution, enhances tail-classes features in client, and we introduce TailDistillation Loss to correct the model during training phase.
- We perform comprehensive experiments on two long-tail datasets. Our method is proved to be better than other FL Approaches in long-tail distribution image classification tasks.
- Our work demonstrates the importance of recognizing the long-tail distribution as early as possible and taking timely measures during FL.

2 Related Work

2.1 Federated Learning

FL was presented in [24], including FedAvg, the most basic and classic algorithm, which obtains the aggregation model by iterating and averaging parameters from the distributed client model, thus avoiding the need to access the client local data. For the data heterogeneity in FL, there has been a lot of work. On the one side, local training is stabilized by fine-tuning the disparity between models for client and server [15,19,32]. On the other side, it aims to optimize the effectiveness of model aggregation [2,34,35]. Among them, knowledge distillation [9] has become an effective solution [11,17,21], enriching the shared model through integrating local model information and alleviating model drift problem caused by heterogeneity. Although the above methods have solved to some extent the problem of data heterogeneity, the final model obtained exhibits unsatisfactory performance on tail-classes, attributed to overlooking the global distribution.

At present, only few studies have recognized the issue of decreased model accuracy resulting from class imbalance (not specific to long-tail distribution). Duan et al. [4] proposed the Astraea framework, which mitigates class imbalance through information augmentation according to global information distribution and multi-client rescheduling on basis of intermediaries. Shang et al. [29] presented an approach of applying federated features for classifier retraining to solve the union issue of data heterogeneity and long-tail distribution. Wang et al. [36] introduced a monitoring solution capable of deducing the constitution of training samples in every FL iteration, aimed at alleviating the influence of global class imbalance. In our scenario, local data presents a long-tail distribution, where some important classes having only small or even no samples. Therefore, it is necessary to design a new algorithm specifically for this unbalanced property to address this problem.

2.2 Long-Tail Learning

In the actual world, training samples always exhibit severe class imbalance, known as long-tail distribution, which has attracted widespread attention in the field of deep learning [37]. The objective of long-tail learning is to develop

high-performance models on the data with long-tail distribution. Currently, some solutions have been proposed, including class rebalancing, information augmentation, and module improvement. The methods based on class rebalancing aim to alleviate the impact of class imbalance through resampling or reweighting technique. Feng et al. [5] suggested monitoring model training across various classes by averaging classification prediction scores. Ren et al. [25] presented to utilize the training label oftenness to reweight prediction logits, in order that the label can mitigate the bias of class imbalance before calculating final losses. The methods based on information augmentation are designed to enhance the model performance of long-tail learning though introducing additional information. Chu et al. [3] proposed to enhance tail-classes with few samples by learning features from head-classes with ample samples. Liu et al. [22] proposed using the geometric information of classifiers with relatively large head-classes to enhance classifier weight assigned to tail-classes, thereby improving tail-classes performance. The methods based on module improvement aim to tackle long-tail problems though ameliorating network modules. Kang et al. [13] introduced an innovative decoupled training approach, to gain a more stable feature space, resulting in improved performance in long-tail learning. Zhou et al. [40] proposed a novel strategy to address long-tail recognition. In detail, traditional studying branch employs evenly extraction to mimic a long-tail distribution, whereas re-balancing branch employs a reverse extractor to augment the sampling of tail-classes. But most of long-tail learning methods need information on global class distribution, which is difficult to apply directly to FL.

3 Proposed Method

3.1 Problem Formulation

Problem Setting. We discuss a standard FL setup where K clients each hold a potential non-IID dataset $\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^K$, and the union of all these datasets $\mathcal{D} = \cup_k \mathcal{D}^k$ follows a long-tail distribution, i.e. $D \in (\mathcal{X}, \mathcal{Y})$ and $\mathcal{Y} \in \{1, \ldots, C\}$. Setting n_c^k be the quantity of samples for class c on client k, and $n_c = \sum_{k=1}^{K} n_c^k$. Wherein all classes are ordered in a diminishing manner based on respective sample numbers, that is, if $c_1 < c_2$, there will be $n_{c_1} \geq n_{c_2}$. We introduce $r \in (0, 1)$, representing the proportion of samples from head-classes to overall sample numbers. Additionally, we specify the global imbalance factor as the quotient of the sample count in the largest class divided by the sample count in the smallest class, i.e., IF$= \frac{n_1}{n_C}$. We also use the Dirichlet distribution factor α to regulate heterogeneity extent. The model we use in FL is a neural network $\phi_{\mathbf{w}}$ with parameter $\mathbf{w} = \{\mathbf{u}, \mathbf{v}\}$, which comprises a feature extractor $f_{\mathbf{u}}$ and a classifier $h_{\mathbf{v}}$. The parameter representation of the local model for client k is $\mathbf{w}_k$.

The FedAvg Baseline Algorithm. FedAvg [24] is the first and most classic FL algorithm. In the t-th round of communication, Initially server arbitrarily chooses a subgroup S^t from all clients and sends a shared model $\mathbf{w}^t$ to them. Then the client uses local data $\mathcal{D}^k, k = 1, \ldots, K$ to perform iterative optimization with learning rate of η to further adjust the model:

$$\mathbf{w}_k^{t+1} \leftarrow \mathbf{w}_k^t - \eta \nabla_{\mathbf{w}} \ell \left(\mathbf{w}^t; \mathcal{D}^k \right). \tag{1}$$

After local updating, clients uploads their updated models to server. At last, server aggregates models through weighted averaging to obtain shared model for the $t+1$ communication round:

$$\mathbf{w}^{t+1} = \sum_{k \in \mathcal{S}^t} \frac{\left| \mathcal{D}^k \right|}{\sum_{k \in S^t} \left| \mathcal{D}^k \right|} \mathbf{w}_k^{t+1}. \tag{2}$$

3.2 Framework of FedGCS

Based on the above assumptions, we propose FedGCS to solve the global and local long-tail distribution problem in FL. It is a two-stage training scheme based on FedAvg. In the stage one, clients jointly train a shared model according to standard FL training steps. In the stage two, each client performs feature separation on the training data locally based on the CAM and selectively fuses the separated class-specific features and class-generic features to generate augmented samples of tail-classes. On this basis, the local classifier is fine-tuned and uploaded to server to further update shared model. Our goal is to enrich data multiplicity and prevent overfitting through feature-level information augmentation. In addition, in this scheme, we also introduce TailDistillation Loss, which allows the client local model to further preserve the information about tail-classes in the shared model, to reduce classifier bias towards head-classes. The framework of FedGCS is depicted in Fig. 2.

Specifically, in the Phase-I, we use images from local dataset of the client to train the model issued by server, and upload updated gradient to server for aggregation. After multiple iterations, we obtain a global model. To subsequently compute the CAM, we select a network structure (ResNet [8]) with a single fully connected layer. Numerous deep convolutional neural networks belong to this category, such as DenseNet [10], MobileNet [26] , and EfficientNet [33].

In the Phase-II, the problem we need to solve is, how to select classes for tail-classes that can extract generic features from them. We believe that classes that are relatively near to specified tail class in the distribution, meaning they are more likely to be confused with it, which has a significant impact on restoring the distribution of tail-classes. Specifically, every client use the shared model obtained in the Phase-I to predict classification scores for all classes, excluding the specified tail class, for each training sample, and by sorting the average classification scores, find its confusing classes C_f.

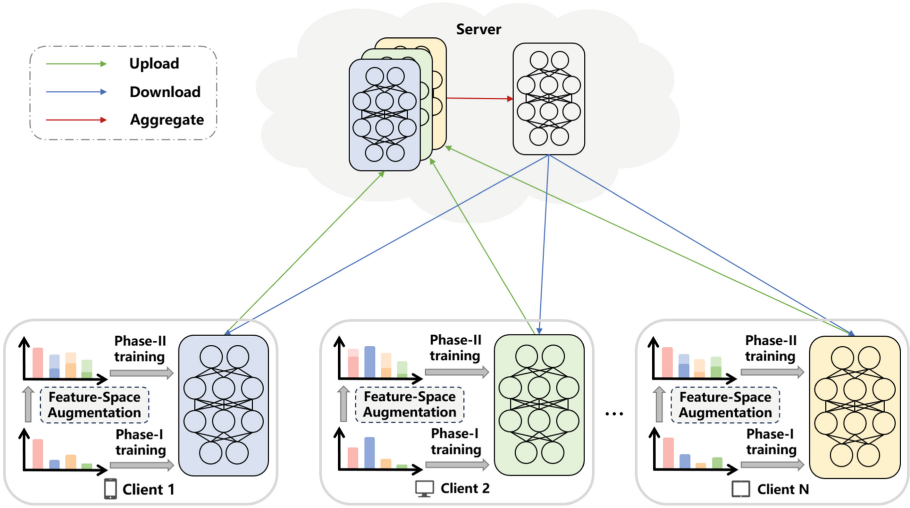


Fig. 2. The framework of FedGCS.

We also need to segment the features of the training data in each client, distinguishing between class-specific and class-generic features that can be utilized to restore the tail class distribution. Motivated by current work on attention and visual interpretation [28,39], we first calculate the score y_c of class c regarding feature activation map A^l gradient of the convolution layer, i.e. $\frac{\partial y^c}{\partial A^l}$. Subsequently, we carry out global avgpooling on backflow gradients, bringing about neuron importance weights α_l^c:

$$\alpha_l^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^l}. \tag{3}$$

Then, we weight the forward activation maps, and subsequently apply a ReLU to get the CAM M_c of class c:

$$M_c = \mathrm{ReLU}\left(\sum_l \alpha_l^c A^l \right). \tag{4}$$

Finally, specifying a threshold of $0 < \xi < 1$. We can disintegrate M_c into M_c^s and M_c^g, and get class-specific and class-generic features, this formula is shown below:

$$M_c^s = \mathrm{sgn}(M_c - \xi) \odot M_c, \tag{5}$$

$$M_c^g = \mathrm{sgn}(\xi - M_c) \odot M_c, \tag{6}$$

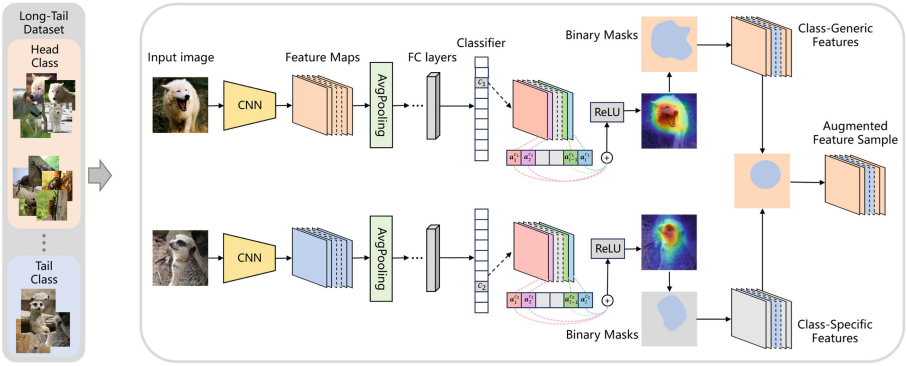


Fig. 3. The procedure of feature enhancement on client. Fusing the class-specific features of tail-classes images with the class-generic features of the corresponding confusing classes images in feature space to produce augmented samples of tail-class.

Next, we use the class-specific features separated from tail-class images and the class-generic features separated from the corresponding confusing class images for feature fusion. To mitigate noise and potential bias, we employ a linear fusion approach in the feature space, guiding the fusion process by randomly generating combination ratios to produce enhanced samples for tail-class, restoring its inherent data distribution, as shown in Fig. 3. For per sample in tail-class, altogether produce N_a augmented samples. Finally, the generated augmented samples are used locally on each client to fine-tune the classifier and uploaded to server to further update shared model.

In addition, we also propose to decrease classifier bias toward head-classes by introducing TailDistillation Loss (denoted by L_{TD}). Specifically, we consider using the CrossEntropy Loss function, which is widely used in multi-class classifiers, as the basic term to define L_{TD} as:

$$\begin{aligned} L_{TD} = (1-\lambda) * \mathrm{CE}\left(gt, \sigma\left(z_s; T=1\right)\right) \\ +\lambda * \mathrm{CE}\left(\sigma\left(z_t'; T=\tau\right), \sigma\left(z_s'; T=\tau\right)\right), \end{aligned} \tag{7}$$

where gt is the ground truth label (one-hot). z_s represents local model's predicted result on local data, z_t' represents the global model's predicted result on local tail-classes data, and z_s' represents the local model's predicted result on local tail-classes data. σ is the softmax function parameterized by the temperature T, and λ is a balancing hyperparameter, which ranges from $[0, 1]$.

4 Experiment

4.1 Experiment Settings

We choose two baseline datasets: CIFAR-10 and CIFAR-100 [16], and according to [1], shape them into long-tail versions (CIFAR-10/100-LT) with varied IFs (10, 50, and 100), to fit into long-tail federated learning setting. Moreover, utilizing Dirichlet distribution to partition the non-IID data between different clients and set the value of α to 0.5. We use ResNet-8, which containing a single fully connected layer. To guarantee an equitable comparison, we have achieved all FL methods for comparison applying same model. By default, we run a total of 200 rounds of global communication with a total of 20 clients, possessing an active user percentage of $C = 50\%$ per round. For per sample in client tail-class, we choose $N_a = 3$. We set the local training batch size to 128, learning rate to 0.01, and the optimizer to SGD.

Table 1. Test accuracy (%) for different FL methods and FedGCS on CIFAR-10/100-LT with diverse IFs.

Family	Method	CIFAR-10-LT			CIFAR-100-LT		
		IF = 10	IF = 50	IF = 100	IF = 10	IF = 50	IF = 100
Heterogeneity-oriented FL methods	FedAvg	59.53	54.95	53.32	38.62	28.47	25.16
	FedProx	60.22	56.43	53.45	38.93	28.52	25.64
	FedNova	61.45	56.79	55.62	39.31	28.19	26.22
Imbalance-oriented FL methods	Fed-Focal Loss	59.39	53.67	50.32	35.17	26.74	24.58
	Ratio Loss	60.96	58.26	55.93	39.45	29.68	27.03
	FedAvg+τ–norm	51.12	48.81	46.67	31.02	23.09	20.32
Proposed Method	FedGCS	**64.57**	**62.32**	**61.75**	**40.83**	**33.54**	**31.76**

4.2 Comparison with Other Methods

To validate the efficaciousness of FedGCS, we compare the proposed method with these heterogeneity-oriented FL methods: FedAvg [24], FedProx [19] and FedNova [35]. In addition, we compare imbalance-oriented FL methods: Fed-Focal Loss [27], Ratio Loss [36], and FedAvg with τ-norm [14].

We evaluate FedGCS on CIFAR-10/100-LT, as depicted in Table 1. FedGCS achieved the optimal test accuracy in all scenarios. Compared to the baseline FedAvg, when IF = 100, FedGCS has the highest performance gain, approximately 8.4% for CIFAR-10-LT and 6.6% for CIFAR-100-LT. It implies that FedGCS has excellent generalization ability. For heterogeneous-oriented FL methods, such as FedProx, exhibit performance similar to FedAvg, as they primarily address non-IID data while disregarding the long-tail distribution. For class imbalance-oriented FL methods, such as Fed-Focal Loss, perform well compared to FedAvg in certain situations. However, when compared to FedGCS, there remains a performance gap as these methods only address global class imbalance concerns and overlook data heterogeneity issues.

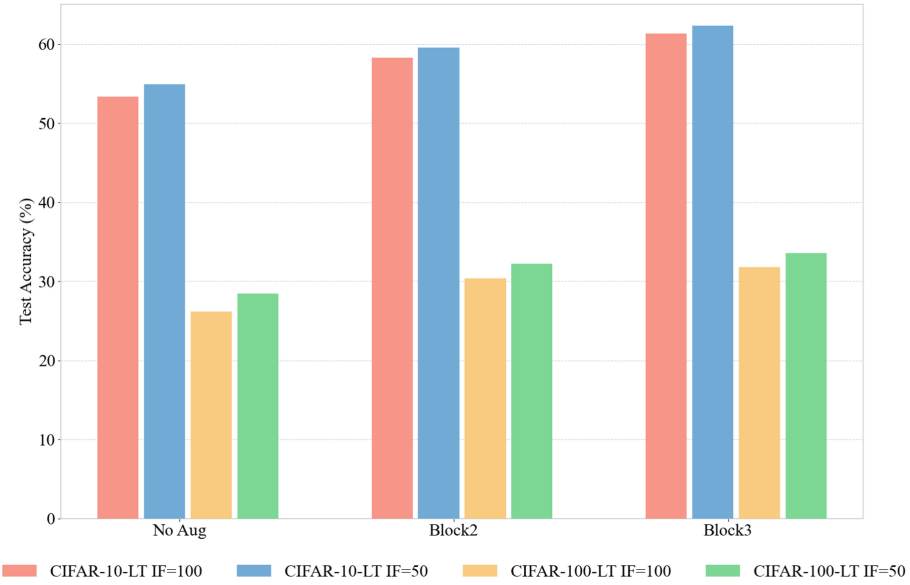


Fig. 4. Test accuracy though applying feature augmentation at diverse depths of ResNet-8.

4.3 Model Validation

Ablation Analysis on Applying Feature Augmentation at Diverse Depths of Network. The ResNet-8 model used for the CIFAR-10/100-LT dataset composes of three convolutional blocks. We perform feature enhancement in the model, subsequent to the final two convolutional blocks. After enhancing the features after Block2 and Block3 and generating augmented samples, the test accuracy after fine-tuning in the Phase-II is shown in Fig. 4. It is obvious that among different datasets and imbalanced factors, feature enhancement after Block3 has the best effect. Feature maps nearer to the input end encompass greater spatial details, but they may introduce extra artifacts. To address this issue, we pass the features generated by Block3 directly to the global pooling layer, removing noise caused by augmentation. Therefore, we choose to apply feature enhancement after Block3. We evaluate the performance without the application of enhancement, referred to as "No Aug" in the figure.

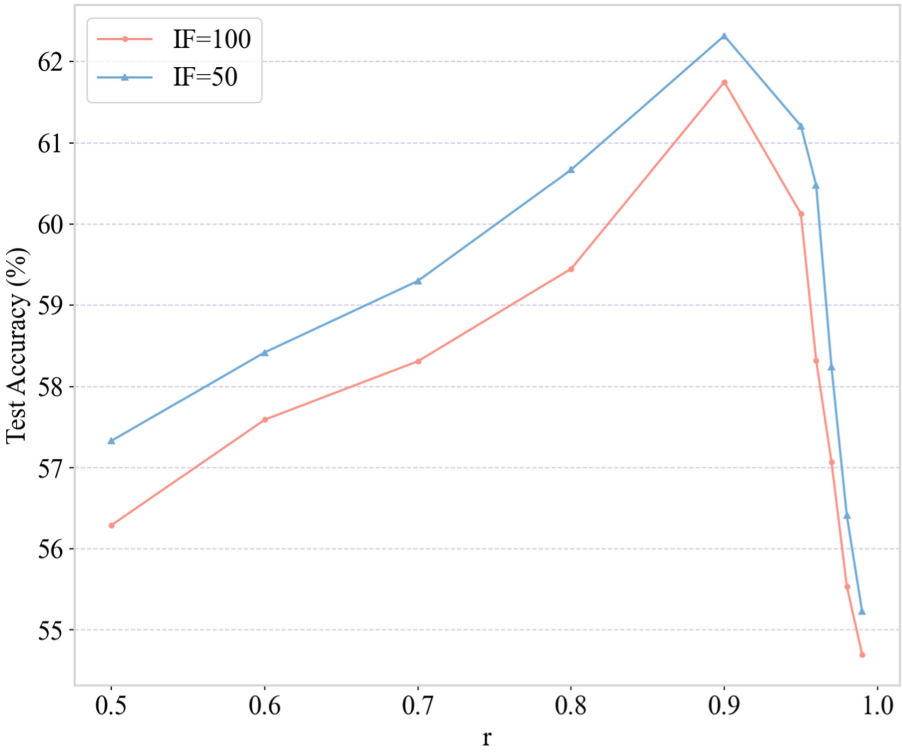


Fig. 5. FedGCS performance across varied r values on CIFAR-10-LT.

Analysis of Hyperparameter Sensitivity. We assess the consequence of choosing the varied values of r on the final classification accuracy in Fig. 5. The curves exhibit peaks near $r = 0.9$. On the ascending side of the peak, only less classes are utilized as head-classes, making it insufficient to capture class-generic features. On the descending side of the peak, only less classes are employed as tail-classes, leaving classes with insufficient samples unable to use augmented samples for fine-tuning. Moreover, we select the smallest r value that meets the condition in every experiment. We also study the impact of the equilibrium factor λ in the loss function. This hyperparameter controls the distillation intensity of the tail class in Eq. 7. In Fig. 6, it is apparent that, the greater the λ value, the intensified the model accuracy degradation. This shows that distillation of tail-classes data alone is not sufficient to obtain an excellent model.

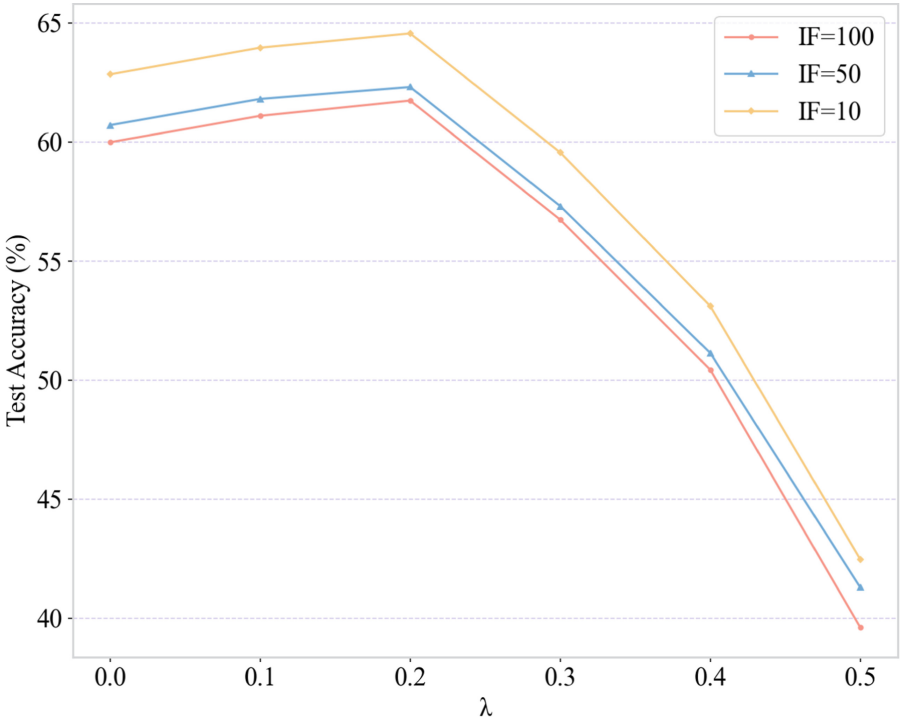


Fig. 6. FedGCS performance across varied λ values on CIFAR-10-LT.

Impact of the Extent of Non-IIDness. Fig. 7 additionally illustrates the test accuracy of the three methods across varied extents of non-IIDness. It is observable that the performance of every method declines with increasing non-IIDness extent. When the α value decreases from 1 to 0.01, the compared methods exhibit a greater decrease in performance compared to FedGCS.

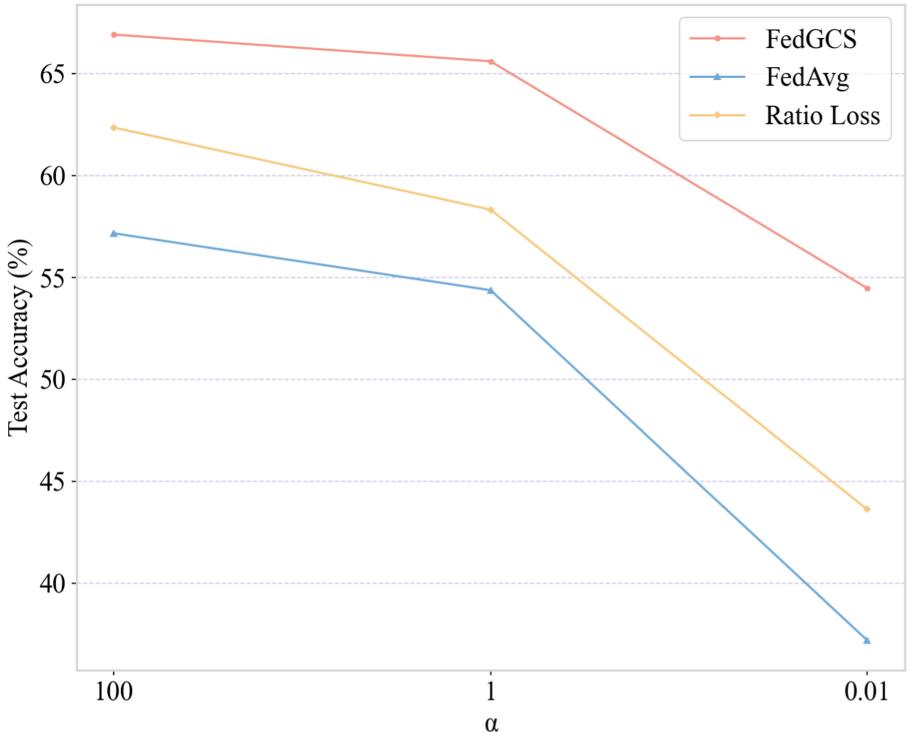


Fig. 7. FedGCS performance across varied α values on CIFAR-10-LT with IF=100.

5 Conclusion

In this study, we present FedGCS to deal with the global and local class imbalance problem in long-tail FL. On the client, the corresponding confusing classes for local tail-classes are computed on basis of global model. Then, feature separation is performed on the local data using the class activation map, and the class-specific features of the tail-classes are integrated with the class-generic features of the corresponding confusing classes to restore tail-classes distribution. Furthermore, we construct TailDistillation Loss to mitigate the effects of global and local class imbalance. Numerous experiments demonstrate that FedGCS outperforms the current heterogeneity-oriented and imbalance-oriented FL approaches significantly when dealing with a long-tail distribution.

Acknowledgements. This work was supported in part by the Shandong Provincial Natural Science Foundation under Grant No. ZR2020LZH010 and No. ZR2022 LZH015, the Project of Key R&D Program of Shandong Province under Grant No. 2022CXGC20106, the Pilot International Cooperation Project for Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) under Grant No. 2022JBZ01-01 and No. 2022GH 007, the One Belt One Road Innovative Talent Exchange with Foreign Experts under Grant No.

DL2022024004L, and the Jinan Scientific Research Leader Studio Project under Grant No. 2021GXRC091. Wei Zhang and Meihong Yang are the corresponding authors.

References

1. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. Adv. Neural Inf. Process. Syst. **32** (2019)
2. Chen, H.Y., Chao, W.L.: Fedbe: making bayesian model ensemble applicable to federated learning. arXiv preprint arXiv:2009.01974 (2020)
3. Chu, P., Bian, X., Liu, S., Ling, H.: Feature space augmentation for long-tailed data. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12374, pp. 694–710. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58526-6_41
4. Duan, M., Liu, D., Chen, X., Liu, R., Tan, Y., Liang, L.: Self-balancing federated learning with global imbalanced data in mobile systems. IEEE Trans. Parallel Distrib. Syst. **32**(1), 59–71 (2020)
5. Feng, C., Zhong, Y., Huang, W.: Exploring classification equilibrium in long-tailed object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3417–3426 (2021)
6. Hao, H., Xu, C., Zhang, W., Yang, S., Muntean, G.M.: Computing offloading with fairness guarantee: a deep reinforcement learning method. IEEE Trans. Circ. Syst. Video Technol. **33**, 6117–6130 (2023)
7. Hao, H., Xu, C., Zhong, L., Muntean, G.M.: A multi-update deep reinforcement learning algorithm for edge computing service offloading. In: Proceedings of the 28th ACM International Conference on Multimedia, pp. 3256–3264 (2020)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
10. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
11. Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., Kim, S.L.: Communication-efficient on-device machine learning: federated distillation and augmentation under non-iid private data. arXiv preprint arXiv:1811.11479 (2018)
12. Jiang, C.M., Najibi, M., Qi, C.R., Zhou, Y., Anguelov, D.: Improving the intra-class long-tail in 3d detection via rare example mining. In: Avidan, S., Brostow, G., Cisse, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13670D, pp. 158–175. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-20080-9_10
13. Kang, B., Li, Y., Xie, S., Yuan, Z., Feng, J.: Exploring balanced feature spaces for representation learning. In: International Conference on Learning Representations (2020)
14. Kang, Bet al.: Decoupling representation and classifier for long-tailed recognition. arXiv preprint arXiv:1910.09217 (2019)
15. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143. PMLR (2020)

16. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
17. Li, D., Wang, J.: Fedmd: heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019)
18. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Signal Process. Mag. **37**(3), 50–60 (2020)
19. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
20. Li, Y., Wang, T., Kang, B., Tang, S., Wang, C., Li, J., Feng, J.: Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10991–11000 (2020)
21. Lin, T., Kong, L., Stich, S.U., Jaggi, M.: Ensemble distillation for robust model fusion in federated learning. Adv. Neural. Inf. Process. Syst. **33**, 2351–2363 (2020)
22. Liu, B., Li, H., Kang, H., Hua, G., Vasconcelos, N.: Gistnet: a geometric structure transfer network for long-tailed recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8209–8218 (2021)
23. Liu, T., Xia, J., Ling, Z., Fu, X., Yu, S., Chen, M.: Efficient federated learning for aiot applications using knowledge distillation. IEEE Internet Things J. (2022)
24. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
25. Ren, J., Yu, C., Ma, X., Zhao, H., Yi, S., et al.: Balanced meta-softmax for long-tailed visual recognition. Adv. Neural. Inf. Process. Syst. **33**, 4175–4186 (2020)
26. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
27. Sarkar, D., Narang, A., Rai, S.: Fed-focal loss for imbalanced data classification in federated learning. arXiv preprint arXiv:2011.06283 (2020)
28. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)
29. Shang, X., Lu, Y., Huang, G., Wang, H.: Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. arXiv preprint arXiv:2204.13399 (2022)
30. Sheller, M.J., et al.: Federated learning in medicine facilitating multi-institutional collaborations without sharing patient data. Sci. Rep. **10**(1), 1–12 (2020)
31. Shuai, X., Shen, Y., Jiang, S., Zhao, Z., Yan, Z., Xing, G.: Balancefl: addressing class imbalance in long-tail federated learning. In: 2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 271–284. IEEE (2022)
32. Dinh, T.C., Tran, N., Nguyen, J.: Personalized federated learning with moreau envelopes. Adv. Neural Inf. Process. Syst. **33**, 21394–21405 (2020)
33. Tan, M., Le, Q.: Efficientnet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114. PMLR (2019)
34. Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y.: Federated learning with matched averaging. arXiv preprint arXiv:2002.06440 (2020)

35. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the objective inconsistency problem in heterogeneous federated optimization. Adv. Neural. Inf. Process. Syst. **33**, 7611–7623 (2020)
36. Wang, L., Xu, S., Wang, X., Zhu, Q.: Addressing class imbalance in federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 10165–10173 (2021)
37. Zhang, Y., Kang, B., Hooi, B., Yan, S., Feng, J.: Deep long-tailed learning: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **45**, 10795–10816 (2023)
38. Zheng, W., Yan, L., Gou, C., Wang, F.Y.: Federated meta-learning for fraudulent credit card detection. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 4654–4660 (2021)
39. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929 (2016)
40. Zhou, B., Cui, Q., Wei, X.S., Chen, Z.M.: BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9719–9728 (2020)



FedRC: Representational Consistency Guided Model Uploading Mechanism for Asynchronous Federated Learning

Sheng Liu[1], Linlin You[1(✉)], and Yuren Zhou[2]

[1] School of Intelligent Systems Engineering, Sun Yat-Sen University, Shenzhen, China
liush235@mail2.sysu.edu.cn, youllin@mail.sysu.edu.cn
[2] Engineering Product Development, Singapore University of Technology and Design, Singapore, Singapore

Abstract. Recently, a novel distributed machine learning paradigm called Federated learning (FL) has caught the eyes of both academics and industries, as it can orchestrate substantial Internet of Things (IoT) devices as clients to learn a global model collaboratively and efficiently without sharing sensitive data. Moreover, while comparing the two modes of FL, i.e., synchronous FL (SFL) and asynchronous FL (AFL), AFL is more scalable and flexible to address the issue of over-fitting and the performance bottleneck caused by the stragglers. However, the data heterogeneity and high communication consumption issues faced by AFL still hamper its further applications and deployments in ubiquitous IoT. Motivated by this, we propose FedRC, a model uploading mechanism for AFL, guided by Representational Consistency (RC). As a layer-wise uploading method for Deep Neural Networks (DNNs), FedRC calculates simplified Representational Dissimilarity Vectors (RDVs) for each local layer and corresponding global layer, respectively, after the local training of each client, and then measures RCs based on the two RDVs to adaptively determine the uploading of model layers. According to the evaluation based on three standard datasets, compared with four state-of-the-art baselines (i.e., FedAvg, FedProx, FedAsync, and PartialNet), FedRC can boost model accuracy by 3.48%, save communication costs by 26.79%, and shorten transmission time by 44.14%, respectively.

Keywords: Federated Learning · Asynchronous Federated Learning · Deep Neural Networks (DNNs) · Model Uploading

1 Introduction

With the rapid development of IoT (Internet of Things), significant amounts of data are generated from massive smart devices such as Unmanned Aerial Vehicles, Intelligent Connected Vehicles, and medical robots [36,39]. Traditionally, these large amounts of data are transmitted to the data centers for utilization. However, this centralized

This research was funded by the National Natural Science Foundation of China (62002398) and the Collaborative Innovation Center for Transportation of Guangzhou (202206010056).

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 239–256, 2024.
https://doi.org/10.1007/978-3-031-63989-0_12

pattern is facing three main challenges, including large communication costs, privacy protection difficulties, and unbalanced computation loads between the center and edge devices.

In this context, a novel distributed learning paradigm, called Federated Learning (FL) [21], starts to attract more attention in both academic and industrial communities. In general, a FL system consists of one server and multiple clients to train a global model collaboratively and iteratively through multiple learning rounds. In each communication round, clients first receive the latest global model from the server, then perform local training based on their own datasets. Lastly, local model updates are uploaded to the server for aggregation to update the global model.

Since raw data are unexposed and edge resources are utilized, FL is more suitable to support ubiquitous IoT [16,40], and has been widely applied to various data-sensitive domains, e.g., to achieve the early-stage detection of dementia disease by sensors in smart homes [13], to perform image process and object detection by RSUs (Road-Side Units) in smart transportation [20], to identify suspicious transactions by banks in smart fintech [17], to recognize human activities by mobile phones in smart lives [5], etc.

Furthermore, according to the collaboration type of clients, FL can be divided into synchronous FL (SFL) and asynchronous FL (AFL) [1,38]. In SFL, a randomly selected or pre-selected subset of clients will participate in a certain round, and the server will execute the global aggregation process after all the participants have uploaded their local models successfully [21]. However, SFL has to wait for the stragglers (who may fail the training or communication process due to power-off or poor connection) in every round, which impedes the actual deployment of SFL. On the contrary, in AFL, clients can work individually and asynchronously, and the server will update the global model immediately once the pre-defined condition is satisfied (such as the maximum waiting time is reached or a certain number of updates are received) without unnecessary waiting [10,34].

Although AFL can improve scalability compared with SFL, it still faces two urgent challenges that affect the Quality of Service (QoS), namely:

- **Data Heterogeneity:** Since the sharing of raw data among clients is unallowed, local models of each client are trained and uploaded separately according to data collected with specific user preferences, which results in an intrinsic issue faced by AFL, i.e., strongly Non-IID (Non-Independent and Identically Distributed) data. Moreover, the staleness model caused by the asynchronous mode exacerbates the heterogeneous problem. Such an issue can degrade the learning performance of AFL significantly, as the representation of trained models can be partial [28].
- **Communication Resource Constraints:** First, the communication resources of edge clients are limited (restricted by hardware and software configurations) and may change over time (such as shifting between 5G and Wi-Fi). Second, each client needs to interact with the AFL server iteratively, which may become intrusive for other service co-running at the edge. The uncontrolled or unrestricted utilization of communication resources may easily become the bottleneck of IoT for the actual deployment. Finally, the communication burden of the AFL server can be large, since all clients are interacting with the AFL server frequently. Hence, training a global model with light communication costs is of significance to the development of AFL [7].

Accordingly, several solutions have been proposed to tackle these two challenges, such as adopting informative aggregation strategies to solve the data heterogeneity issue [30,37] and employing compression methods to reduce communication costs [18]. Specifically, to support the training of DNNs (Deep Neural Networks), a novel idea that can resolve the two issues simultaneously is to upload deep and shallow layers separately, as shallow layers contain fewer parameters while extracting more general features compared with deep layers [3,15,37]. However, currently, the fixed uploading frequency of different layers can not quantize the data heterogeneity level dynamically, and may also deteriorate the overall model performance. Such that, an adaptive mechanism, which measures the layer heterogeneity level caused by data heterogeneity for each client and only uploads those "fine" layers in each round to improve learning performance as well as save communication resources, is still missing.

To fill this gap, this paper proposes FedRC, a model uploading mechanism for AFL, guided by Representational Consistency (RC). To measure the layer heterogeneity adaptively, for each global layer and corresponding local layer, a simplified Representational Dissimilarity Vector (RDV) is calculated based on stimuli and pairwise distance functions after the local training and before the local model uploading. Then, the RC is computed according to the two RDVs, and utilized as the probability of uploading the local layer. Finally, after local layers are uploaded, each layer is aggregated at the server separately to form the layer-wise global model.

Accordingly, the main contributions of this paper are summarized as follows.

- This paper proposes a novel layer-wise uploading mechanism based on RC, called FedRC, to jointly address data heterogeneity and communication resource constraints issues faced by AFL.
- This paper designs a simplified version of RDM for the calculation of RC, named as RDV, to reduce the computation complexity significantly, and then, utilizes RC as the uploading probability of each layer in a round, to avoid the additional training of threshold.
- This paper evaluates FedRC based on three standard datasets (i.e., MNIST, FMNIST, and CIFAR-10), and compared with four state-of-the-art baselines (i.e., FedAvg, FedProx, FedAsync, and PartialNet), FedRC can improve model accuracy by 3.48%, reduce communication costs by 26.79%, and shorten transmission time by 44.14% simultaneously and respectively.

The remainder of this paper is organized as follows. First, Sect. 2 summarizes related work about AFL, model uploading methods, and similarity-based FL. Second, FedRC is presented and evaluated in Sect. 3 and Sect. 4, respectively. Finally, Sect. 5 concludes the work and sketches the future research directions.

2 Related Work

In this section, first, AFL is introduced briefly as the background; then current model uploading strategies for communication cost reduction are summarized; finally, the development of similarity in FL is reviewed. Note that the key related work is summarized in Table 1.

Table 1. The summary of key related work.

Group	Resolved Issue	Reference	Main contributions
Model Uploading	Update compression	[6]	Compressing gradients periodically and using local gradient tracking
		[35]	Introducing a self-learning quantization factor
		[29]	Combining NOMA with adaptive gradient quantization
		[26]	Presenting a Kashin compression method
	Topology optimization	[2]	Bridging SFL and AFL through tiers
		[9]	Introducing a multi-layer hybrid learning framework called fog learning
	Model split	[8]	Utilizing ordered dropout to extract lower footprint sub-models of DNNs
		[15]	Uploading the deep and shallow layers of DNNs with different frequencies
Similarity Analysis	Clustering	[27]	Grouping clients with similar data distributions by multitask learning
		[23]	Using pairwise client gradients to compute similarity for clustering
	RSA	[22]	Introducing RC to investigate the individual differences among DNNs
		[15]	Employing RC as adaptive weights to enhance global aggregation for AFL

2.1 Asynchronous Federated Learning

FedAsync is the first and the canonical AFL algorithm [34]. In FedAsync, the server will update the global model with a hyper-parameter related to the staleness in a weighted aggregation way after receiving a single local model from any client. ASO-Fed improves FedAsync by feature representation and dynamic learning rate, and supports online learning with continuous streaming local data [4].

However, such fully asynchronous methods have an unsteady learning process with model accuracy degraded, as the participant frequency and staleness among clients exit significant differences. Besides, significant transmission costs are also required because of the frequent interaction between clients and the server. Such that, FedSA [19] and SAFA [33] are proposed, which allow multiple clients instead of one client to participate in a global round and optimize the parameters of the learning process.

2.2 Model Uploading Methods

In general, the model uploading methods for communication cost reduction can be summarized in three ways, i.e., update compression, topology optimization, and model split, as illustrated in Table 1 "Model Uploading" group.

First, update compression refers to encoding and uploading updates by a smaller number of bits. Specifically, Haddadpour et al. [6] proposed FedCOMGATE: a federated averaging algorithm with compression and local gradient tracking for the heterogeneous setting; Xu et al. [35] designed FTTQ, which introduced a self-learning quantization factor to reduce the transmission of redundant parameters; Sun et al. [29] utilized adaptive gradient quantization and sparsification to facilitate the uploading scheme; Safaryan et al. [26] proposed Kashin compression: an unbiased compression method inspired by the Kashin representation of vectors.

Table 2. The summary of notations used in this paper.

Notation	Meaning
K	The total number of clients
K_t	The number of participants in the t^{th} round
w_k	The local model of client k
D_k	The local dataset of client k
$\|D_k\|$	The local data size of client k
w_t	The global model in the t^{th} round
$F_k(w_k)$	The local loss function of client k
η	The learning rate
L	The total number of layers
w_k^l	The l^{th} layer of local model w_k
K_t^l	The number of uploaded l^{th} layer in the t^{th} round
M	The stimulus set
h	The total number of stimuli
o_i^l	The output vector of the l^{th} layer on the stimulus i
P^l	The output dimension of the l^{th} layer
E	The dimension of RDV

Second, as for topology optimization, the novel idea is to design a hierarchical communication network, in which, each device only connects to a sub-center for relay uploading, and the server interacts with sub-centers periodically. E.g., Chai et al. [2] presented a tier-based mechanism named FedAT, which adopted a three-layer communication structure bridging SFL and AFL; and Hosseinalipour et al. [9] advocated fog learning as a multi-layer hybrid learning framework to achieve efficient communication based on flexible topology.

Finally, regarding the model split, the model parameters are split by various strategies for communication cost savings. E.g, Horvath et al. [8] proposed ordered dropout, which allowed each client to only train and upload a submodel of original DNNs; Chen et al. [3] and Liu et al. [15] both split the uploading schemes of deep and shallow layers of DNNs; and Wu et al. [32] presented FedKD based on adaptive mutual knowledge distillation, in which, a mentee model and a mentor model were learned from each other but only the parameters of mentee model were uploaded by clients.

2.3 Similarity Analysis in FL

As listed in Table 1 "Similarity Analysis" Group, it has been widely discussed in FL to discover divergent clients or boost the convergence speed [31]. Specifically, most related researches focus on the cluster of clients using different similarity measures, e.g., Sattler et al. [27] introduced clustered FL, which grouped clients with similar data

Table 3. The summary of abbreviations in this paper.

Abbr	Description
IoT	Internet of Things
FL	Federated Learning
SFL	Synchronous Federated Learning
AFL	Asynchronous Federated Learning
DNNs	Deep Neural Networks
QoS	Quality of Service
RSA	Representational Similarity Analysis
RDM	Representational Dissimilarity Matrix
RDV	Representational Dissimilarity Vector
RC	Representational Consistency
Non-IID	Non-Independent and Identically Distributed

distributions in multitask learning; and Palihawadana et al. presented FedSim [23], which clustered clients with similar gradients and utilized a metric called Privacy-preserving Non-IID Index to discover the relationship between similarity and Non-IID.

However, the Representational Similarity Analysis (RSA) [11,12,22], which can characterize the inner stimulus representation of DNNs, has not yet been explored in FL for the optimization of uploading. Even though Liu et al. [15] proposed Fed2A, which introduced RSA into AFL and computed the representational consistency as the adaptive weight in the global aggregation process, RSA is still not used in the model uploading process.

In general, AFL is more suitable for ubiquitous IoT because of its flexibility and scalability. However, the high communication consumptions impede its further application. Accordingly, update compression, topology optimization, and model split methods are widely studied to save communication resources. Whereas, side-effects such as performance dropping are also observed, and similarity-based methods,e.g., RSA, still need further exploration for the layer-wise uploading of DNNs, which are the key purposes of FedRC.

3 Methodology

As shown in Fig. 1, FedRC is proposed to optimize the uploading stage of AFL to save communication costs but without compromising QoS. In this section, first, the problem formulation is described; then the details of FedRC are presented; finally, the algorithm of FedRC is introduced. For ease of expression, the key notations and abbreviations used in this paper are summarized in Table 2 and Table 3, respectively.

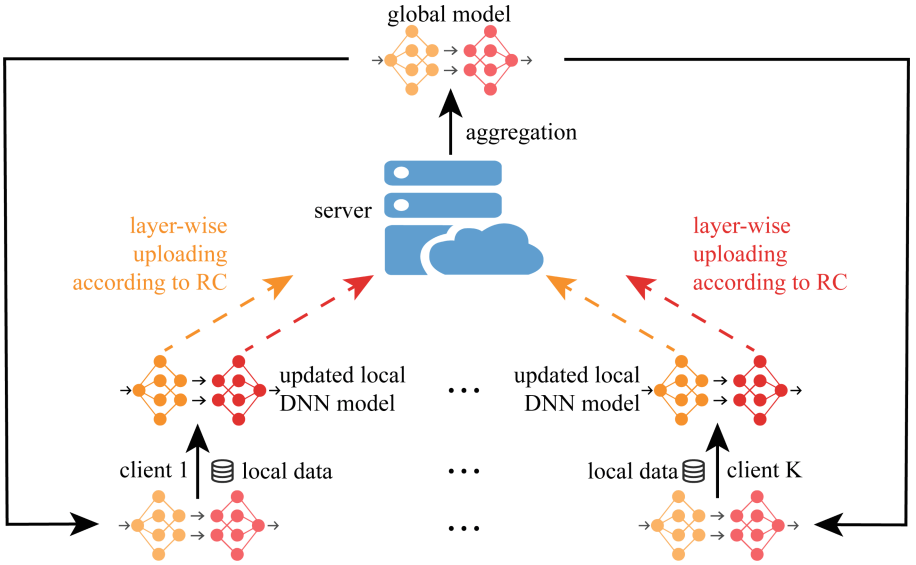


Fig. 1. The overall diagram of FedRC.

3.1 Problem Formulation

We consider an AFL system with total K clients, and each client k trains a local model w_k on its local dataset D_k with the size of $|D_k|$. Accordingly, the loss function of client k is defined as Formula 1,

$$F_k(w_k) = \frac{1}{|D_k|} \sum_{d_{k,i} \in D_k} f(d_{k,i}; w_k) \tag{1}$$

where D_k holds the training samples $d_{k,1}, d_{k,2}, ..., d_{k,|D_k|}$, and $f(\cdot)$ is a user-specified loss function, such as cross entropy and mean squared error.

After client k receives a global model w_{t_k} in the t_k^{th} global round, it conducts several local iterations for local update by formula 2,

$$w_k = w_{t_k} - \eta F'(w_{t_k}, D_k) \tag{2}$$

where η is the learning rate, and $F'(\cdot)$ is the gradients. Note that in AFL, for two different clients i and j, t_i is not necessarily equal to t_j, so as w_{t_i} and w_{t_j}.

Once the training of w_k is finished, w_k is uploaded to the server for global aggregation. Since in AFL, the aggregation process is unblocked, it can be triggered by a pre-defined condition, such as the maximum waiting time and the certain number of received local models of the server. Furthermore, the global updating can be executed by Formula 3,

$$w_{t+1} = \sum_{k=1}^{K_t} (\frac{|D_k|}{|D_t|} \times w_k) \tag{3}$$

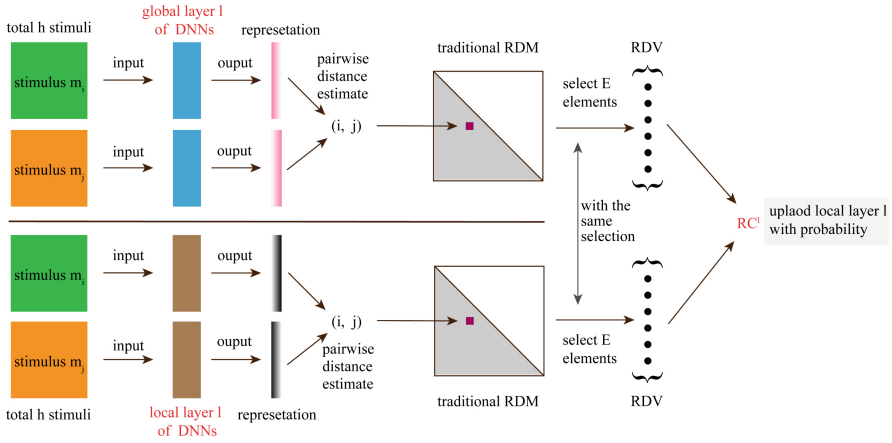


Fig. 2. The schematic diagram of FedRC for a client in a certain round.

where K_t and D_t are the number of participants and the total training samples in the t^{th} global round, respectively.

In this paper, we focus on the training of DNNs, hence w_k consists of multiple layers, which are denoted as L. Intuitively, the data heterogeneity brings model heterogeneity, i.e., layer heterogeneity. Moreover, deep layers (such as fully connected layers) contain many more parameters to learn ad hoc features compared with shallow layers (such as convolutional layers) [3]. Therefore, a layer-wise model uploading strategy is more suitable for AFL considering data heterogeneity and limited communication resources. In this context, we rewrite Formula 3 to Formula 4,

$$w_{t+1} = \sum_{l=1}^{L} \sum_{k=1}^{K_t^l} \left(\frac{|D_k|}{|D_t^l|} \times w_k^l \right) \tag{4}$$

where w_k^l is the l^{th} layer of w_k; K_t^l is the number of uploaded local layer l in the t^{th} round; and $D_t^l = \sum_{k=1}^{K_t^l} D_k$. Such that, the essential problem is determining whether to upload w_k^l in each round, which is solved by FedRC.

3.2 Details of FedRC

To measure the layer importance adaptively and guide the layer-wise model uploading strategy, Representational Similarity Analysis (RSA) [25] is adopted by FedRC. As shown in Fig. 2, given a set of stimuli $M = \{m_1, m_2, ..., m_h\}$, traditional RSA needs to compute a Representational Dissimilarity Matrix (RDM) with size $h \times h$ according to Formula 5,

$$RDM[i, j] = g(o_i^l, o_j^l), \quad i, j \leq h \tag{5}$$

where $o_i^l = o^l(m_i)$ denotes the output vector of l^{th} layer given stimula m_i; and $g(\cdot)$ measures the distance between two vectors, such as cosine distance (Formula 6), correlation distance (Formula 7), and Euclidean distance (Formula 8).

$$cos(o_i^l, o_j^l) = \frac{o_i^l \cdot o_j^l}{|o_i^l||o_j^l|} \tag{6}$$

$$cor(o_i^l, o_j^l) = 1 - \frac{Cov(o_i^l, o_j^l)}{\sqrt{D(o_i^l)}\sqrt{D(o_j^l)}} \tag{7}$$

$$euc(o_i^l, o_j^l) = \sqrt{\sum_{p=1}^{P^l}(o_i^l(p) - o_j^l(p))^2} \tag{8}$$

Note that the stimuli set M are prepared by the server according to public datasets and then are transmitted to all clients for computation. To reduce the computational complexity, we only randomly select and calculate E elements of the RDM to form a Representational Dissimilarity Vector (RDV). Such that, the simplified Representational Consistency (RC) between the global layer and local layer is defined according to Formula 9,

$$RC^l(RDV_{glo}^l, RDV_{loc}^l) = \rho(RDV_{glo}^l, RDV_{loc}^l)^2 \tag{9}$$

where ρ is the Pearson correlation coefficient.

A threshold can be designed to decide whether to upload w_k^l according to RC_k^l. However, such a threshold is a hard-coded uploading condition, and it requires strong experience or multiple experiments to set an appropriate threshold. To overcome this problem, FedRC introduces probability. Note that $0 \leq RC \leq 1$, hence RC^l can be used as the probability of uploading the l^{th} layer. To further regulate the probability to a proper scale, Min-Max normalization is employed according to Formula 10.

$$pro(w_k^l) = \frac{RC_k^l - min(RC^l)}{max(RC^l) - min(RC^l)} \tag{10}$$

3.3 Algorithm of FedRC

As shown in Algorithm 1, FedRC consists of two parts, namely:

Part 1: (in Each AFL Client). First, each AFL client k receives the global model w_{t_k} from the server in the t_k round. Second, client k clones w_{t_k} as the initialized local model w_k currently. Third, w_k is updated based on local dataset D_k. Fourth, the representational consistency RC_k^l is calculated for each layer. Finally, w_k^l is uploaded to the server-side with a probability computed by Formula 10.

Algorithm 1. The proposed FedRC algorithm

PART 1: Executed in each AFL client

for $k \leq K$ in parallel **do**
  Receiving the global model w_{t_k}
  $w_k = w_{t_k}$
  Training the local model w_k with D_k
  for layer $l \in L$ **do**
    Calculating RC_k^l by Formula 9
    Calculating $pro(w_k^l)$ by Formula 10
    Uploading w_k^l with probability $pro^(w_k^l)$
  end for
end for

PART 2: Executed in the AFL server

Thread 1: Receiving local models continuously
Thread 2: Aggregating local models periodically
while the aggregation condition is triggered **do**
  for $l \leq L$ **do**
    $w_{t+1}^l = \sum_{k=1}^{K_t^l} (\frac{|D_k|}{|D_t^l|} \times w_k^l)$
  end for
  $w_{t+1} = \sum_{l=1}^{L} w_{t+1}^l$
  Transmitting w_{t+1} to clients
  Increasing the global round t to $t+1$
end while

Part 2: (in the AFL Server). First, the AFL server receives local models from clients consistently. Second, once the aggregation condition is triggered, the server will update the global model layer by layer. Finally, the new global model w_{t+1} is transmitted to the client-side and the next global round begins.

Accordingly, the time complexity of the algorithm is $O(L \times h \times e_{max})$, where e_{max} denotes the maximum dimension of the output among the L layers. Although FedRC introduces additional computation for RC, the cost is actually negligible compared with the local training of DNNs.

In summary, as a layer-wise uploading AFL mechanism for the training of DNNs, FedRC introduces a simplified RC as the probability of transmission for each local layer at the client-side, while the server receives and aggregates local layer updates to improve the global model.

4 Experiment and Discussion

This section first describes the preparation of the experiment and then analyses the performance of FedRC comprehensively. Finally, the pros and cons of FedRC are further discussed based on the observations from experimental results.

4.1 Experiment Setting

We set up an AFL system comprising one server and 80 clients to evaluate the effectiveness and efficiency of FedRC. Note that the transmission time of clients ranges from 3 s to 30 s, and the AFL server aggregates the global model every 5 s. The system is developed based on TensorFlow libraries by using Python language, and we execute the experiment on a hardware device equipped with 4 NVIDIA GeForce RTX 3090 GPUs with 24 GB RAM based on Windows 10 Pro and CUDA v11.4.

Datasets and Models. The evaluations are conducted on three public datasets to train three corresponding CNN models. The default training settings are summarized in Table 4, and the details of datasets and corresponding model structures are listed as below:

- **MNIST**[1]: Modified National Institute of Standards and Technology dataset contains 60,000 handwritten digits for training and 10,000 for the test, with 10 labels and the image size of $28 \times 28 \times 1$. The CNN model has two stacked 5×5 convolution layers (32, 64 channels) followed by a 2×2 max pooling layer, two fully connected layers (128, 256 units) with relu activation function, and a softmax layer (10 units).
- **FMNIST**[2]: Fashion MNIST dataset has the same number of training samples, testing samples, and labels as MNIST. However, the images of FMNIST are clothes such as shirts and coats, which are more challenging to be classified correctly. The corresponding CNN model also shares the same structures as MNIST.
- **CIFAR-10**[3]: CIFAR-10 dataset comprises 60,000 $32 \times 32 \times 3$ colour images in 10 classes (such as the airplane and frog), with 6,000 images per class. There are 50,000 training images and 10,000 testing images. Since the task complexity is high, the CNN layers of CIFAR-10 have doubled channels or units compared with MNIST, and the other configurations are the same as MNIST.

Data Partition. To simulate a real-world IoT scenario, instead of IID data, we prepare Non-IID data based on the three datasets according to [3]. Specifically, for a dataset (i.e., MNIST, FMNIST, or CIFAR-10), each client is assigned with a data partition with training samples ranging from 500 to 1,500, and label numbers ranging from 1 to 6. Note that the fraction of each category in a partition is also imbalanced. As for the testing samples, they are untouched in the learning process and only used for evaluation.

[1] http://yann.lecun.com/exdb/mnist.
[2] https://github.com/zalandoresearch/fashion-mnist.
[3] https://www.cs.toronto.edu/~kriz/cifar.html.

Table 4. The summary of training setting.

Term	MNIST	FMNIST	CIFAR-10
Local epoch	5	5	5
Learning rate	0.003	0.001	0.001
Batch size	48	48	48
Optimizer	SGD	Adam	Adam

Benchmarks and Performance Metrics. Four state-of-the-art methods are adopted as the benchmarks for performance comparison, namely:

- **FedAvg** [21]: The canonical FL method proposed by Google, which selects participants randomly and aggregates local models averagely.
- **FedProx** [14]: It is a variant of FedAvg, which adds a proximal term μ on the local optimization function to address the data heterogeneity issue.
- **FedAsync** [34]: It is a popular AFL algorithm, which updates the global model immediately after receiving a local model according to a mixed hyperparameter α.
- **PartialNet** [24]: It is a communication-efficient FL algorithm, which only transmits the parameters of the largest layer of DNNs in each round.

To evaluate the learning performance, three metrics are used, i.e., 1) the maximum test accuracy during the training process according to Formula 11, where TP, TN, FP, and FN denote true positives, true negatives, false positives, false negatives, respectively; 2) the transmission time accumulated when the target accuracy (i.e., 90%, 85%, and 50% for MNIST, FMNIST, and CIFAR-10, respectively) is first reached; and 3) the total communication cost when the target accuracy is reached, which is calculated by Formula 12, where $n^r_{parameter}$ represents the size of uploaded parameters in the r^{th} round.

$$acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{11}$$

$$cost_{target} = \sum_{r=1}^{r_{target}} \left(\frac{4 \times n^r_{parameter}}{1024 \times 1024}\right) \tag{12}$$

4.2 Performance Evaluation

First, stimuli sets are needed to calculate RC for the uploading probability of model layers. Specifically, for each dataset, 100 images (10 categories $\times$ 10 stimuli) from the testing samples are prepared as the stimuli, and the dimension of RDV E is set as 100. Compared with the original RDM, which consists of $4,900$ unique elements, RDV can significantly save computational resources. Since there are three different distance functions as described by Formula 6-8, RedRC has three variants, namely FedRC-cos/cor/euc, and they will be evaluated and compared respectively.

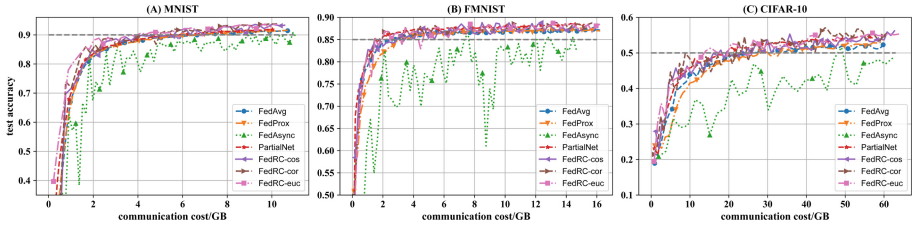


Fig. 3. Test accuracy vs. communication cost in (A) MNIST; (B) FMNIST; and (C) CIFAR-10.

Table 5. The experiment results of baselines and FedRC.

Method	Maximum test accuracy			Transmission time (in seconds)		
	MNIST	FMNIST	CIFAR-10	MNIST	FMNIST	CIFAR-10
FedAvg	91.48%	87.57%	52.35%	3623	1416	2148
FedProx	91.67%	87.34%	53.70%	3562	1615	2680
FedAsync	91.78%	87.60%	54.76%	**534**	**458**	**545**
PartialNet	**91.87%**	**88.77%**	**54.97%**	3247	979	1954
FedRC-cos	95.47%	88.84%	56.26%	475	270	325
FedRC-cor	**95.56%**	**88.99%**	57.67%	505	**185**	**290**
FedRC-euc	95.21%	88.89%	**58.36%**	**395**	195	330

1) Evaluation of Test Accuracy

First, as shown in Table 5, among the four baselines, PartialNet achieves the highest test accuracy on all three datasets, i.e., 91.87% on MNIST, 88.77% on FMNIST, and 54.97% on CIFAR-10, which suggests the advantages of layer-wise FL algorithm.

Second, all the three FedRC variants can outperform the best baseline PartialNet on the three datasets. Specifically, the best variant for each dataset, namely FedRC-cor, FedRC-cor, and FedRC-euc, can improve accuracy by about 4.02%, 0.25%, and 6.17% on MNIST, FMNIST, and CIFAR-10, respectively.

Third, the three FedRC variants have similar test accuracies on MNIST and FMNIST, while the differences on CIFAR-10 are slightly large, which indicates that complicated tasks are more sensitive to the distance function.

In summary, the saving on communication costs of FedRC will not deteriorate the model performance, and surprisingly, an average improvement of 3.48% on the test accuracy is observed. The results show that FedRC can filter the redundancy of local models caused by data heterogeneity, and only uploads and aggregates more useful local layers to form a better global model.

2) Evaluation of Communication Efficiency

First, as illustrated in Fig. 3, FedAsync needs much more communication resources for model training compared with other baselines (i.e., FedAvg, FedProx, and PartialNet), and the curve of FedAsync is much more vibrant, which suggests that general AFL algorithms sacrifice communication efficiency and learning stability for stragglers.

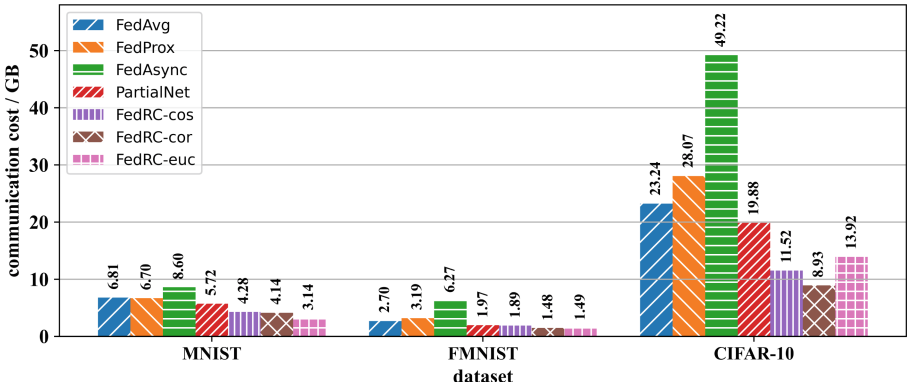


Fig. 4. The communication cost of FedRC compared with the baselines in the three datasets.

Second, when consuming the same communication cost, FedRC can not only improve accuracy significantly compared with FedAsync, but also boost model performance slightly compared with FedAvg, FedProx, and PartialNet.

Finally, as shown in Fig. 4, FedRC spends less communication cost to reach target accuracy compared with the baselines, namely:

- **For MNIST**: FedRC-cos, FedRC-cor, and FedRC-euc only need 4.28, 4.14, and 3.14 GB to reach 90% accuracy respectively, and an average reduction of 32.63% is observed compared with the best baseline PartialNet (5.72 GB);
- **For FMNIST**: The average consumption of the three FedRC variants is 1.62 GB, while the best baseline PartialNet needs 1.97 GB to get an accuracy of 85%. Hence, the reduction of cost is 17.77%;
- **For CIFAR-10**: Compared with the best performance of the four baselines (i.e., PartialNet 23.24 GB to reach 50%), the maximum, minimum, and average improvements in communication efficiency of FedRC variants are 55.08%, 29.98%, and 42.37%, respectively.

In summary, since FedRC adopts the layer-wise uploading approach based on RC-based uploading probability, the improvement of FedRC on communication efficiency is significant, and an average cost reduction of 26.79% against the best baseline of the three datasets is observed. Note that all three distance functions are favourable, and the stability of FedRC is not affected by the reduction of uploaded parameters.

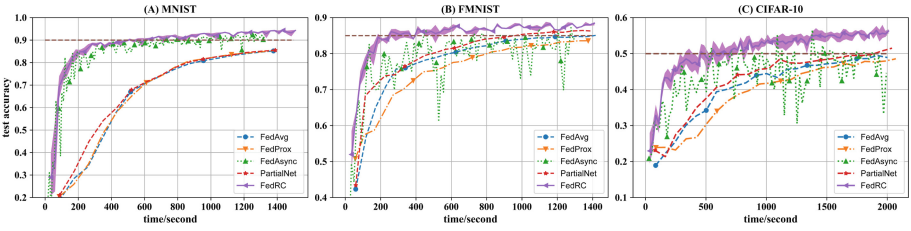


Fig. 5. Test accuracy vs. transmission time in (A) MNIST; (B) FMNIST; and (C) CIFAR-10.

3) Evaluation of Time Efficiency
First, as summarized in Table 5, among the four baselines, FedAsync presents the shortest transmission time (534/458/545 s) to reach the target accuracy on all three datasets (MNIST/FMNIST/CIFAR-10), which indicates the high time efficiency of AFL algorithms.

Second, all three FedRC variants can reach the target accuracy faster than FedAsync. Specifically, 1) FedRC-cor performs the best on both FMNIST and CIFAR-10, as it only needs 185 s and 290 s to reach the target accuracies of 85.00% and 50.00% with a reduction of 59.61% and 46.79% compared with FedAsync, respectively; 2) FedRC-euc is the fastest to reach the accuracy of 90% on MNIST, and a boost on the learning speed of 26.03% against the best baseline is observed.

Finally, as illustrated in Fig. 5, the three FedRC variants share a similar time-accuracy curve, which outperforms FedAsync as well as FedAvg, FedProx, and PartialNet. Note that FedRC does not vibrate sharply compared with FedAsync, and can gain accuracy improvement quickly compared with FedAvg, FedProx, and PartialNet.

In summary, FedRC inherits and enhances the advantages in time efficiency of AFL methods, and in the meanwhile, can maintain a more stable learning curve.

4.3 Discussion

First, four main advantages of FedRC are observed from the results: 1) improving the model accuracy; 2) saving communication costs; 3) shortening the transmission time; 4) maintaining a stable learning curve.

Second, as for the comparisons among different distance functions, we discover the following three interesting observations: 1) all three distance functions are effective as they share similar performance improvements and learning curves; 2) the complicated dataset may be more sensitive to the distance function, as its accuracies under different distances are more diverse; 3) the correction distance has higher compatibility for FedRC and can be the default method, as FedRC-cor outperforms FedRC-cos and FedRC-euc slightly in most cases in terms of the test accuracy and transmission time. Intuitively, all three distance functions can measure the similarity between two representational vectors, and the correlation distance may be more suitable for complex datasets, which generally require high-dimensional representations.

Finally, although the learning curve of FedRC is much more stable compared with FedAsync, it still vibrates slightly compared with FedAvg and FedProx, which may be due to the absence of utilizing temporal attributes to improve the aggregation process.

5 Conclusion

In this paper, we present FedRC, a layer-wise model uploading mechanism for asynchronous federated learning. Based on the representational dissimilarity vector, representational consistency is calculated as the layer uploading probability, since it can capture the similarity of stimulus representation between the local layer and the corresponding global layer.

A comprehensive evaluation using three datasets (i.e., MNIST, FMNIST, and CIFAR-10) demonstrates that FedRC outperforms four state-of-the-art baselines (i.e., FedAvg, FedProx, FedAsync, and PartialNet) by about 3.48%, 26.79%, and 44.14% in terms of learning performance, communication efficiency, and transmission time efficiency, respectively. The results prove that representational consistency can indeed be utilized as the uploading probability to transmit more valuable layer parameters with communication cost reduced and model accuracy improved.

However, a slight performance fluctuation during the learning is observed in FedRC when compared to FedAvg and FedProx, and additional computation is required for the calculation of RC, which indicate that FedRC can be further improved. In the future, we will focus on the following directions: 1) the RDV will be further optimized to enhance the representation level without increasing computational resources; 2) complicated DNNs with more layers will be utilized in the experiments to further evaluate FedRC and reveal the relationship between layer number and representational consistency; and 3) layer-wise aggregation strategies will be designed to improve learning stability.

References

1. Bonawitz, K., et al.: Towards federated learning at scale: system design. Proc. Mach. Learn. Syst. **1**, 374–388 (2019)
2. Chai, Z., Chen, Y., Anwar, A., Zhao, L., Cheng, Y., Rangwala, H.: Fedat: a high-performance and communication-efficient federated learning system with asynchronous tiers. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–16 (2021)
3. Chen, Y., Sun, X., Jin, Y.: Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. IEEE Trans. Neural Netw. Learn. Syst. **31**(10), 4229–4238 (2020). https://doi.org/10.1109/TNNLS.2019.2953131
4. Chen, Y., Ning, Y., Slawski, M., Rangwala, H.: Asynchronous online federated learning for edge devices with non-iid data. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 15–24. IEEE (2020)
5. Ek, S., Portet, F., Lalanda, P., Vega, G.: Evaluation and comparison of federated learning algorithms for human activity recognition on smartphones. Pervasive Mob. Comput. **87**, 101714 (2022). https://doi.org/10.1016/j.pmcj.2022.101714
6. Haddadpour, F., Kamani, M.M., Mokhtari, A., Mahdavi, M.: Federated learning with compression: unified analysis and sharp guarantees. In: International Conference on Artificial Intelligence and Statistics, pp. 2350–2358. PMLR (2021)
7. Herabad, M.G.: Communication-efficient semi-synchronous hierarchical federated learning with balanced training in heterogeneous iot edge environments. Internet Things **21**, 100642 (2023). https://doi.org/10.1016/j.iot.2022.100642
8. Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., Lane, N.: Fjord: fair and accurate federated learning under heterogeneous targets with ordered dropout. Adv. Neural. Inf. Process. Syst. **34**, 12876–12889 (2021)
9. Hosseinalipour, S., Brinton, C.G., Aggarwal, V., Dai, H., Chiang, M.: From federated to fog learning: distributed machine learning over heterogeneous wireless networks. IEEE Commun. Mag. **58**(12), 41–47 (2020). https://doi.org/10.1109/MCOM.001.2000410
10. Hu, C.H., Chen, Z., Larsson, E.G.: Scheduling and aggregation design for asynchronous federated learning over wireless networks. IEEE J. Sel. Areas Commun. **41**(4), 874–886 (2023). https://doi.org/10.1109/JSAC.2023.3242719

11. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: International Conference on Machine Learning, pp. 3519–3529. PMLR (2019)
12. Kriegeskorte, N., Mur, M., Bandettini, P.: Representational similarity analysis - connecting the branches of systems neuroscience. Front. Syst. Neurosci. **2** (2008). https://doi.org/10.3389/neuro.06.004.2008
13. Li, J., et al.: A federated learning based privacy-preserving smart healthcare system. IEEE Trans. Ind. Inf. **18**(3), 2021–2031 (2022). https://doi.org/10.1109/TII.2021.3098010
14. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
15. Liu, S., Chen, Q., You, L.: Fed2a: federated learning mechanism in asynchronous and adaptive modes. Electronics **11**(9), 1393 (2022). https://doi.org/10.3390/electronics11091393
16. Liu, S., Qu, H., Chen, Q., Jian, W., Liu, R., You, L.: Afmeta: asynchronous federated meta-learning with temporally weighted aggregation. In: 2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta), pp. 641–648 (2022). https://doi.org/10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00100
17. Liu, Y., et al.: Fedvision: an online visual object detection platform powered by federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 13172–13179 (2020)
18. Lu, X., Liao, Y., Lio, P., Hui, P.: Privacy-preserving asynchronous federated learning mechanism for edge network computing. IEEE Access **8**, 48970–48981 (2020). https://doi.org/10.1109/ACCESS.2020.2978082
19. Ma, Q., Xu, Y., Xu, H., Jiang, Z., Huang, L., Huang, H.: FEDSA: a semi-asynchronous federated learning mechanism in heterogeneous edge computing. IEEE J. Sel. Areas Commun. **39**(12), 3654–3672 (2021)
20. Manias, D.M., Shami, A.: Making a case for federated learning in the internet of vehicles and intelligent transportation systems. IEEE Network **35**(3), 88–94 (2021). https://doi.org/10.1109/MNET.011.2000552
21. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
22. Mehrer, J., Spoerer, C.J., Kriegeskorte, N., Kietzmann, T.C.: Individual differences among deep neural network models. Nat. Commun. **11**(1), 1–12 (2020). https://doi.org/10.1038/s41467-020-19632-w
23. Palihawadana, C., Wiratunga, N., Wijekoon, A., Kalutarage, H.: Fedsim: similarity guided model aggregation for federated learning. Neurocomputing **483**, 432–445 (2022). https://doi.org/10.1016/j.neucom.2021.08.141
24. Paragliola, G., Coronato, A.: Definition of a novel federated learning approach to reduce communication costs. Expert Syst. Appl. **189**, 116109 (2022). https://doi.org/10.1016/j.eswa.2021.116109
25. Raghu, M., Gilmer, J., Yosinski, J., Sohl-Dickstein, J.: SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In: 31st International Conference on Neural Information Processing Systems, pp. 6078–6087 (2017)
26. Safaryan, M., Shulgin, E., Richtárik, P.: Uncertainty principle for communication compression in distributed and federated learning and the search for an optimal compressor. Inf. Inference J. IMA **11**(2), 557–580 (2022). https://doi.org/10.1093/imaiai/iaab006
27. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2021). https://doi.org/10.1109/TNNLS.2020.3015958

28. Sattler, F., Wiedemann, S., Müller, K.R., Samek, W.: Robust and communication-efficient federated learning from non-i.i.d. data. IEEE Trans. Neural Netw. Learn. Syst. **31**(9), 3400–3413 (2020). https://doi.org/10.1109/TNNLS.2019.2944481
29. Sun, H., Ma, X., Hu, R.Q.: Adaptive federated learning with gradient compression in uplink noma. IEEE Trans. Veh. Technol. **69**(12), 16325–16329 (2020). https://doi.org/10.1109/TVT.2020.3027306
30. Wang, X., Li, R., Wang, C., Li, X., Taleb, T., Leung, V.C.M.: Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. IEEE J. Sel. Areas Commun. **39**(1), 154–169 (2021). https://doi.org/10.1109/JSAC.2020.3036946
31. Wang, Y., Wolfrath, J., Sreekumar, N., Kumar, D., Chandra, A.: Accelerated training via device similarity in federated learning. In: Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking, pp. 31–36 (2021)
32. Wu, C., Wu, F., Lyu, L., Huang, Y., Xie, X.: Communication-efficient federated learning via knowledge distillation. Nat. Commun. **13**(1), 1–8 (2022). https://doi.org/10.1038/s41467-022-29763-x
33. Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S.: SAFA: a semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. **70**(5), 655–668 (2021). https://doi.org/10.1109/TC.2020.2994391
34. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. arXiv preprint arXiv:1903.03934 (2019)
35. Xu, J., Du, W., Jin, Y., He, W., Cheng, R.: Ternary compression for communication-efficient federated learning. IEEE Trans. Neural Netw. Learn. Syst. **33**(3), 1162–1176 (2022). https://doi.org/10.1109/TNNLS.2020.3041185
36. You, L., He, J., Wang, W., Cai, M.: Autonomous transportation systems and services enabled by the next-generation network. IEEE Netw. **36**(3), 66–72 (2022). https://doi.org/10.1109/MNET.006.2100542
37. You, L., Liu, S., Chang, Y., Yuen, C.: A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement. IEEE Internet Things J. **9**(23), 24199–24211 (2022). https://doi.org/10.1109/JIOT.2022.3188556
38. You, L., Liu, S., Zuo, B., Yuen, C., Niyato, D., Vincent Poor, H.: Federated and asynchronized learning for autonomous and intelligent things. IEEE Netw. (2023). https://doi.org/10.1109/MNET.2023.3321519
39. You, L., Tunçer, B., Zhu, R., Xing, H., Yuen, C.: A synergetic orchestration of objects, data, and services to enable smart cities. IEEE Internet Things J. **6**(6), 10496–10507 (2019). https://doi.org/10.1109/JIOT.2019.2939496
40. Zhang, Y., Suleiman, B., Alibasa, M.J.: Fedgroup: a federated learning approach for anomaly detection in iot environments. In: Longfei, S., Bodhi, P. (eds.) MobiQuitous 2022. LNCS, vol. 492, pp. 121–132. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-34776-4_7

RADEAN: A Resource Allocation Model Based on Deep Reinforcement Learning and Generative Adversarial Networks in Edge Computing

Zhaoyang Yu[1,2], Sinong Zhao[1], Tongtong Su[1], Wenwen Liu[1], Xiaoguang Liu[1(✉)], Gang Wang[1], Zehua Wang[2], and Victor C. M. Leung[2]

[1] College of Computer Science, ICIC, TMCC, SysNet, DISSec, GTIISC, Nankai University, Tianjin, China
{yuzz,zhaosn,sutt,liuww,liuxg,wgzwp}@nbjl.nankai.edu.cn
[2] Department of Electrical and Computer Engineering, WiNMos Lab, University of British Columbia, Vancouver, Canada
{zwang,vleung}@ece.ubc.ca

Abstract. Edge computing alleviates the network congestion and latency pressure on the remote cloud as well as the computation stress on end devices. However, facing numerous tasks, how to effectively allocate and manage the resources of edge servers is of great significance, which affects both the benefits of service providers and Quality of Service (QoS) of users. This paper proposes a resource allocation model based on Deep Reinforcement Learning (DRL) and Generative Adversarial Networks (GAN)(RADEAN). It considers the future resource occupancy of edge servers, and applies multi-replay memory with priority to eliminate the interaction between experiences and improve sampling efficiency. We maximize the average resource utilization of edge servers while ensuring the average transmission latency (ATL) and average execution time (AET) of tasks in a long-term view. Specifically, based on the state which consists the predicted resource occupation output by GAN, the current resource usage status of edge servers and the characteristics of the task queue, DRL agent makes resource allocation decision for each task. We conduct experiments using real-world data trace, and show that RADEAN outperforms traditional and state-of-the-art models with great generalization, reaching the maximum performance improvement of 48.21% compared with MMRA. The ATL and AET of tasks are also presented to reflect the QoS guarantee. Ablation experiments prove the effectiveness of multi-replay memory and priority sub-modules.

Keywords: edge computing · resource allocation · deep reinforcement learning · generative adversarial networks

This research is supported in part by the NSF of China (62272253, 62272252, 62141412) and Fundamental Research Funds for the Central University, China Scholarship Council (CSC). The first author is supported by CSC (Grant No. 202106200061) as a visiting Ph.D. student at the University of British Columbia, Canada under the supervision of Prof. Victor C.M. Leung.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 257–277, 2024.
https://doi.org/10.1007/978-3-031-63989-0_13

1 Introduction

Recently, the number of Internet of Things (IoT) terminals increases exponentially and is expected to reach 125 billion by 2030 [1]. Substantial delay-sensitive and computing-intensive tasks, such as Augmented Reality (AR), Virtual Reality (VR) and target detection, bring huge challenges to task execution correspondingly. Multiple edge clouds cooperate for processing tasks, reducing task execution delay, data transmission security and backhaul impact in remote cloud and the computing pressure on end devices.

An overloaded CPU will cause long execution delay, which is unacceptable for latency-sensitive tasks. Compared with the remote cloud, the conditions of computing resource allocation in edge are more complicated. First, the computation resource capacities, storage space and processing speed of edge servers are relatively limited. Second, the heterogeneity and dynamics of end-user devices and edge nodes determine that the resource allocation in edge is NP-hard [2]. User tasks from various hardware specifications with different objectives, architectures, and communication protocols, etc. Edge clouds with various coverage areas are equipped with different edge servers, conducting diverse strategies of resource allocation. Third, the mobility of users between multiple edge clouds involves the release and reallocation of resources. Putting these all together creates a more serious challenge for resource allocation in edge.

For each task submitted by users, the resource allocation manager in the edge server decides how many computing resources are going to be assigned to this task and how long it is allowed to occupy resources. If the task is allocated few computing resources, its increased occupation time may cause task congestion. The waiting and completion delay of tasks will raise accordingly, which is potentially fatal for latency-sensitive tasks. However, assigning too many computing resources to a certain task may result in resource deficit for others. In this case, these tasks must be migrated to other edge clouds or even remote clouds for execution. Task migration brings extensive transmission time, protracted execution latency and high migration cost, thereby reducing the Quality of Service (QoS) for users. Therefore, it's an urgent problem that how to allocate and schedule resources in edge servers to achieve high resource utilization, improve the efficiency of service providers, and guarantee users' QoS in the meantime.

Some previous works involved in resource allocation focused on the cost [3,4], energy consumption [5] and latency [6]. The research with the optimization goal of resource utilization [7] ignored the impact of future resource usage of edge servers. Applying Deep Reinforcement Learning (DRL) in resource management [8,9] makes up for the shortcoming of the high computation cost of traditional approaches such as heuristic [10] and meta-heuristic [3,11] algorithms. Most DRL methods enhance the performance by improving the model structure during training process, but ignore the optimization from the perspective of task characteristic analysis. In fact, a job contains multiple tasks with dependencies, and the generation and requests of tasks are correlated with time series to a certain extent. Therefore, facing the dynamic and predictable nature of task offloading, the resource utilization of edge servers can be predicted based on the

historical system data, so as to make better resource allocation decisions. The semi-supervised method Generative Adversarial Networks (GAN) performs confrontational training without requiring a large number of training samples and prior knowledge. It generates clearer data and establishes description model to enhance the prediction effect, behaving well for time series prediction in workload forecast [12], financial prediction [13] and other fields. Hence, we choose GAN to estimate the future resource condition and assist the resource allocation decisions for DRL agent. Moreover, we build an improved DRL approach to adjust the resource allocation strategy through the trial-return feedback mechanism. It tracks the obtained reward for resource allocation decisions and feeds back to the agent for policy adjustment, which is more robust and adaptable in edge to achieve the long-term maximum average resource utilization.

Taking the above factors into consideration, we propose a **R**esource **A**llocation model based on **D**eep **Re**inforcement Learning and Generative **A**dversarial **N**etworks (RADEAN). We apply GAN to predict the future resource utilization of edge servers, and make resource allocation decisions through a multi-replay memory DRL agent with priority to maximize the long-term average resource utilization. Meanwhile, we guarantee the transmission latency and execution time of tasks to ensure the users' QoS. The contributions are as follows:

- We propose a RADEAN model for efficient resource allocation, which uses an improved Deep Q Network (DQN) with multiple replay buffers to reduce the interaction between multiple samples, and prioritize samples to further promote the decision performance of the proposed model.
- We apply GAN to forecast the CPU resource usage of the next episode, exploiting the semi-supervised learning to enhance prediction accuracy and considering it as one of bases for better resource allocation decisions.
- We conduct experiments on real-data sets, validating that RADEAN exceeds traditional and advanced resource allocation methods with great generality while ensuring transmission latency and execution time. The ablation experiments prove the effectiveness of multi-replay memory and priority settings.

2 Related Work

Resource allocation refers to assigning the task sets with different QoS requirements to densely distributed edge/fog nodes, which is determined to achieve the goals of improved resource utilization, decreased energy consumption, makespan, and cost, or lower task calculation latency [14]. We present several methods for resource allocation in edge computing.

Traditional: LF Bittencourt et al. [15] used three different policies to conduct resource management based on application characteristics. J Lin [16] introduced Lyapunov optimization to propose a dynamic resource scheme to minimize cost and maximize system utility. A hybrid market-based resource transaction mechanism was established in [17] to maximize utility. But these methods were complex with high computational cost and unable to obtain optimal solutions.

Heuristic and Meta-heuristic: Studies applied a multi-criteria-based policy [18] and a heuristic-based [10] resource allocation scheme. Meta-heuristic approaches dealt with various constraints caused by heuristic methods. Some work suggested game-theoretic resource management techniques to minimize infrastructure energy consumption and costs [3]. While the other developed algorithms to determine the Stackelberg equilibrium and chose the best resource demand strategy [11]. Such mechanisms were prone to fall into local minima.

Deep and Reinforcement Learning: Authors utilized an LSTM-based algorithm [5] and multiple parallel deep neural networks [19] for resource management in edge. Reinforcement learning (RL) had stronger self-improvement ability than deep learning. Researchers exploited the actor-critic reinforcement learning [20] and a parameterized DQN [6] to solve the joint decision-making issue, minimizing the average end-to-end latency. A DRL-based resource allocation scheme with minimum computing cost was created in [4]. None of them studied the impact of future resource conditions on allocation decisions.

Hybrid: Several hybrid approaches combing other algorithms with DRL were employed for resource allocation. Heuristic and meta-heuristic algorithms were incorporated with RL in [21] and [22], respectively. And in [23], a recurrent neural network-based long short-term memory combined with DRL was completed. We also apply a hybrid method, which uses an improved DQN and GAN to achieve the maximum long-term average resource utilization.

3 Problem Definition

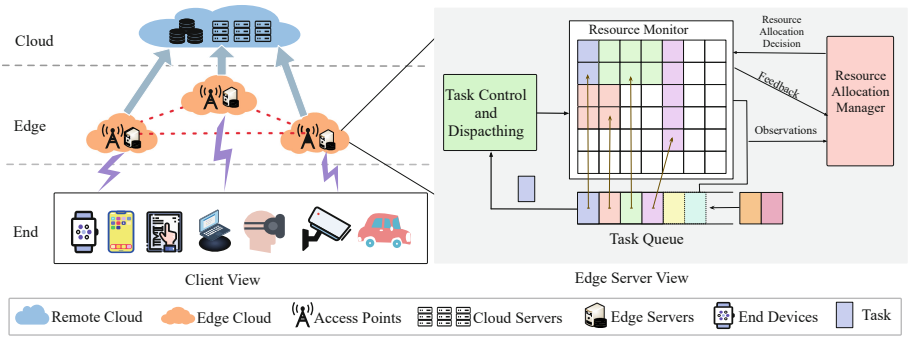


Fig. 1. The scenario of resource allocation in edge computing.

As shown in Fig. 1, it is a typical cloud-edge-end three-layer architecture. In intelligent driving application scenario, various smart devices such as self-driving cars, continuously offload tasks to the edge layer for certain computing resource requests from edge servers. The edge layer includes multiple edge clouds, which

is usually equipped with multiple edge servers. The Task Queue in the edge server stores tasks from terminal devices and their execution locations are determined by Task Control and Dispatching agent. If the edge server cannot meet the resource requirements of the tasks currently, tasks will be migrated to another edge server or forwarded to the remote cloud for procession. The Resource Monitor observes the edge server's resource usage. Based on the resource utilization status, the Resource Allocation Manager runs the resource allocation strategy to assign its own resources to all tasks on this server to meet user demands. The edge clouds cooperate with the remote cloud, which handles the tasks that can not be processed due to computation limitation of the edge layer. Also, the remote cloud defines and adjusts AI reasoning model, being responsible for unified operation management, manual review and persistent storage of key data. It makes the entire system more efficient and supportable for richer applications.

Unreasonable resource allocation mechanisms affect users' QoS and interests of service providers. If a task is allocated too few computing resources, its execution time will increase. And there will be a congestion in the task queue due to numerous unfinished tasks, which greatly raises the overall latency of tasks. Besides, the resources that could have been used will be idle instead, reducing the resource utilization of the edge server. If too many computing resources are allocated to a task, it may lead to resource deficit and data migration for other tasks. The subsequent increased data migration costs and transmission delay greatly influence latency-sensitive tasks especially. Higher latency will cause tasks such as target detection not be able to get feedback in time, which is detrimental both to users and service providers.

Assume that within a period of time, a total of V jobs are received from the view of the edge server. The job set is expressed as $\mathcal{J} \triangleq \{j_1, j_2, \ldots, j_V\}$. Each job $j_v, j_v \in \mathcal{J}$ contains multiple tasks and each task includes multiple instances. There are N tasks in all and they are stored in the Task Queue $\mathcal{Q} \triangleq \{q_1, q_2, \ldots, q_N\}$, according to their arrival time and following the First In First Out (FIFO) principle. For each task $q_n, q_n \in \mathcal{Q}$, the profile of each task is $q_n = (c_n, \omega_n, \rho_n, \xi_n)$, where c_n means the amount of requested computing resources, ω_n and ρ_n are the number of instances and data volume carried by q_n, respectively. The symbol ξ_n represents the deadline request of task q_n.

For simplicity, we only consider computing resources here. Set the computing resource capacity of an edge server as K, and the length of the sliding time window as H. The Resource Monitor observes the resource usage and pays close attention to a matrix $\mathcal{M}$ with size $K \times H$. Each row of $\mathcal{M}$ represents the state of a certain computing resource from current time to H timesteps in the future, $m(k,:) = \{m_{k,1}, m_{k,2}, \ldots, m_{k,h}, \ldots, m_{k,H}\}, k \in \{1, 2, \ldots, K\}$. Each column of $\mathcal{M}$ depicts the status of all computing resources at the current timestep, $m(:,h) = \{m_{1,h}, m_{2,h}, \ldots, m_{k,h}, \ldots, m_{K,h}\}, h \in \{1, 2, \ldots, H\}$. Each unit $m_{k,h} \in \{0, 1\}$ indicates the usage of the computing unit of the k-th row at h-th timestep, where 1 means that the resource is already occupied, while 0 reveals not. $\mathcal{M}$ is a matrix of all 0 s initially, illustrating that all computing resources are available.

Table 1. Table of notations and representations

Type	Nota.	Representations
Job	$\mathcal{J}$	the set of jobs
	j_v	the v-th job in job set $\mathcal{J}$
	V	the total number of jobs
Task	q_n	the n-th task
	c_n	the amount of requested computing resources of task q_n
	ω_n	the number of instances carried by task q_n
	ρ_n	the data volume carried by task q_n
	ξ_n	the deadline request of task q_n
	l_n	the time offset of task q_n relative to the sliding window
	d_n	the execution duration time of task q_n
	N	the total number of tasks
Edge Server	K	the quantity of computing resource capacity
	$\mathcal{M}$	the observation resource matrix of the edge server
GAN	y_e	the real resource utilization at episode e
	$\tilde{y}_{e+1}$	the predicted resource utilization at episode $e+1$
	F	the length of prediction series
DRL	$\mathcal{Q}$	the task queue
	$\mathcal{Q}_A$	the observation task sub-queue
	L	the length of task queue $\mathcal{Q}$
	H	the length of sliding window
	a_n	the action (resource allocation decision) of task q_n
	s	the current state of the edge server
	r_n	the reward after performing action a_n
	s'	the next state after performing a_n

At timestep t, we make a resource allocation decision a_n for each task q_n, determining how many resources it will be allocated per unit timestep. Its execution duration time d_n can be calculated as $d_n = c_n/a_n$. Next, under the least time offset, we search for an all 0 s sub-matrix with size $a_n \times d_n$ in the current sliding window and set all values in the sub-matrix to 1, indicating that the corresponding resources are allocated to task q_n. After that, the resource utilization of the edge server will be updated to u_t accordingly. The sliding window moves forward by a timestep. The specific length of each time unit can be determined in terms of actual application conditions. As shown in (1), given a certain period of time including T timesteps, our goal is to maximize the average resource utilization of the edge server in a long-term view, while ensuring the completion time of tasks for users' QoS. The sum of the resource requests of all tasks on an edge server cannot exceed its resource capacity limitation.

$$\Pi(T) = \arg\max \frac{1}{T} \sum_{t=1}^{T} u_t, \quad \text{s.t.} \sum c < K. \tag{1}$$

Relevant notations and representations are shown in Table 1. A few points to note: First, the task requests submitted by users are completely offloaded from end devices to the edge server for execution, including all the carried data. All instances carried by the task execute in parallel. Second, we suppose that once the task starts, it will not be interrupted due to any reason, such as the termination of hardware factors, forced surrender of resources for higher-priority tasks, power failure, network connection interruption and so on. Third, the execution locations of tasks that have already begun are not allowed to change, nor are they allowed to migrate between multiple edge clouds or multiple edge servers.

4 Proposed Model

In this section, we first introduce the workflow of the proposed RADEAN, and then present the GAN-based computing resource prediction module and DRL-based resource allocation module.

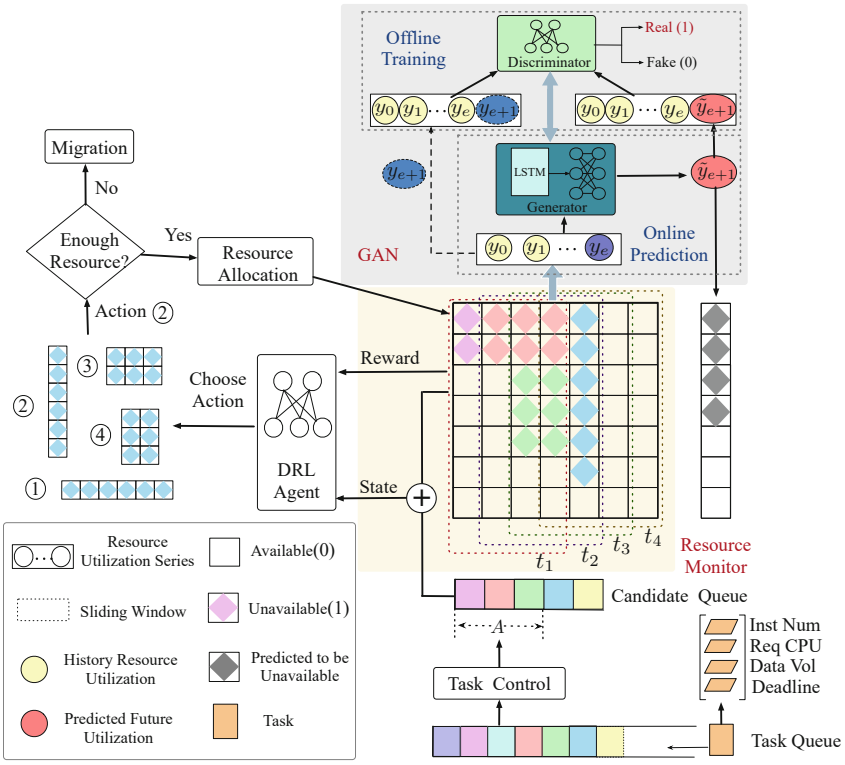


Fig. 2. The framework of RADEAN

The workflow of RADEAN is shown as Fig. 2. Tasks are sorted in the Task Queue according their arriving time and performed following FIFO principle. The tasks sequentially passed through the Task Control module, which is responsible for judging whether the task meets the basic conditions for execution on this edge server. If the deadline requirement of the task cannot been met under infinite resources, the task will be rejected. Otherwise, the task enters the Candidate Queue. The Resource Monitor supervises the server's resource usage. We select a sub-queue $\mathcal{Q}_A$ from the Candidate Queue, the size of which is A. The DRL-based agent decides the amount of allocated computing resources per time unit to the task and its start time, according to the current resource status, task sub-queue state and future resource utilization predicted by GAN. Then, the edge server checks whether the resource allocation requirement can be satisfied in the current sliding widow (sufficient computing resources and qualified all-zero sub-matrix to avoid prolonged waiting time). If so, the task is processed immediately. Otherwise, the task is migrated for execution.

When task is assigned relevant computing resources, the resource condition of the edge server modifies accordingly. The Resource Monitor calculates the resource utilization of the current sliding window, and feeds it back to the agent as a reward for resource allocation strategy adjustment. Then the sliding window moves forward one timestep, preparing for the next task's resource allocation.

We decouple the time into two phases, episode and timestep. There are E episodes in total and one episode contains A multiple timesteps. Resource allocation is executed for a task at each timestep. GAN predicts the future resource occupation $\tilde{y}_{e+1}$ of the next episode $e+1$, which is performed only after an episode ends. The GAN is trained in advance and only the Generator is used to conduct the online resource utilization prediction when making resource allocation decisions. The resource utilization records of the sliding window at different moments form a time series. And the Generator is trained offline and regularly updated through the Discriminator and these new data.

Tasks in edge are dynamic and heterogeneous. They are submitted from various applications with different resource requests and optimization subjects, arriving in arbitrary time and order. This makes the future resource occupancy has great uncertainty, which in turn affects the resource allocation decision and execution of tasks, influencing users' QoS. Consequently, it is meaningful to predict the future computing resource utilization. GAN is suitable and qualified for resource prediction: Firstly, for characterizing resource workload, one of the challenges of an effective fitting process is the prior selection of the model type, such as normal distribution [12]. While GAN does not require prior knowledge or assumptions about the sample distribution and underlying models themselves. It performs effective model descriptions based on the sample data, and estimate appropriate parameters. Secondly, GAN mines the information behind the resource utilization data, which is far richer than that expressed by sparse labels. It generates clearer and more real data according to the characteristics of the historical resource utilization trace of edge server and is superior to traditional neural network classification and feature extraction. Finally, the production cost

of the sample set is very high, while GAN can perform semi-supervised learning based on a small amount of trace data instead of massive data to train networks and complex models. The adversarial training between the Generator and the Discriminator further improves the prediction effect of the Generator.

GAN provides the more accurate predicted resource utilization of edge servers, contributing a more valuable state input for DRL to enhance resource allocation strategies and improve the overall system performance. DRL does not require manually labeled training samples and complication operations to obtain optimization results. It acquires resource utilization data from the Resource Monitor for training and adjusts the strategy based on feedback. For achieving the long-term maximum average resource utilization, DRL is more robust and adaptable to dynamic edge environment, benefiting both users and service providers.

Although the replay memory setting solves the problem of increased computational complexity and storage cost of Q-learning due to exponential number of states and actions, there are mutual influences between samples [8]. Therefore, we propose an improved DQN model, which puts transition samples into different replay buffers with various parameters according to the predicted resource occupancy values to achieve better exploration. It saves the space to store training data, ensures the diversity of samples, and eliminates the mutual sample influences to enhance data efficiency and training effect. Moreover, transitions may be more or less surprising, redundant than expected, and may not even be immediately useful to the agent. The key to the effectiveness of relevant transitions lies in the mass of highly redundant failure [24]. So it is not efficient enough to sample uniformly in replay buffers. We apply prioritized sampling to further improve the utilization of samples, reduce the number of training iterations, and promote the performance of the decision model.

4.1 Resource Prediction Based on GAN

GAN is pre-trained and its parameters are updated regularly using the historical resource trace. It includes a Generator and a Discriminator. In online prediction at episode e, the Generator predicts the resource utilization percent of episode $e+1$, using the historical trace series of previous F moments, $y_r = \{y_{e-F+1}, \ldots, y_{e-1}, y_e\}$. In offline training, the predicted result $\tilde{y}_{e+1}$ is combined with the real sequence y_r to form $y_G = \{y_{e-F+1}, \ldots, y_{e-1}, y_e, \tilde{y}_{e+1}\}$. There is another real trace $y_{G'} = \{y_{e-F+1}, \ldots, y_{e-1}, y_e, y_{e+1}\}$, in which y_{e+1} is the real resource utilization of episode $e+1$. The two series y_G and $y_{G'}$ are input into the Discriminator to be distinguished. The networks continue training until the Discriminator cannot recognize the real data and generated ones. At this time, the Generator can obtain the data distribution and accurately predict the computing resource occupancy. Here, the Discriminator differentiates the entire sequence rather than the single resource utilization at episode $e+1$, y_{e+1} and $\tilde{y}_{e+1}$. This design better captures the correlation between y_{e+1} and the sequence information [13]. As (2) shows, the output of Long Short-Term Memory (LSTM) is input into a fully connected layer to obtain the prediction result in Generator,

$$G(y_r) = \tilde{y}_{e+1} = \delta(Wg(y_r) + b), \tag{2}$$

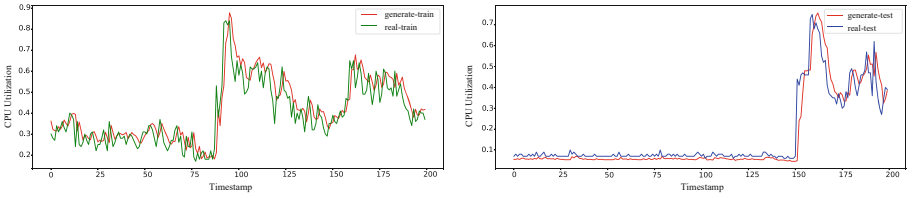
where $g(\cdot)$ is the output of LSTM, δ means the ReLU activation function, W and b are the weights and bias in the fully connected layer, respectively. A Multi Layer Perceptron (MLP) including three fully connected layers is regarded as the Discriminator and Leaky ReLU is applied as the activation function between hidden layers. As shown in (3), Sigmoid function is used in the output layer,

$$D(y_G) = \sigma(f(y_G)), D(y_{G'}) = \sigma(f(y_{G'})), \tag{3}$$

where $f(\cdot)$ is the output of connected layers and σ is activation function. Both networks are trained using Binary Cross Entropy Loss and the value function in (4) shows the difference between real resource utilization and generated ones,

$$\min_G \max_D V(G, D) = E(\log D(y_{G'})) + E[\log(1 - D(y_G))]. \tag{4}$$

Compared to the simple use of LSTM, this adversarial training method improves the accuracy of prediction. We set $F = 4$, using the resource occupancy rate of the previous 4 episodes to predict the next one. The real usage trace of 3 machines is chosen randomly from Cluster-trace-v2018[1], with a total of 64946 sequences. It is divided to a training and a test set according to the ratio of 8:2. The number of neurons in the hidden layer of Generator is 512. The initial historical resource utilization of GAN is randomly generated. As shown in Fig. 3, we select 200 prediction results in the training and test sets. It can be seen that GAN can accurately predict the resource utilization at the next moment.



(a) The results on training data (b) The results on test data

Fig. 3. Resource utilization prediction results by GAN

4.2 Resource Allocation Based on DRL

In our design, the DRL agent acts as the Resource Allocation Manager, and the Resource Monitor is treated as the environment. Based on the observed states of resource usage and task queue in the edge server, the agent makes resource allocation actions, determining the amount of allocated resources for each task. Then the Resource Monitor modifies the resource condition and feeds the reward back to the agent to adjust the resource allocation strategy accordingly.

[1] https://github.com/alibaba/clusterdata/.

State: The current state s is a triplet $s = (\mathcal{M}, \mathcal{Y}, \mathcal{Q}_A)$, where $\mathcal{M}$ represents the current computing resource matrix of the edge server, $\mathcal{Y}$ describes the predicted computing resource matrix of the next episode, and $\mathcal{Q}_A$ means the observation of the task sub-queue. The size of $\mathcal{M}$ is $K \times H$ and the size of $\mathcal{Y}$ is $K \times 1$. The output computing resource utilization of the GAN is $\tilde{y}_{e+1} \in [0, 1]$. We mark the values of the first $\tilde{y}_{e+1} \times K$ units as 1 to indicate the predicted resource occupancy condition at the next episode. The original dimension of $\mathcal{Q}_A$ is $A \times 4$. Each task contains four features including the number of instances, the carried data volume, the amount of requested computing resources and the deadline limitation. In order to unify the dimensions, we transpose $\mathcal{Q}_A$ and expand its dimension to $K \times 4$. Therefore, the dimension of state s is $K \times (H + 1 + 4)$.

Action: The action a indicates the amount of computing resources allocated to the task q_n per time unit. We filter the action values output by the network to ensure its effectiveness [8]. The value of action a_n must be a divisor of the requested resource c_n, that is $c_n(\mathrm{mod} a_n) = 0$. When using the ε-greedy algorithm, the action a_n randomly selects a value that be divisible by c_n. If the optimal value output by the network is not exactly divisible, we select the divisor closest to the output as the action. The resource occupation time d_n can be calculated correspondingly and the start time l_n is the least time offset under the condition that the available resource sub-matrix for allocation is all 0 s.

Reward: Our goal is to maximize the average resource utilization in a time period from a long-term perspective. Hence, after the task is allocated resources at time t, we use the current resource utilization within the sliding window of the edge server to represent the reward $r = u_t$.

The improved DQN includes three replay buffers with priority. We conduct resource utilization prediction at each episode instead of each timestep, avoiding frequent prediction of GAN, and update the sample priority only in current buffer, decreasing the time complexity of the algorithm. At timestep t, based on the current state s, task q_n is assigned resources according to action $a = a_n$, acquiring the next state s'. We store the transition (s', a, r, s) in these three buffers according to the value of predicted resource utilization $\tilde{y}_{e+1}$. Different values of ε is applied for the ε-greedy algorithm in various buffers. Besides, we adopt TD-error as the sampling priority to enhance the original uniform sampling method, which represents the difference between the action value output by the current value function and the estimated ones. Higher TD-error indicates that the output of the current value function is less accurate, implying that there is still more information can be learned and larger space for improvement in the prediction accuracy from this sample. So the sample will be assigned higher priority. Data structure SumTree is used to implement proportional prioritization, whose complexity is only $O(\log n)$ [24]. To ensure that every stored transition can be sampled, the new transition will be given a large priority.

Algorithm 1. Algorithm of RADEAN

1: Initialize Network Q with θ and Target Q-network Q' with $\theta' = \theta'$; Initialize replay buffers B_1, B_2, B_3 with same size ζ and different parameters ε_1, ε_2, ε_3; Mini-batch λ; Average reward R_{avg}, Reward sum R_{sum}; Average execution time P_{avg}; Execution time sum P_{sum}; Average transmission latency Z_{avg}; Transmission latency sum Z_{sum}; Timestep T; Task num N; Step ς; Edge server capacity K; Sliding window length H; Data transmission speed η; Task sub-queue length A;
2: Set initial values of GAN prediction windows
3: Put jobs into priority queue $\mathcal{Q}$ according to arriving time
4: **for** $e = 0$ to E **do**
5: obtain current task q_n at timestep t; N+=1
6: Task control according to ξ_n
7: get the predicted resource utilization $\tilde{y}_{e+1}$ from GAN
8: Choose replay buffer B_i according to $\tilde{y}_{e+1}$
9: Form current state $s = (\mathcal{M}, \mathcal{Y}, \mathcal{Q}_A)$
10: With probability ε_i choose a_n randomly
11: Otherwise $a_n = \arg\max Q(s, a; \theta)$
12: Filter and choose valid a_n
13: Get duration time $d_n = c_n / a_n$
14: **if** no enough resource in H for q_n **then**
15: Migrate q_n; $Z_{\text{sum}} += 2 \times (\rho_n / \eta)$
16: **else**
17: T+=1; Perform action $a = a_n$; Get reward $r = u_t$
18: Gain updated resource matrix $\mathcal{M}'$
19: $R_{\text{sum}} += r$; $P_{\text{sum}} += d_n$; $Z_{\text{sum}} += (\rho_n / \eta)$
20: Sliding window forwards one timestep
21: Form new state $s' = (\mathcal{M}', \mathcal{Y}, \mathcal{Q}_A)$
22: Store Transitions (s', a, r, s) to B_i
23: Sample a mini-batch of transitions (s_{j+1}, a_j, r_j, s_j) from B_i
24:

$$\mathcal{Q}_{\text{Target}} = \begin{cases} r_j, & \text{for terminal } s_{j+1}, \\ r_j + \gamma \max\limits_{a'} Q'(s_{j+1}, a'; \theta'), & \text{otw.} \end{cases}$$

25: Compute TD-error $\varphi_j = \mathcal{Q}_{\text{Target}} - Q(s_j, a_j, \theta)$
26: Update transition priority $p_j \leftarrow |\varphi_j| + \epsilon$
27: Perform a gradient descent step on Loss value
28: **end if**
29: Every ς steps, update Q' using $\theta' = \theta$
30: **end for**
31: **return** $R_{\text{avg}} = R_{\text{sum}} / T$; $P_{\text{avg}} = P_{\text{sum}} / T$; $Z_{\text{avg}} = Z_{\text{sum}} / N$;

In the improved DQN, the Target Q-network Q' computes target Q-value $\mathcal{Q}_{\text{Target}} = r + \gamma \max\limits_{a'} Q'(s', a'; \theta')$, where γ indicates the impact of further moments on present. The loss is calculated as $\mathcal{L}(\theta) = \mathbb{E}[(\mathcal{Q}_{\text{Target}} - Q(s, a; \theta))^2]$. As shown in Algorithm 1, we first conduct task control and apply GAN to predict the future resource utilization (Lines 6–7). After selecting the replay buffer and obtaining current state, the ε-greedy algorithm is used to perform exploration and

exploitation (Lines 8–11). An action is randomly generated under probability ε_i in replay buffer B_i, otherwise the action with the maximum value function is selected. The value of ε_i keeps decreasing during the training process. Then we filter the valid values to determine action a (Line 12). When the resource allocation decision of task q_n is determined, we judge if there are enough resources in the observation window for q_n (Line 14). If not, we migrate the task to a nearest selected server in other edge clouds or the remote cloud, during which the transmission time of the task is at least twice that of executed on the local edge server (Line 15). Otherwise, the action is performed, the resources are allocated. The agent obtains the corresponding reward and the status of system is modified (Lines 17–21). The transition (s', a, r, s) are distributed to different buffers according to the predicted resource utilization $\tilde{y}_{e+1}$ (Line 22). Every time we select a mini-batch with size λ to update the network, which performs iterative update with reduced complexity (Line 23). TD-error is computed to update the transition priority, and ϵ is used to ensure the value of transition priority is non-zero (Lines 25–26). The parameters of the Q network are utilized to update the Target-Q network every ς steps, which eliminates the divergence and oscillations during update (Line 29). Finally we compute and return the average resource utilization, average execution time and average transmission latency (Line 31).

5 Experiments

5.1 Experiments Setup

We deployed the simulation environment on the machine (2 CPU with 1 cores, type: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz, 48 GB memory) installed with Linux 5.4.0 system. All models are implemented by Python 3.8.10. Because preprocessed cloud task records can be used for experiments in edge [25], we select dataset Cluster-trace-v2018 (See Footnote 1), containing task execution and server status trace of 4000 machines in 8 days. We randomly choose the attributes of one server as a reference for edge server property settings. A job in this dataset contains multiple tasks, and each task includes multiple instances. The scheduling unit is task in our experimental design and five different test data sets of 10 jobs (50 tasks), 50 jobs (178 tasks), 100 jobs (357 tasks), 200 jobs (773 tasks), and 500 jobs (1827 tasks) are tested. In order to verify the performance of the RADEAN model, the following methods are selected as baselines for comparison.

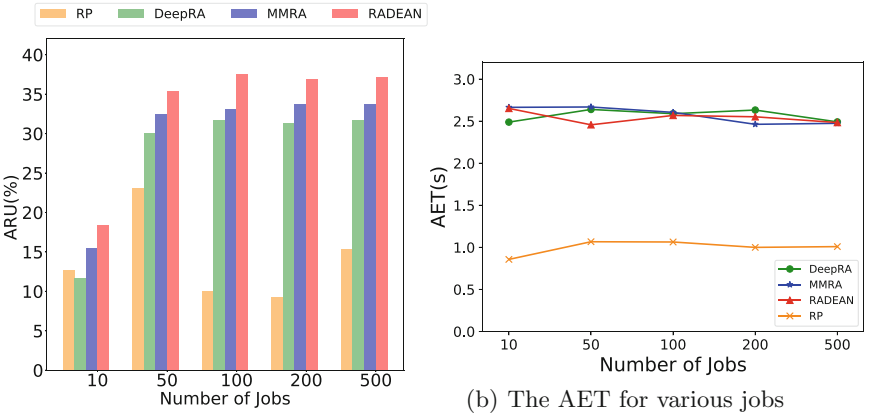
- Multi-Replay Memory-based Resource Allocation (MMRA): DRL with multiple replay buffers determines the resource allocation amount of tasks. The resource request of the next task is applied to calculate the resource utilization of the edge server at the next moment [8].
- Deep Reinforcement Learning-based Resource Allocation (DeepRA): It directly uses the ordinary DQN with a single replay buffer without priority for resource allocation and excludes resource prediction [9].

– Resource Precedence (RP): Within the scope of edge server computing resource capacity, it allocates the most available resources to meet the resource requests of tasks as many as possible [8].

5.2 Experiments Results and Analysis

Basic Experiments. Firstly, the computing resource capacity of the edge server is set to 48, half of the machine capacity in Cluster-trace-v2018 (See Footnote 1). Then we test the Average Resource Utilization (ARU) of different models to demonstrate the improved performance of RADEAN on resource utilization. At the same time, we present the Average Transmission Latency (ATL), the Average Execution Time (AET) and Migration Percent (MP) of tasks to measure the models' guarantee of QoS. All results are the average of five experimental operations.

It can be seen from Fig. 4(a), the ARU of RADEAN is highest for task sets of different scales. The effect of RP is the worst and the ARU of RADEAN is 4 times that of RP when job set is 200. Because RP allocates as many available resources as possible based on the task's request. At this time, the execution time of the task is the shortest, which confirms the result in Fig. 4(b) that the AET of RP is much lower than that of other models. The reason is that when the sliding window moves forward, the previous tasks may have been finished and will not occupy the resources in the current sliding window, gaining lower ARU. However, by using other three models, tasks will occupy several consecutive time units, thereby obtaining higher ARU and longer AET.



(a) The ARU for various jobs

(b) The AET for various jobs

Fig. 4. Experiment results when K is 48

DeepRA only makes resource decisions based on current resource status and task queue, ignoring the future resource utilization. RADEAN outperforms DeepRA by 58.08% on 10 job set and 17.08%–18.34% on others. Compared

with DeepRA, MMRA has better ARU performance, because MMRA not only considers future resource occupancy, but also uses multiple buffers to eliminate the interaction between experiences. But RADEAN still has a maximum performance improvement of 19.32% than MMRA when job set is 10. The reason is that the future resource utilization of MMRA is only obtained by the resource request of the next task in the task queue. It ignores the resource requests of subsequent tasks and the distribution of the resource utilization time series of the edge server, only accomplishing a short-term sub-optimal decision result.

RADEAN applies GAN to predict the resource utilization, considering previous a few timesteps of historical data comprehensively. It better mines the hidden information behind the training samples and captures the correlation between the resource utilization at the next moment and the previous time series. The adversarial learning between the Generator and Discriminator further improves the prediction accuracy. Moreover, RADEAN assigns different priorities to samples, which makes better use of rare and important experience information and achieves higher ARU in a long-term view.

As displayed in Fig. 4(b), apart from RP, RADEAN has the minimum AET on 50 jobs and has near AET with MMRA and DeepRA on other data sets. In Table 2, we see the ATL results of various models. RADEAN is closest to the optimal results of RP when job sets are 100 and 500, and achieves the best ATL on the rest. Hence, RADEAN attains excellent ARU with acceptable AET and ATL, ensuring the users' QoS and indicating its great generalization.

Table 2. The ATL and MP on various job sets

Cap.	Model	Job									
		ATL (s)					MP (%)				
		10	50	100	200	500	10	50	100	200	500
$K = 48$	RADEAN	**1.245**	**1.196**	1.669	**1.753**	1.849	**19.20**	**23.37**	62.96	**76.51**	85.90
	DeepRA	1.302	1.212	1.706	1.759	1.872	24.00	24.72	67.45	78.29	88.16
	MMRA	1.309	1.225	1.687	1.760	1.875	26.00	24.83	64.87	77.33	88.47
	RP	1.716	1.207	**1.554**	1.844	**1.823**	58.00	24.72	**52.10**	85.90	**83.14**
$K = 96$	RADEAN	1.196	1.048	1.390	1.521	1.784	14.40	7.98	36.25	53.68	79.29
	DeepRA	1.192	1.065	1.436	1.570	1.929	14.80	9.66	40.39	58.47	79.69
	MMRA	1.233	1.054	1.421	1.567	1.792	17.20	8.65	38.99	58.14	80.08
	RP	**1.061**	**0.965**	**1.202**	**0.993**	**1.767**	**2.00**	0.00	**18.21**	0.00	**77.39**

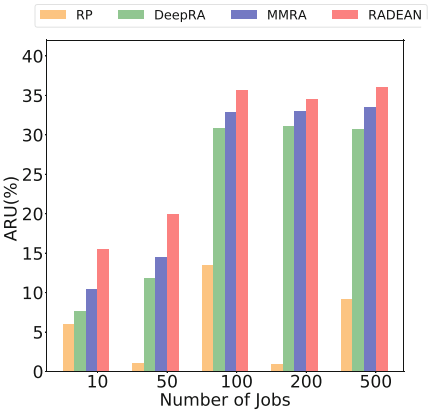
We also analyze the migration percentage of the tasks in Table 2. When K is 48, we see that the outcomes of RADEAN are near to the best results RP on job set 100 and 500, and RADEAN has the least migration amount in other cases. The increase in the task migration ratio leads to a growth in data transmission delay, which corresponds exactly to the ATL results in Table 2. RADEAN not only reduces ATL, ensuring users' QoS, but also decreases the migration of task data, cutting down the cost of task migration. We find that when the number of tasks is large enough, the resources of the edge server can no longer meet task requirements, which leads to a very high percent of task migration. Therefore, we

increase the resource capacity configuration of the edge server, and then proceed to verify the performance of the proposed model.

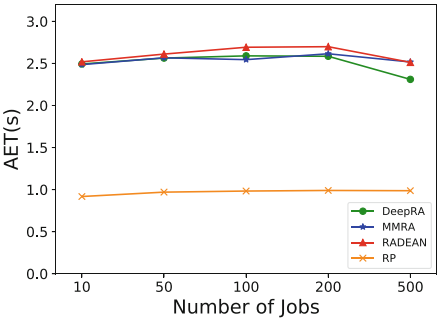
Table 3. New results on two new task sets

Task Num	ARU (%)	AET (s)	ATL (s)	MP (%)
178	5.31	0.966	1.024	0.56
773	13.07	0.998	1.213	21.34

Extended Experiments. We set the resource capacity of edge server to be 96, the same as the configuration of the selected machine in the data trace. In Fig. 5, it can be seen that the ARU of RADEAN is still higher than other baselines, achieving the same level of AET of tasks as other models except RP. Compared with MMRA, RADEAN has the largest performance improvement of 48.21% when job set is 10. Besides, when $K = 96$ in Table 2, RP has the least ATL. Because setting larger resource capacity satisfies the resource requests for more tasks when resource precedence mechanism is adopted, reducing task migration and ATL. RADEAN accomplishes the sub-optimal solution. Also, as Table 2 displays, the migration ratio of tasks decreases after we raise the edge server resource capacity. When the job numbers are 50 and 200, the ARUs of RP are extremely low in Fig. 5(a) and the migration percent of RP is 0 in Table 2. We analyze that this is because the resource requests of most tasks in these two data sets are far lower than the resource capacity of the edge server. Therefore, tasks are allocated with maximum resources and processed quickly, achieving very short execution duration time. There are few resources occupied by unfinished tasks in the sliding window, thereby gaining extremely small ARU. As Table 3 shows, we randomly sample another two new task sets with same size, and find the ARUs of RP increase, which verifies our explanation.



(a) The ARU for various jobs



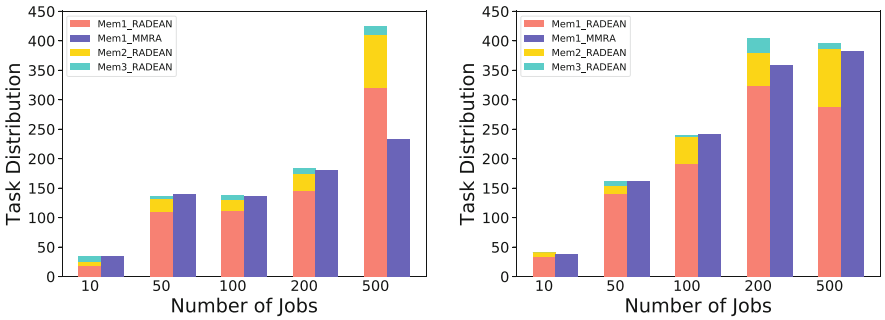
(b) The AET for various jobs

Fig. 5. Experiment results when K is 96

Besides, in Fig. 6, we display the sample distribution in various replay buffers when the numbers of migration in RADEAN and MMRA models are the least among five experimental results. The tasks performed by RADEAN are the same as or higher than MMRA's. Moreover, it can be seen that regardless of the capacity of computing resources, experiences will be stored in three reply buffers of RADEAN. While MMRA stores all samples in one of the three replay buffers in all cases. These results prove that the proposed model RADEAN distributes samples into various buffers more evenly compared with MMRA. And it better predicts the resource utilization of edge server from a long-term view, separates the experiences, and reduces the interaction between them.

Ablation Study. In order to verify the impact of multi-replay memory and sampling priority settings in RADEAN on resource allocation decisions, we conduct ablation experiments, which use GAN to predict the future resource utilization. The relevant models are described as follows:

- GM: Resource Allocation Model with GAN and Multiple Non-priority Replay Buffers (GM). It verifies the influence of the priority in replay buffers.
- GSP: Resource Allocation model with GAN and a Single Replay Buffer with Priority (GSP). It certifies the significance of multi-replay memory.
- GS: Resource Allocation Model with GAN and a Single Non-priority Replay Buffer (GS), proving the impact of both priority and multi-replay memory.

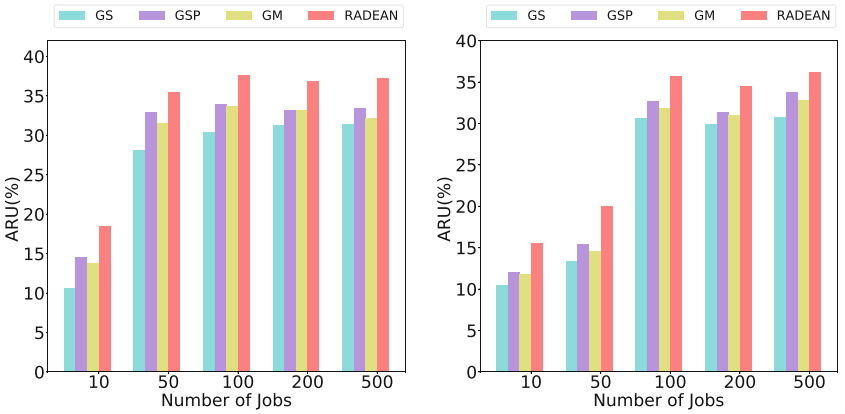


(a) The results when K=48 (b) The results when K=96

Fig. 6. Sample distribution on various cases

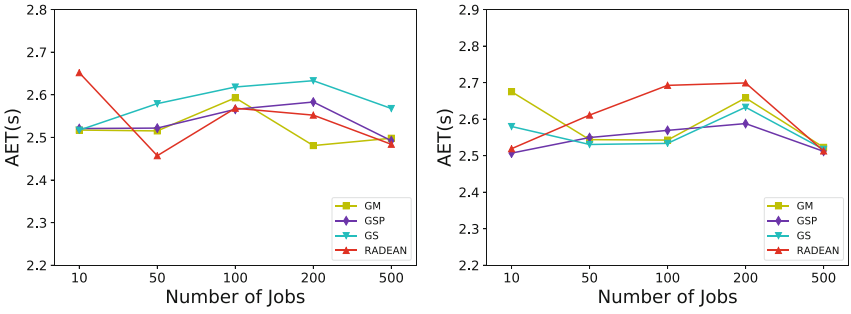
As shown in Fig. 7, GS has the worst effect and RADEAN has the best results. The maximum performance enhancement of RADEAN reaches 74.40% compared with GS when K=48 and job set is 10. This shows that setting multi-replay memory and adding priorities improve the performance of resource allocation. Multiple replay buffers reduce the mutual influence between samples, thereby enhancing the training effect. So RADEAN obtains higher ARU in the long-term view. By assigning different weights for samples, those experiences which

are more important and useful for resource allocation decisions have a greater chance of being sampled. This design further advances the model effect. The ARU of GSP is slightly higher than that of GM, indicating that priority has a higher enhancement than multi-replay memory. The performance improvements of GSP and GM are 4.55%–37.09% and 2.61%–30.26% compared with GS, respectively.



(a) The ARU result when K=48 (b) The ARU result when K=96

Fig. 7. The ARU ablation experiment results

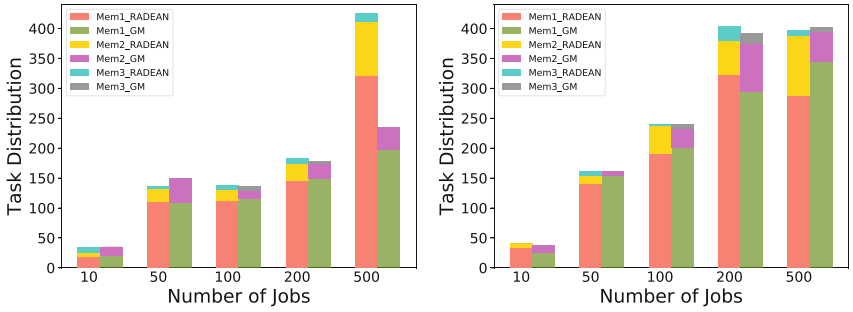


(a) The AET when K=48 (b) The AET when K=96

Fig. 8. The AET ablation experiment results

Table 4. The ATL and MP ablation results on various job sets

Cap.	Model	Job									
		ATL (s)					MP (%)				
		10	50	100	200	500	10	50	100	200	500
$K = 48$	RADEAN	**1.245**	**1.196**	**1.669**	**1.753**	**1.849**	**19.20**	**23.37**	**62.96**	**76.51**	**85.90**
	GM	1.277	1.209	1.687	1.763	1.866	22.00	23.82	64.76	77.64	87.56
	GSP	1.279	1.214	1.695	1.760	1.864	21.60	24.72	65.71	77.25	87.35
	GS	1.284	1.212	1.694	1.761	1.867	22.80	25.28	66.67	78.58	87.74
$K = 96$	RADEAN	**1.196**	**1.048**	**1.390**	1.521	1.784	**14.40**	**7.98**	**36.25**	**53.68**	**79.29**
	GM	1.215	1.067	1.441	**1.501**	**1.782**	16.00	10.45	40.62	54.13	80.14
	GSP	1.216	1.054	1.404	1.536	1.789	16.00	9.89	37.31	55.16	79.83
	GS	1.226	1.077	1.450	1.546	1.792	16.80	10.90	41.68	56.22	80.14



(a) The results when K=48 (b) The results when K

Fig. 9. Sample distribution ablation results

In Fig. 8, the AET of RADEAN is close to or slightly higher than other ablation models. But a higher ARU is realized with the acceptable time increase. From Table 4, we can see that RADEAN gets the smallest migration percent of tasks in all conditions and GS has the worst results. Also, RADEAN has the optimal ATL results almost in all conditions, and GS has the highest ATL. This is because GS lacks a multi-replay memory design, which leads to mutual influence between samples. The uniform sampling method of GS further reduces the effect of resource allocation decisions, leading to an increase in the MP of tasks and ATL. We also count the number of samples in different replay buffers when GM and RADEAN migrate the least number of tasks among the five experiments. As Fig. 9 displays, RADEAN distributes samples more evenly to different buffers compared with GM. What's more, RADEAN executes more tasks, which reduces the MP and migration cost of tasks.

6 Conclusion

Performing resource allocation efficiently of the edge server is of great significance both to users and service providers. In this paper, we have proposed a model

named RADEAN, which uses an improved deep reinforcement learning of multiple replay buffers with priority for resource allocation. Combined the resource utilization rate predicted by GAN, current resource usage and task queue status together, DRL makes resource allocation decisions for tasks. We have conducted experiments on real-word data. The results prove that RADEAN obtains higher average resource utilization with generalization than traditional and state-of-the-art methods under acceptable task execution time and transmission latency, reaching the maximum performance improvement of 48.21% over MMRA. The ablation experiments confirm the effectiveness of the multi-replay memory and priority settings to improve the resource allocation performance.

References

1. Alnoman, A., Sharma, S.K., Ejaz, W., et al.: Emerging edge computing technologies for distributed IoT systems. IEEE Netw. **33**(6), 140–147 (2019)
2. Anu, A.S.: Resource allocation in fog computing based on meta-heuristic approaches: a systematic review. IJCSNS **22**(9), 503 (2022)
3. Zakarya, M., Gillam, L., Ali, H., et al.: epcAware: a game-based, energy, performance and cost-efficient resource management technique for multi-access edge computing. IEEE Trans. Serv. Comput. **15**(3), 1634–1648 (2020)
4. Zhang, Y., Zhang, M., Fan, C., et al.: Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment. EURASIP J. Adv. Sig. Process. **2021**(1), 33 (2021)
5. Dlamini, T., Vilakati, S.: LSTM-based traffic load balancing and resource allocation for an edge system. Wireless Commun. Mob. Comput. **2020**, 1–15 (2020)
6. Liu, T., Ni, S., Li, X., et al.: Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing. IEEE Trans. Mob. Comput. **22**(7), 3870–3881 (2023)
7. Sharif, Z., Jung, L.T., Razzak, I., et al.: Adaptive and priority-based resource allocation for efficient resources utilization in mobile edge computing. IEEE Internet Things J. **10**(4), 3079–3093 (2021)
8. Xiong, X., Zheng, K., Lei, L., et al.: Resource allocation based on deep reinforcement learning in IoT edge computing. IEEE J. Sel. Areas Commun. **38**(6), 1133–1146 (2020)
9. Lakhan, A., Mohammed, M.A., Obaid, O.I., et al.: Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm. Autom. Softw. Eng. **29**, 1–25 (2022)
10. Bhajantri, L.B., Gangadharaiah, S.: Heuristic-based resource allocation for Internet of Things in gateway centric multi-layer fog computing. In: Tuba, M., Akashe, S., Joshi, A. (eds.) ICT4SD 2022. LNNS, vol. 516, pp. 567–579. Springer, Singapore (2022). https://doi.org/10.1007/978-981-19-5221-0_54
11. Chen, Y., Li, Z., Yang, B., et al.: A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing. Future Gener. Comput. Syst. **108**, 273–287 (2020)
12. Haverkort, B.R., Finkbeiner, F., de Boer, P.T.: Machine learning data center workloads using generative adversarial networks. ACM SIGMETRICS Perform. Eval. Rev. **48**(2), 21–23 (2020)
13. Zhang, K., Zhong, G., Dong, J., et al.: Stock market prediction based on generative adversarial network. Procedia Comput. Sci. **147**, 400–406 (2019)

14. Jamil, B., Ijaz, H., Shojafar, M., et al.: Resource allocation and task scheduling in fog computing and internet of everything environments: a taxonomy, review, and future directions. ACM Comput. Surv. (CSUR) **54**(11s), 1–38 (2022)
15. Bittencourt, L.F., Diaz-Montes, J., Buyya, R., et al.: Mobility-aware application scheduling in fog computing. IEEE Cloud Comput. **4**(2), 26–35 (2017)
16. Lin, J., Huang, L., Zhang, H., et al.: A novel Lyapunov based dynamic resource allocation for UAVs-assisted edge computing. Comput. Netw. **205**, 108710 (2022)
17. Huang, X., Gong, S., Yang, J., et al.: Hybrid market-based resources allocation in mobile edge computing systems under stochastic information. Future Gener. Comput. Syst. **127**, 80–91 (2022)
18. Naha, R.K., Garg, S.: Multi-criteria-based dynamic user behaviour-aware resource allocation in fog computing. ACM Trans. Internet Things **2**(1), 1–31 (2021)
19. Wang, Z., Lv, T., Chang, Z.: Computation offloading and resource allocation based on distributed deep learning and software defined mobile edge computing. Comput. Netw. **205**, 108732 (2022)
20. Wei, Y., Yu, F.R., Song, M., et al.: Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning. IEEE Internet Things J. **6**(2), 2061–2073 (2018)
21. Lee, S., Lee, S.K.: Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. IEEE Internet Things J. **7**(10), 10450–10464 (2020)
22. Udayakumar, K., Ramamoorthy, S.: Intelligent resource allocation in industrial IoT using reinforcement learning with hybrid meta-heuristic algorithm. Cybern. Syst. **54**(8), 1241–1266 (2022)
23. Math, S., Tam, P., Kim, D.Y., et al.: Intelligent offloading decision and resource allocations schemes based on RNN/DQN for reliability assurance in software-defined massive machine-type communications. Secur. Commun. Netw. **2022**, 4289216 (2022)
24. Schaul, T., Quan, J., Antonoglou, I., et al.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
25. Liu, L., Tan, H., Jiang, S.H.C., et al.: Dependent task placement and scheduling with function configuration in edge computing. In: Proceedings of the International Symposium on Quality of Service, pp. 1–10 (2019)

Networks



A Stream Data Service Framework for Real-Time Vehicle Companion Discovery

Zhongmei Zhang[1(✉)] and Shuai Zhang[2]

[1] School of Management Engineering, Shandong Jianzhu University, Jinan 250101, Shandong, China
zhangzhongmei20@sdjzu.edu.cn

[2] Jinan Branch of China Unicom Software Research Institute, Jinan 250100, Shandong, China
zhangs837@chinaunicom.cn

Abstract. Because of the large-scale and complexity nature of vehicle trajectory data, existing methods have struggled to guarantee the efficiency and effectiveness of vehicle companion discovery in real-time. This paper proposes a stream data service framework to real -time discover vehicle companion. It relaxes the spatial and temporal constraints of vehicle companion definition to find more potential companion vehicles. And to ensure the performance, we make use of flexible service collaboration to process data generated by related monitoring sites selectively, and employ a stream partition strategy to realize service collaboration in parallel. Experiments conducted with real and simulated data demonstrate that our method can identify a greater number of potential vehicle companion sets while requiring less transmission and processing resources.

Keywords: Stream Data Service · Trajectory Data · Vehicle Companion · Service Collaboration

1 Introduction

Recently, with the rapid development of GPS, Mobile Smart Devices, RFID and other technologies, a vast amount of trajectory data with space-time attributes have been produced, capable of recording the movement of vehicles, people, animals and natural phenomena in social networks, traffic management, scientific research and military surveillance and other applications. Researchers expect to find various companions in moving groups [1–4] based on processing trajectory data. The study of vehicle companion can discover the movement patterns of vehicles that travel together within specific time range, which can provide good solutions for accident investigation [5], group tracking [6], and carpooling recommendation [7].

Currently, the companion vehicle discovery problem is mainly based on two kinds of vehicle companion's definition: (1) time point-based companion [8–14], which is applied to Global Positioning System (GPS) trajectory data. As GPS equipment may be only installed in limited equipment, and may not satisfy specific mining requirements.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 281–296, 2024.
https://doi.org/10.1007/978-3-031-63989-0_14

(2) spatial point-based companion [7, 15–17], which is applied to Automatic Number Plate Recognition (ANPR) data. Contrasted with GPS data, ANPR data offers the capability for continuous monitoring across significantly larger areas, potentially spanning entire cities. To summarize, existing methods are limited to adjusting either temporal or spatial constraints for vehicle companion, potentially overlooking companion vehicles that fall within broader constraints and thereby impacting the overall completeness of identified companions. Besides, existing work primarily focusses on static data sets and rely on centralized methodologies, which results an increase in communication and computational costs. Hence, it is quite important to provide a high-quality and low-cost method to discover vehicle companion in real-time based on amount of trajectory data stream.

Due to the reusability and interoperability of service, many researchers [18–20] have integrated Service-Oriented Architecture (SOA) technologies into Internet of Thing systems to decouple low-level streaming data from upper-layer applications. The servitization of trajectory data can help developers discretely process trajectory data through flexible management of software components. In our previous studies [21, 22], we introduced a proactive data service as a form of service abstraction, enabling flexible service collaboration in the context of dynamic stream data.

For discovering more complete companion vehicles, we proposed a new vehicle companion definition in our work [23]. In this paper, we aim to propose a stream data service framework based on ANPR data to process real-time trajectory data more flexibly and efficiently. We provide a service collaboration-based method to find vehicle companion, and examine a distributed strategy to realize service collaboration in parallel. We firstly encapsulate trajectory data produced by each monitoring site into fine-grained services, then build more powerful services based on dynamic service collaboration, and generate the vehicle companion sets in gradually and discrete way. In summary, this paper has followed contributions:

- we redefine vehicle companion by loosening the spatial constraints based on existing spatial point-based companion, resolve the issue of vehicle redundancy, and can find more potential companion vehicles.
- instead of accessing and processing all trajectory stream data, our proposed service collaboration-based method dynamically accesses and processes relative trajectory data based on the vehicles' real-time status. And it can ensure efficiency due to largely reducing of the data processing volume.
- to further increase the efficiency of our method, the service framework employs a stream partition strategy and run the service collaboration in a distributed processing environment. It can divide monitoring sites into partitions and realize vehicle companion discovery in parallel.

The rest of the paper is structured as follows: Sect. 2 introduces related work. Section 3 presents a motivational scenario and definitions of the problem. Section 4 introduces our method for vehicle companion discovery and outlines our parallel framework. Section 5 presents the results of our experiments. Lastly, in Sect. 6, we conclude and discuss avenues for future research.

2 Related Work

2.1 Stream Data Service

With the fast development of stream data researches, researchers have explored integrating SOA [18–20] technology into stream data applications, leading to two kinds of approaches based on the core concept of service models.

One kind of approach center on treating "resource" as the core concept [24–27], where sensors and their generated streams are viewed as shareable resources with the external world. Many studies [26, 27] have aimed at leveraging lightweight Web service protocols to enable the servitization of sensor data, effectively transitioning IoT into the WoT (Web of Things). These works realize sharing and interoperability of sensor stream data by using open Web standards. The other approach [7, 28–30] takes "function" as the core concept, focusing on providing a service model centered around stream data processing. These studies encapsulate stream data processing functions into services, allowing applications to readily utilize general or complex stream data processing capabilities through service invocation.

Overall, existing works in stream data servitization have proven effective in enhancing the sharing and reusability of stream data across diverse user scenarios. The encapsulation of stream data processing capabilities streamlines the development of complex application systems by enabling efficient service composition and collaboration. Moreover, adapting service composition processes to dynamic operational environments or data requirements has been addressed [31–33], aiming to alleviate user burdens and actively manage complex and dynamic data processing demands.

2.2 Companion Patterns Discovery

Many works were proposed obvious definitions and method to explore companion pattern. There are some typical companion pattern definitions such as Flock [8, 9], Convoy [10, 11], Swarm [12, 13], Platoon [14, 15], etc. The Flock [8, 9] definition was among the earliest proposed companion pattern definitions, specifying that companion sets must appear simultaneously within a circular geographic space across continues time points. Convoy [10, 11] extended the spatial constraints to include density-reachable geographic areas, building upon the foundation established by Flock. Swarm and Group [12, 13] relaxed Convoy's time constraint, enabling companion sets to be simultaneously present in density-reachable geographic spaces at multiple discrete time points. Platoon [14, 15] further refined the time constraints by specifying thresholds for the number of time points during which companion groups co-occurred, as well as the number of consecutive time points during which companion sets co-occurred. Platoon required the companion sets to co-occur in density-reachable geographic spaces during enough continuously time points. These works were focused on companion patterns at specific time points, and were more suitable for stream generated by floating sensors, such as GPS data.

In addition to the above definitions, many studies [7, 16, 17] were focused on data produced by fixed sensors, and regarded moving objects that occur successively or simultaneously at the same spatial point within a given time range as site companion. Based

on the site companion, moving objects occurred in multiple site companion relations are considered as a companion set.

In recent years, many work tried to focus on the efficiently of discovering co-occurrence patterns from large-scale of trajectory data. Zhu et al. [15] introduced the PlatoonFinder algorithm for real-time mining the Platoon companion patterns from automatic number plate recognition data streams, which adopted pseudo-projection technologies for effectively coping with the speed and scale of data streams. Xiao et al. [17] integrated streaming data with dynamic graph calculations to identify vehicle companion in real-time scenarios. Zhang et al. [34] introduced a distributed framework for mining companion patterns from data streams, aiming to enhance the discovery capability and performance. Currently, most of existing research is centralized, requiring substantial processing of stream data.

The ANPR data focused on this paper was generated by fixed traffic camera sits. Therefore, we aim to propose a stream data service framework in the basic of improved site companion, and we try to encapsulate traffic stream data into proactive data services, and realize dynamic service collaboration to efficiently find vehicle companion sets in parallel.

3 Scenario and Problem Definition

We firstly introduce a typical scenario about vehicle companion discovery.

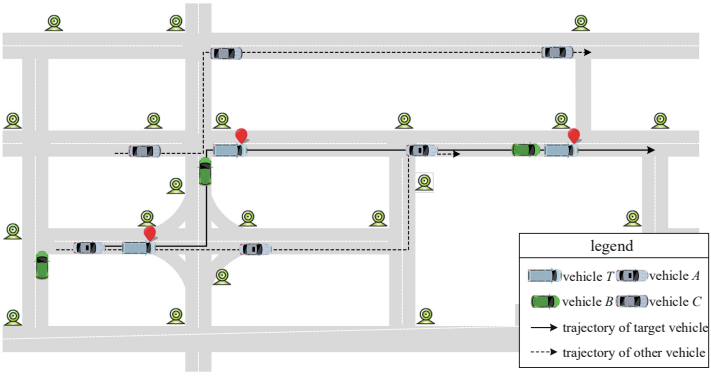


Fig. 1. Examples of vehicle companions based on ANPR data.

Crimes special targeting vehicles such as bus, taxis, armored car, and vehicles transporting dangerous or important goods have become increasingly prevalent. By promptly detecting suspicious tracking vehicles and alerting the tracked vehicle, occurrences of such crimes can be effectively prevented when a specific vehicle passes through a traffic camera. Figure 1 illustrates some examples of vehicle companions. The discovery of the vehicle companions for target vehicles in real time serves as an effective way to identify suspicious tracking vehicles.

In existing works, vehicles that pass the same site in given time range with target vehicle (vehicle T) several times can be identified as its companion vehicles. Figure 1 shows that vehicle B and target vehicle T passed by the same sites within a close time range, indicating identical trajectories. While in practice, tracking vehicles often opt not to follow the exact same trajectory as the target vehicle to avoid easy detection. For example, although vehicle A and vehicle C in Fig. 1 had different trajectories with target vehicle T, the monitoring sites they passed through are adjacent to the sites that vehicle T passed through, making them potential suspicious tracking vehicles. Therefore, solely relying on identical sites as the criterion for identifying vehicle companions may not fully uncover all companion vehicles.

For avoiding missing some potential companion vehicles, we redefine vehicle companion pattern by loosening the spatial constraint of spatial point-based companion pattern. We firstly give a definition about area companion based on site companion [7], which indicates vehicles passed through exactly the same monitoring site or spatial point within given time range.

Definition 1. Area Companion. An areas companion set a_k can be represented as $a_k = \{<v, c, t>\}$, is as set of triples including vehicle v, the monitoring site c passed by v, and the time point t that vehicle v passed monitor site c. And for any two triples $<v_i, c_i, t_i>$ and $<v_j, c_j, t_j>$ in a_k, exists $|t_i - t_j| <= \delta_t$, where δ_t is a given time range threshold, and c_i and c_j have adjacent relationship.

Figure 2 show some example of site companion and area companion. In Fig. 2, five monitoring sites are shown as the vertical axis, and it can be seen that there is a site companion set $\{v_1, v_2, v_3\}$ at monitoring site A. While area companion refers to vehicles passing nearby monitoring sites at similar time points. Based on Definition 1, if monitoring site A and B in Fig. 2 have adjacent relations, vehicles $\{v_5, v_7\}$ and $\{v_1, v_2, v_3\}$ will also have companion relation.

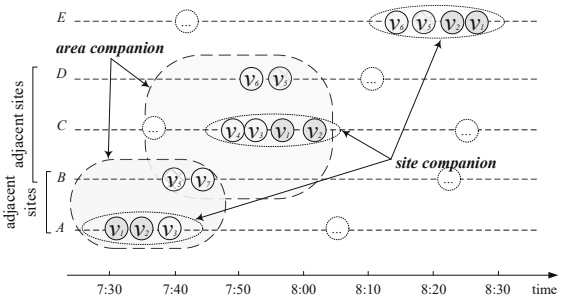


Fig. 2. Examples of Site Companion, Area Companion and Vehicle Companion

It is important to note that because of the loosen of temporal and spatial constraints in area companion, redundancy of vehicles and monitoring sites may occur across different sets of area companions. For example, v_5 appears twice with different sites in one area companion as shown in Fig. 2.

Vehicles that occurred in the same area companion several times can be regard as vehicle companions.

Definition 2. Vehicle Companion. An vehicle companion set means a set $vc = \{v\}$, for two vehicles v_i and v_j in it, their companion degree $cd(v_i, v_j) <= \delta_{com}$, where

- $cd(v_i, v_j) = \sum p(c_{ik}, c_{jk})/max(n_i, n_j)$, in which n_i and n_j are the numbers of monitoring sites passed by vehicle v_i and v_j, and $p(c_{ik}, c_{jk})$ is the adjacent degree of the monitoring sites when they appear in the k-th area companion.
- $\delta_{com} \in (0,1]$ is a given threshold.

For a given vehicle, we can obtain more complete companion vehicle set based on above definitions. Taking Fig. 2 as an example, assuming that monitoring sites $\{A, B\}$ have respectively adjacent relationships with sites $\{C, D\}$, and giving δ_{com} as 0.5, for target vehicle v_1, sets $\{v_2, v_3, v_5, v_7\}$, $\{v_2, v_3, v_4, v_5, v_6\}$ and $\{v_2, v_5, v_6\}$ are all area companion sets with v_1. And the vehicles have vehicle companion relationship with v_1 are in set $\{v_2, v_3, v_5, v_6\}$.

Traditional methods predominantly employed a centralized processing way to identify vehicle companions. These methods involved gathering vast amounts of stream data into a single data center, followed by analysis and processing rely on various discovery methods and companion definitions, and required significant transmission and processing resources. Therefore, this paper aims to address the challenge of efficiently processing stream data while ensuring the effectiveness and performance of vehicle companion discovery.

4 Method

Based on the definition of vehicle companion introduced above, we firstly introduce our service collaboration method to flexibly and efficiently find companion vehicle for given vehicle. Then we introduce our parallel strategy in the service framework which can further ensure the performance of our method.

4.1 Vehicle Companion Discovery Based on Service Collaboration

Figure 3 illustrates the schematic of our service collaboration-based method. This paper creates three types of stream data services, i.e., Site Companion Discovery (SCD) services that corresponding to the monitoring sites, several Area Companion Discovery (ACD) services, and one Vehicle Companion Discovery (VCD) service.

To be specific, SCD service is mainly in charge of monitoring of target vehicle and generate site companion vehicles in real-time. ACD service is in charge of receiving site companion results from SCD services and discovering area companion results. And VCD service receives trajectory information from SCD services to obtain the completed trajectory, and receives area companion information from corresponding ACD services to generate final vehicle companion sets.

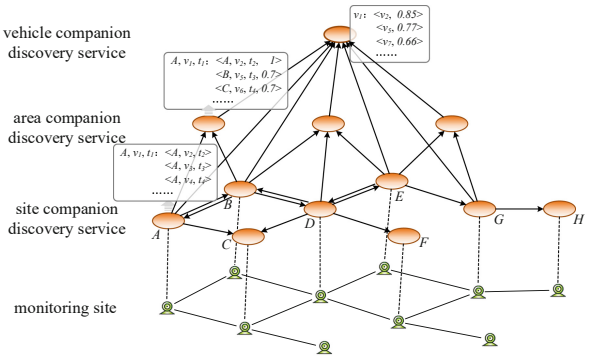


Fig. 3. Overview of vehicle companion discovery method.

When an SCD service monitors that given vehicle passed the service's corresponding site, it will target a series of actions, i.e., the site companion discovery function of this service, starting of SCD services corresponding to nearby monitoring sites, and starting one ACD service. The detail execution process of each SCD service is shown in Algorithm 1.

Algorithm 1. execution process of SCD service.
Input:
given vehicle v_i;
time threshold δ_t;
spatial threshold δ_s.
Output:
vehicle v_i discovery event e_i.

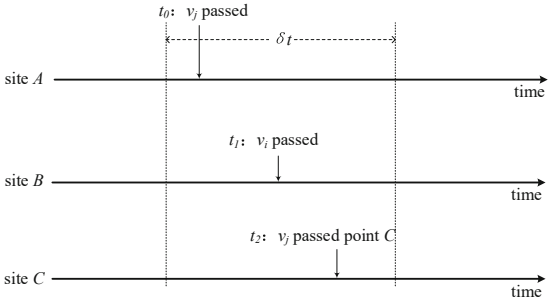
1. start procedure *VehicleMonitor*(v_i);
2. if discover e_i
3. start procedure *SiteCompanionMonitor*(e_i, δ_t);
4. get neighbor ACD services based on δ_s;
5. send e_i , δ_t, δ_s to neighbor SCD services;
6. send e_i to ACD service;
7. send e_i to VCD service;
8. stop procedure *VehicleMonitor*(v_i);
9. end if

For reducing the volume of trajectory data to process, at the initial stage, it needs all SCD services to monitor target vehicle v_i only (line 1). Once one service discovers vehicle v_i is passed by, an event e_i which contain vehicle's id, site's id and passed time is generated (line 2), and the area companion discovery process *SiteCompanionMonitor* will be started (line 3). Each service will contain a buffer to obtain vehicles passed corresponding site in certain time range according δ_t and e_i, and the *SiteCompanionMonitor* will stop automatically at appointing time. Nearby SCD services will be targeted to continue monitor given vehicle and generate other site companions (line 5), and the number of SCD services is determined by spatial threshold (line 4). One ACD service and VCD

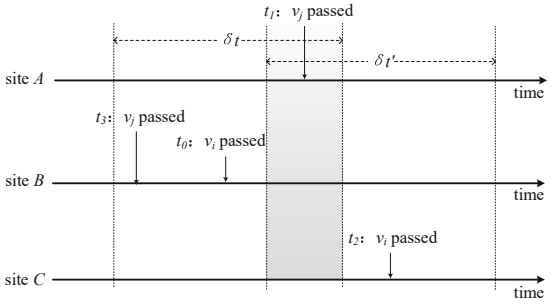
service are also targeted to further generated area companion and vehicle companion (line 6, 7). And the monitoring of vehicle v_i is no need (line 8).

To be mentioned, VCD service also in charge of stop all the monitoring procedure of irrelevant SCD services at the initial stage once one SCD services find target vehicle's first appearance. Except the procedure shown in Algorithm 1, each SCD service also executes two other procedures, i.e., monitoring events from VCD service that target vehicle is appeared, and keeping monitoring events from other SCD service to target the procedure of Algorithm 1.

One ACD service will be called once target vehicle passes one monitoring site, and it will collect all site companions from corresponding SCD services to generate area companion. As mentioned above, there may be redundancy in the area companion sets. Figure 4 shows some example of two types of area companion redundancy.



(a) redundancy in the same area companion



(b) redundancy in different area companions

Fig. 4. Examples of area companion redundancy.

As shown in Fig. 4(a), the first kind of redundancy is redundancy in the same area companion. In Fig. 4(a), monitoring sites A and C are adjacent sites of monitoring site B. When vehicle v_i passes through monitoring site B at t_1, all vehicles passing through monitoring site A and C within given time range threshold δt have area companion relationship with vehicle v_i, such as $\{<v_i, B, t_1>, <v_j, A, t_0>\}$. Since vehicle v_i also passed through site C within δt, there is also a companion relationship as $\{<v_i, B, t_1>, <v_j, C, t_2>\}$, which resulting repeating vehicle v_j in the same area companion.

Figure 4(b) shows one example of redundancy in different area companions. In Fig. 4(b), monitoring site A has respectively adjacent relationships with monitoring site B and set C. When vehicle v_i passed monitoring site B and site C respectively in time t_0 and t_2, there may be an interleaved time range between monitoring sites A and C as the dash area in Fig. 4(b). Vehicles passed by certain sites within this interleaved time range can both have area companion relationships with $<v_i, B, t_0>$ and $<v_i, C, t_2>$. For example, $<v_j, A, t_1>$ is both in area companion $\{<v_i, B, t_0>, <v_j, A, t_1>\}$ and $\{<v_i, C, t_2>, <v_j, A, t_1>\}$. To be mentioned, it is also happened that one vehicle is both belong to these two kinds of redundancy. For example, vehicle v_i in Fig. 4(b) is also shown in area companion $\{<v_i, B, t_0>, <v_j, A, t_2>, <v_j, A, t_3>\}$ twice.

The redundancy vehicles need to be removed for avoiding affection to vehicle companion discovery. However, the removing of vehicles may also affect the final result. Taking Fig. 4(b) as an example, if we remove $<v_j, A, t_3>$ from area companion $\{<v_j, A, t_2>, <v_j, A, t_2>, <v_j, A, t_3>\}$, there is still exists redundancy in area companion $\{<v_i, B, t_0>, <v_j, A, t_1>\}$ and $\{<v_i, C, t_2>, <v_j, A, t_1>\}$. And if we further remove $<v_j, A, t_1>$ from $<v_i, B, t_0>, <v_j, A, t_1>\}$. It results that the co-occurrence number of vehicle A and B is reduced to one from two. Hence, instead of removing redundancy of the same companion in ACD services, we keep all the redundancy and process them together in VCD service. To avoid above problem, if there are vehicles both belong to two kinds of redundancy, we firstly remove ones that in different area companions.

In summary, our method leverages the dynamic collaboration of SCD and ACD services to extract trajectory and site companion sets for a given vehicle by processing only relevant stream data, and send the information to the VCD service. Based on Definition 2, the VCD service can eliminates redundancy in area companionship, generates vehicle companion sets in real-time by receiving trajectory and area companion information, and continuously provides them as output.

4.2 Parallel Framework

Figure 5 shows our framework that can create proactive data service, and support dynamically service collaboration for satisfying users' requirements.

To shield users from the details about the data stream source and offer various representations, our framework adopts the Representational State Transfer (REST) architectural style. Since traditional Web services, with their "request-and-response" pattern, may not effectively facilitate developers in retrieving data streams. We utilize a Web push technology, i.e., Web Socket, and define declarative rules to realize stream data service based on our proposed proactive data service abstract [20, 21]. With Web Socket, stream data service can proactively transmit events containing data records to users over the web. By utilizing predefined declarative rules, users can dynamically get events that satisfy their specific requirements.

For ensuring the performance of our service and service composition, we utilize could environment and design a partitioning strategy to process the stream data service in parallel. We employ a straightforward approach of dividing the monitoring sites into several partitions and processing the corresponding services in parallel. However, several considerations must be taken into account when dividing the sites into partitions. Firstly, to prevent data skew among worker nodes, it is essential to evenly distribute the data

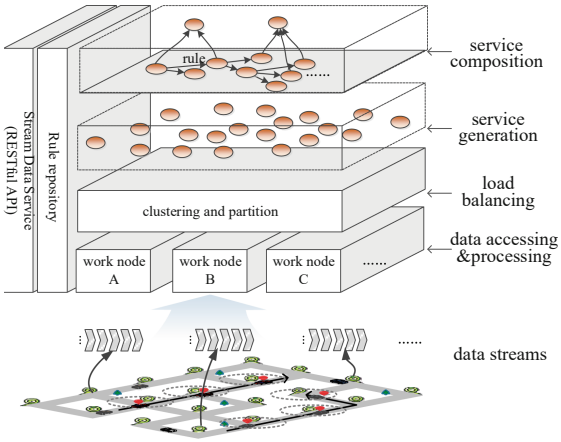


Fig. 5. The architecture of proposed framework.

volume across each partition. Secondly, to minimize communication overhead between workers, neighboring monitoring sites should ideally be grouped within the same set. For solving the above issues, this paper proposes a stream partition strategy and realized it based on the improvement of the Number Partition Problem (NPP) [35].

Definition 3. Stream Partition Strategy. Given a data stream S produced by a set of monitoring sites, the stream partition strategy sp can be regarded as a partition function $sp(S) = \{S_1, S_2 \ldots S_k\}$, s.t.

(1) $S = \bigcup_{i=1\ldots k} S_i$;
(2) $S_i \cap S_j = \emptyset, i, j = 1 \ldots k \cap i \neq j$;
(3) For any data stream $s \in S_i$, $\exists s' \in S_i$ and s' is the data generated by neighboring site which directly connect with the site generated s;
(4) min $\left(\sum_{i,j=1\ldots k}^{k} \left(|S_i| - \left|S_j\right|\right)\right)$, where $|S_i|$ means the number of data streams in set S_i.

It should be noted that condition (3) and (4) may not always be satisfied simultaneously. In this paper, we prioritize satisfying condition (3), which can ensure that there are no isolated sites within one partition set, and then we satisfy condition (4) as possible to ensure the balance of different partitions. The relationship of SCD services can be represented by monitoring sites' relations. Firstly, we traverse the graph breadth-first, and then partition the services into k partitions.

5 Experiments

We will respectively evaluate the efficiency and effectiveness of our discovery method by comparing with PlatoonFinder [15], which was realized based on spatial point companion and adopted centralized processing method.

5.1 Experiment Setup

1. **Datasets**
   The datasets used in our experiment is the real ANPR data from municipal traffic administration bureau of three large cities in China. In each data record, the data includes information about the vehicle, the monitoring site, the timestamp, and other additional data. The detail of data sets is show in Table 1. In the experiment, we convert real ANPR data into streaming data based on the generation time of each record, enabling real-time discovery of vehicle companions.

Table 1. Detail of data sets

datasets	Number of monitoring sites	Number of vehicles	Size
City 1	1794	2.3 million	25.7 GB
City 2	1040	1.7 million	18.9 GB
City 3	985	1.6 million	17.4 GB

2. **Experiment environments**
   Our method is implemented in a cluster consisting of 5 nodes, each running on virtual machines with CentOS release 6.4 and java 1.80. Table 2 shows the detailed configuration information of our cluster.

Table 2. Configuration of the Cluster

IP of node	CPU	Memory
10.61.6.153	Intel Xeon E312xx (Sandy Bridge)	6G
10.61.6.154	Intel Xeon E312xx (Sandy Bridge)	6G
10.61.6.155	Intel Xeon E312xx (Sandy Bridge)	3G
10.61.6.156	Intel Xeon E312xx (Sandy Bridge)	3G
10.61.6.157	Intel Xeon E312xx (Sandy Bridge)	3G

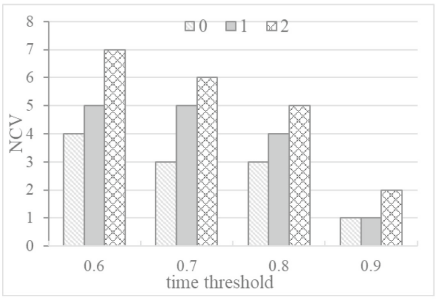
3. **Experiment criteria**
   We used (1) the Number of discovered Companion Vehicles (NCV) to verify the effectiveness; (2) Data Volume Processed per second (DVP) and (3) Average Response Time (ART) to verify the efficiency.

   The DVP in this paper means per second's received data volume from the stream data source side to the service side. And the ART is the average time difference between data sending time from data source side and the completion time of the data processing.

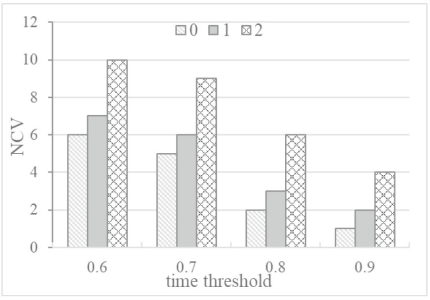
5.2 Effectiveness

Firstly, we conduct the discovery of vehicle companions using the same historical datasets, and compare the real-time discovered companion vehicles of our method by adjusting various thresholds. We randomly select 50 target vehicles from three data sets respectively, and set the time range threshold as 5 min, spatial threshold as 0, 1, 2, and the companion degree threshold as 0.6, 0.7, 0.8, 0.9. Figure 6 shows average NCV for three data sets.

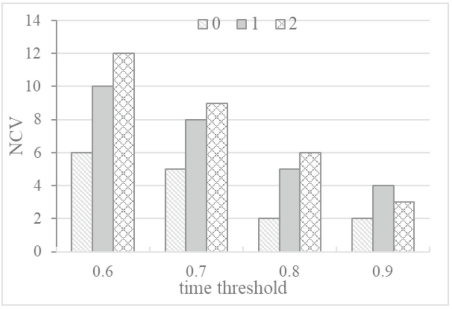
In fact, the results of our method shows that we can discover accurately the same companion vehicles with the results based on the same history trajectory data sets, which indicate that our real-time method can discover vehicle companion accurately. To simplify the figure, we only show one set of results in Fig. 6. It can also be observed that as the spatial threshold increases, more companion vehicles can be identified.



(a) city 1



(b) city 2



(c) city 3

Fig. 6. The average NCV for three data sets with different threshold.

Then, we compare our method with PlatoonFinder. We randomly selected 10 vehicles as target vehicles to respectively discover their vehicle companion sets. We respectively set the spatial threshold of our method as 2, the minimum companion degree threshold as 0.6, the time range threshold as 5 min, and record the average companion vehicles' numbers at given vehicles' 4, 6, 8, 10th passed monitoring sites to show continuously effectiveness of different methods.

Figure 7 illustrates the comparison of the detected companion vehicles' number of different methods. It is evident that our method consistently identifies more potential

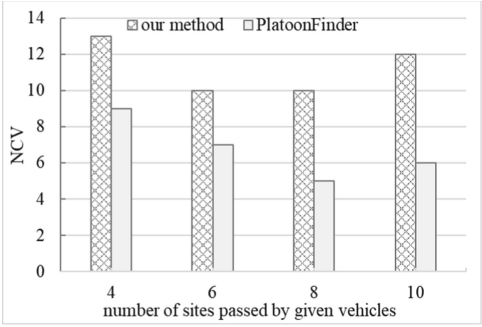


Fig. 7. Comparison of the companion vehicles' number.

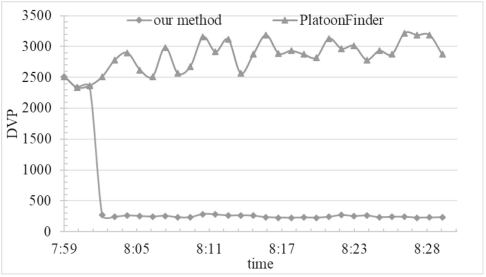
companion vehicles given the same vehicle movements. This is attributed to our method's relaxation of the spatial constraints for vehicle companionship, allowing for the discovery of vehicles passing through adjacent areas in given time range threshold. In essence, PlatoonFinder is equivalent to our method with a time threshold of 0.

5.3 Performance

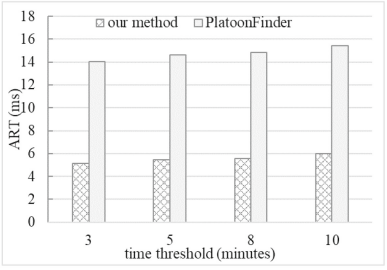
For evaluating the performance, we firstly compare our method with PlatoonFinder method. We set the minimum companion degree threshold as 0.6, spatial threshold as 2, the time range threshold as 8 min, and record the DVP of different methods continuously for half an hour. We also set time threshold as 3, 5, 8, 10 min, and calculate the ART of different methods.

Because our method is also supported running in parallel, then we evaluate the expandability of our method by adjusting degrees of parallelism. We simulated trajectory stream data with higher frequency as 1000 Hz, set the parallelism degree as 1, 3, and 5, and respectively calculate the ART.

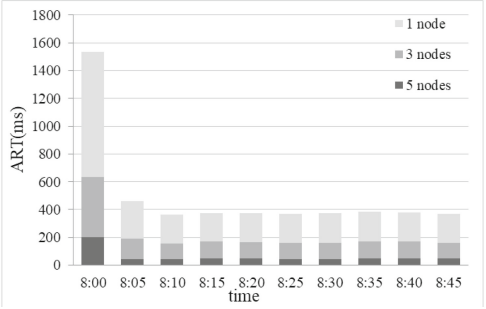
Figure 8 shows the results of the efficiency of our service collaboration-based method and PlatoonFinder method, and the efficiency of our method with different degree of parallelism. As shown in Fig. 8(a), the Data Volume Processed per second of PlatoonFinder method consistently remained high. Conversely, the DVP of our method initially peaked but then sharply declined after a certain period, stabilizing at a significantly lower level. This discrepancy arises from the centralized processing approach, which necessitates receiving ANPR data generated by all sites throughout the entire execution period. In contrast, our method only requires receiving all ANPR data at the beginning, resulting in improved efficiency over time. Once the target vehicle is discovered, it only needs to receive ANPR data from relevant sites, thereby avoiding the reception and processing of large amounts of irrelevant data. Likewise, Fig. 8(b) shows that our method's ART was also greatly lower than PlatoonFinder method. And Fig. 8(c) shows that with the degree of parallelism increasing, the ART in our method is reduced. Especially at the beginning of vehicle companion discovery, when we need to process all the ANPR data, the difference becomes even more pronounced. This highlights the effectiveness of service composition parallelism in enhancing performance, even when dealing with substantial amounts of data streams.



(a) Different methods' DVP



(b) Different methods' ART



(c) The ART of differernt digrees of parallelism

Fig. 8. Comparison of efficiency between different methods.

6 Conclusion

We propose a stream data service framework to discover vehicle companion efficiently and effectively. Specifically, we redefined the vehicle companion pattern based on relaxed temporal and spatial constraints, and deal with vehicle redundancy in area companions. The definition of vehicle companion helps more potential companion vehicles' discovery. And we encapsulate each trajectory data generated by monitoring site into corresponding stream data services, and enhance value density of trajectory data initially. These services are designed to collaborate with each other dynamically based on the vehicles' real-time status and the monitoring sites' adjacent relationships. Through this collaboration, higher-level services are constructed to ultimately identify the companion vehicles. Dynamic service collaboration ensures efficiency of our method because of the reducing of unnecessary communication and processing costs. And a stream partition strategy that divides services into partitions is also proposed to further ensure the performance of our method. The results of experiment based on real datasets demonstrate that our discovery method can identify more potential companion vehicles while requiring fewer processing and communication resources.

For future work, we intend to study more interesting and extensive correlations and patterns in trajectory streams, and study how to better utilize stream data service in intelligence transportation area.

Acknowledgment. This work was supported by Youth Foundation of Shandong Natural Science Foundation (No. ZR2021QF099) and Doctor Scientific Research Foundation of Shandong Jianzhu University (No. X21007Z).

References

1. Gao, Q., Zhang, F.L., Wang, R.J., Zhou, F.: Trajectory big data: a review of key technologies in data processing. J. Softw. **28**(4), 959–992 (2017)
2. Kong, X., Li, M., Zhao, G., et al.: COOC: visual exploration of co-occurrence mobility patterns in urban scenarios. IEEE Trans. Comput. Soc. Syst. **6**(3), 403–413 (2019)
3. Jia, J., Ying, Hu., Zhao, B., Ji, G., Liu, R.: Discovering collective converging groups of large scale moving objects in road networks. In: Jensen, C.S., et al. (eds.) DASFAA 2021. LNCS, vol. 12682, pp. 307–324. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-73197-7_21
4. van Mulken, M., Speckmann, B., Verbeek, K.: Density approximation for moving groups. In: Morin, P., Suri, S. (eds.) WADS 2023. LNCS, vol. 14079, pp. 675–688. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38906-1_45
5. Ning, Z., Xia, F., Ullah, N., Kong, X.J., Hu, X.P.: Vehicular social networks: enabling smart mobility. IEEE Commun. Mag. **55**(5), 16–55 (2017)
6. Zheng, Y., Xie, X., Ma, W.Y.: GeoLife: a collaborative social networking service among user, location and trajectory. Bull. Tech. Committee Data Eng. **33**(2), 32–39 (2010)
7. Han, Y.B., Wang, G.L., Yu, J., et al.: A service-based approach to traffic sensor data integration and analysis to support community-wide green commute in China. IEEE Trans. Intell. Transp. Syst. **17**(9), 2648–2657 (2016)
8. Vieira, M.R., Bakalov, P., Tsotras, V.J.: On-line discovery of flock patterns in spatio-temporal data. In: Proceedings of the 17th ACM International Symposium on Advances in Geographic Information Systems (ACM SIGSPATIAL), pp. 286–295. Association for Computing Machinery, New York (2009)
9. Zaleshina, M., Zaleshin, A.: Flock patterns when pigeons fly over terrain with different properties. In: ICPRAM, pp. 334–341 (2019)
10. Jeung, H., Shen, H.T., Zhou, X.F.: Convoy queries in spatio-temporal databases. In: Proceedings of the IEEE International Conference on Data Engineering (ICDE), Washington, pp. 1457–1459. IEEE Computer Society (2008)
11. Yan, S., Wu, B., Shang, L., Wang, Y., Lyu, J.: A convoy discovering algorithm for passengers in the cruise based on UWB positioning. In: 2021 6th International Conference on Transportation Information and Safety (ICTIS), pp.392–397 (2021)
12. Li, Z.H., Ding, B.L., Han, J.W., Kays, R.: Swarm: mining relaxed temporal moving object clusters. Proc. VLDB Endow. **3**(1), 723–734 (2010)
13. Wang, X., Zhang, Y., Wang, L., et al.: Task decision-making for UAV swarms based on robustness evaluation. In: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 242–248 (2019)
14. Li, Y.X., Bailey, J., Kulik, L.: Efficient mining of platoon patterns in trajectory data-bases. Data Knowl. Eng. **100**, 167–187 (2015)
15. Zhu, M.L., Liu, C., Wang, X.B., Han, Y.B.: Approach to discover companion pattern based on ANPR data stream. Ruan Jian Xue Bao/J. Softw. **28**(6), 1498–1515 (2017)
16. Zhuofeng, Z., Shuai, L., Yanbo, H.: Similar trajectory query method based on massive vehicle license plate recognition data. J. Tsinghua Univ. (Sci. Technol.) **57**(2), 220–224 (2017)
17. Xiao, Y., He, X., Yang, C., Liu, H., Liu, Y.: Dynamic graph computing: a method of finding companion vehicles from traffic streaming data. Inf. Sci. **591**, 128–141 (2022)

18. Showail, A., Tahir, R., Zaffar, M., et al.: An internet of secure and private things: a service-oriented architecture. Comput. Secu. **120**, 102776 (2021)
19. Mishra, S., Sarkar, A.: Service-oriented architecture for Internet of Things: a semantic approach. J. King Saud Univ. Comput. Inf. Sci. **34**(10), 8765–8776 (2021)
20. Huang, B., Zhang, B., Sheng, Q.Z., Lam, K.-Y.: A multi-task learning approach for predicting intentions using smart home IoT services. In: Troya, J., Medjahed, B., Piattini, M., Yao, L., Fernández, P., Ruiz-Cortés, A. (eds.) ICSOC 2022. LNCS, vol. 13740, pp. 413–421. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20984-0_29
21. Zhang, Z., Liu, C., Su, S., et al.: SDaaS: a method for encapsulating sensor stream data as services. China J. Comput. **40**(2), 445–463 (2017)
22. Han, Y., Liu, C., Su, S.: A decentralized and service-based approach to proactively correlating stream data. In: S2 International Conference on Internet of Things, pp. 93–100 (2016)
23. Zhang, Z., Hu, Q., Hou, G., Zhang, S.: A real-time discovery method for vehicle companion via service collaboration. Int. J. Web Inf. Syst. **19**(5/6), 263–279 (2023)
24. Ali, Z.H., Ali, H.A., Badawy, M.M.: A new proposed the Internet of Things (IoT) virtualization framework based on sensor-as-a-service concept. Wireless Pers. Commun. **97**(1), 1419–1443 (2017)
25. Silva, B.N., Khan, M., Han, K.: Integration of Big Data analytics embedded smart city architecture with RESTful web of things for efficient service provision and energy management. Future Gener. Comput. Syst. **107**, 975–987 (2018)
26. Belhadi, A., Djenouri, Y., Srivastava, G., Lin, J.C.: Fast and accurate framework for ontology matching in web of things. ACM Trans. Asian Low Resour. Lang. Inf. Process. **22**(5), 147:1–147:19 (2023)
27. Ahrabian, A., Kolozali, S., Enshaeifar, S., et al.: Stream data analysis as a web service: a case study using IoT sensor data. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, New Orleans, United States, pp. 6000–6004. IEEE (2017)
28. Aguilar, J., Sanchez, M., Cordero, J., et al.: Learning analytics tasks as services in smart classrooms. Univ. Access Inf. Soc. **17**(4), 693–709 (2019)
29. Malik, M., Abdallah, S., Alaraj, M.: Data mining and predictive analytics applications for the delivery of healthcare services: a systematic literature review. Ann. Oper. Res. **270**(1–2), 287–312 (2018)
30. Lu, Y., Misra, A., Wu, H.: Smartphone sensing meets transport data: a collaborative framework for transportation service analytics. IEEE Trans. Mob. Comput. **17**(4), 945–960 (2018)
31. Zatout, S., et al.: A model-driven approach for the verification of an adaptive service composition. Int. J. Web Eng. Technol. **15**(1), 18–26 (2021)
32. Zhang, Z.M., Yang, Z.G., Ali, S., Asshad, M.: A dynamic declarative composition scheme for stream data services. Mob. Inf. Syst. **1–8**, 2021 (2021)
33. Wang, Y., Wang, S., Yang, B., et al.: An effective adaptive adjustment method for service composition exception handling in cloud manufacturing. J. Intell. Manuf. **33**, 735–751 (2022)
34. Zhang, J., Liu, S., Yang, Q., Zhou, Y.: DMFUCP: a distributed mining framework for universal companion patterns on large-scale trajectory data. J. Comput. Res. Dev. **59**(3), 647–660 (2021)
35. Mertens, S.: The easiest hard problem: number partitioning. Comput. Complex. Stat. Phys. **125**, 125–139 (2003)



KS-Autoformer: An Autoformer-Based SOC Prediction Framework for Electric Vehicles

Yaoyidi Wang[1], Niansheng Chen[1], Lei Rao[1(✉)], Dingyu Yang[2], Guangyu Fan[1], Songlin Cheng[1], and Xiaoyong Song[1]

[1] Shanghai Dianji University, Shanghai 201306, China
raol@sdju.edu.cn
[2] Alibaba Group, Hangzhou 310056, China

Abstract. Accurate state of charge prediction is essential for battery management systems, which is crucial for improving battery utilization efficiency and ensuring safety performance. Current SOC prediction models only consider battery-related features but ignores vehicle information. Additionally, there are challenges preventing gradient disappearance and explosion during model training due to excessive data and noise. This paper introduces a new framework that integrates laboratory battery data with vehicle features to improve the accuracy of SOC. First, we apply Matlab/Simulink to simulate an electric vehicle and process the generated vehicle data with spearman correlation analysis to identify the most relevant features, such as the electric motor, differential, and aerodynamic drag. Then we develop a data fusion model to synchronize the heterogeneous datasets with different frequencies to capture the sudden change in electric vehicles. Furthermore, we propose a KS-Autoformer prediction model to address the overfitting problem caused by data redundancy and noise, which improves accuracy by smoothing sensor noise and enhancing time delay aggregation in an auto-correlation mechanism. Finally, we utilize different driving cycles for training and testing in order to effectively evaluate the extrapolation and adaptability of our model. Experimental results show that our model achieves a significant improvement in predicting SOC in terms of accuracy and robustness.

Keywords: State of Charge · Battery Management System · Autoformer

1 Introduction

Global emissions and pollution from transportation are increasing, and electric vehicles (EVs) are seen as the best solution to replace conventional internal combustion engine vehicles [1,2]. EVs use lithium-ion batteries (LIBs) for their

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 297–312, 2024.
https://doi.org/10.1007/978-3-031-63989-0_15

high voltage and power density, and are equipped with a battery management system (BMS) to regulate battery usage and scheduling, as well as monitor battery status and perform maintenance [3,4].

The State of charge (SOC) is defined as the ratio of the remaining capacity to the current maximum available capacity. Accurate prediction of the state of charge is an important step in BMS [5]. By accurate prediction of the state of charge, the BMS can help to prevent overcharging or undercharging of the battery [6], which would lead to decreased performance and a shorter lifespan. Since the internal battery chemistry is a complex process, the SOC prediction is susceptible to the battery discharge rate, ambient temperature, and battery degradation. Meanwhile, there exists a significant non-linear correlation between the external variables of an electric vehicle and its battery, which can make accurate SOC prediction challenging. For example, the aerodynamic drag coefficient of the electric vehicle varies with its speed, and the mechanical power loss of the motor can result in faster battery discharge, further contributing to inaccurate SOC prediction.

Researchers have proposed various methods for SOC prediction, including traditional methods, model-based methods, and data-driven methods.

Traditional methods include the Ampere-hour integration method [7,8] and the open-circuit voltage method [9,10]. These methods are still used in commercial electric vehicles because of their simplicity and low computational resource requirements. The Ampere-hour integration method predicts SOC by integrating the discharge current over time, but it has a strong dependence on the accuracy of the initial SOC value. Moreover, measurement errors can accumulate over time due to low-precision current sensors, resulting in increasingly inaccurate SOC prediction. The open-circuit voltage (OCV) method predicts SOC by measuring the battery's OCV and checking the OCV-SOC curve, but the battery's OCV can only be obtained by long-term offline maintenance, making this method difficult to predict SOC in real-time. Therefore, traditional methods have limitations on low precision and difficulty in real-time application.

A model-based method using an equivalent circuit model (ECM) [11] has been proposed to predict the state of a battery. This method applies the ampere-hour integration approach based on the last moment SOC to predict the current SOC. The simulated voltage of the battery, combined with the equivalent circuit model, is then used to predict the SOC. An adaptive filter is introduced to calibrate the prediction of SOC by continuously reducing the error between the simulated value and the actual measured voltage, which creates a closed-loop SOC prediction. Common ECM models include the PNGV model [12], Thevenin model [13], and second-order RC model [14,15]. Common adaptive filtering algorithms include particle filtering [16,17], extended Kalman filtering (EKF) [18,19], unscented Kalman filtering (UKF) [20,21], and cubature Kalman filtering (CKF) [22,23]. While this method can fix initial errors and avoid error accumulation over time, it has poor adaptability to resistance and capacitance parameters. Additionally, it ignores the internal chemical characteristics of the battery and nonlinear errors in the equivalent model.

The data-driven method is a machine learning approach that uses historical battery usage data to predict the state of charge (SOC) of the battery [24, 25]. This method does not require any knowledge of the internal workings of the battery and can be easily implemented by inputting training data into a data-driven model. Statistical learning methods such as support vector machines (SVM) [26] and Gaussian process regression (GPR) [27,28] are faster and more computationally efficient, making them more suitable for sparse features.

However, existing SOC prediction frameworks have several limitations that need to be addressed, including:

(I) Current battery input data cannot reflect the sudden changes in operating conditions of electric vehicles. Tian et al. [29] proposed a DNN approach for SOC prediction in EVs using data from a 10-minute charging session, which consists of convolutional layers, GRU layers and dense layer. Huang et al. [30] developed a hybrid neural networks CNN-GRU model using single time-point measurements of voltage, current, and temperature. However, battery SOC fluctuate significantly during acceleration and deceleration of vehicle, making it difficult for models trained on single-time-point or longer-duration data to capture sufficient temporal information.

(II) External vehicle conditions are not considered in predicting SOC. Previous studies on SOC prediction mainly used laboratory battery data as input variables. Furthermore, in actual EV operations, it is hard to obtain external conditions that affect battery discharge due to cost and feasibility constraints. Huang et al. [31] used real-world vehicle data from NDANEV but equalized battery degradation effects with vehicle driving mileage. Li et al. [32] integrated weather and vehicle data and used a novel dual-dropout-based neural network to predict SOC. Although these studies considered the impact of vehicle data on SOC prediction, they were both suffered from significant data loss and sampling limitations.

(III) Current SOC prediction models fail to address overfitting from large data and noise. The noise in battery data collection and the overfitting caused by large data will result in unstable SOC prediction and reduced accuracy, which affects EV safety and driving range.

Therefore, the main contributions of this paper are as follows:

(I) A novel SOC prediction framework: a simulation model of an electric vehicle is built using Matlab/Simulink, which incorporates external vehicle data and laboratory battery data to establish the relationship between external vehicle conditions and battery consumption. To the best of our knowledge, we are the first to formulate this framework that can be customized for specific vehicle operating conditions by inputting corresponding driving cycles and electric vehicle parameters.

(II) A data fusion method for heterogeneous datasets with different frequencies: We resample the data to capture the sudden change of electric vehicles with features in our simulation and enhance the robustness of the trained model for complex battery conditions.

(III) We propose a KS-Autoformer model, which utilizes decomposition architectures to handle large amounts of data. The Auto-Correlation attention mechanism is used to better predict the SOC of electric vehicles under driving cycles. Moreover, we apply Kalman smoothing using a recursive algorithm to filter and eliminate noise from laboratory-collected battery discharge data.

(IV) We applied our framework to different driving cycles for training and testing. Through comparison and ablation experiments, we show that our proposed framework and prediction model can achieve more accurate SOC predictions.

2 Prediction Framework for Electric Vehicle

In this section, we describe the details of our prediction framework of SOC. It includes three modules: laboratory battery data and simulated external vehicle data generation, data fusion utilizing high-frequency sampling data, and a deep-learning based prediction model called *KS-Autoformer*. The proposed framework is illustrated in Fig. 1. We first outline the process of constructing an electric vehicle simulation and analyze the generated features to identify the most related information. Next, we will introduce the fusion of laboratory battery and simulated external vehicle data, which finally will be performed in our proposed prediction model *KS-Autoformer*.

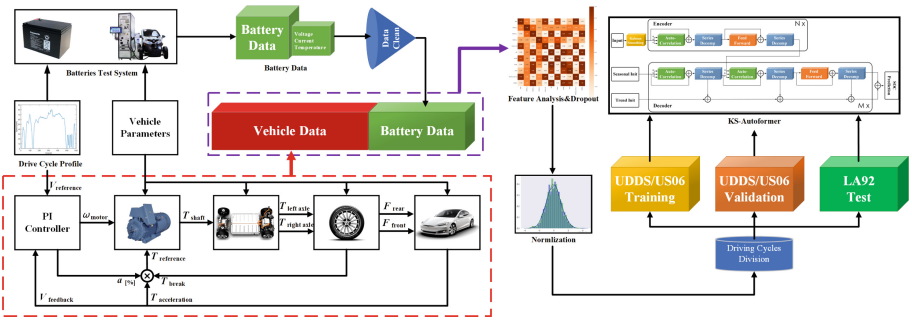


Fig. 1. The overview of SOC prediction architecture

2.1 Electric Vehicle Simulation

Since it is difficult to measure or obtain vehicle data (such as losses in the electric motor, power system, and air resistance) in real systems, we conduct a simulation of an electric vehicle model using Matlab/Simulink [33]. Simulink's Powertrain Blockset [34,35] is a powerful computational tool that has been widely used in simulations and analysis of vehicle dynamics. The Powertrain Blockset consists of several blocks that are designed to calculate various aspects of the motor system.

In order to generate the electric motor's mechanical power data W_{motor}, we utilize the Mapped Motor block to control the output shaft torque T_{shaft}, based on the reference torque $T_{\text{reference}}$. This reference torque is determined by the acceleration torque $T_{\text{acceleration}}$ and braking torque T_{brake}.

Next, the Limited Slip Differential and Longitudinal Wheel blocks are used to calculate the corresponding torques $T_{\text{left axle}}$ and $T_{\text{right axle}}$ and the net longitudinal forces F_{rear} and F_{front} acting on the rear and front wheels, respectively, which allow us to determine the mechanical power transferred through the differential W_{differ}.

By utilizing the Vehicle Body 1DOF Longitudinal block, which considers the feedback speed V_{feedback} and acceleration torque $T_{\text{acceleration}}$, we are able to obtain the mechanical power transferred from the axle to the wheel W_{wheel}, as well as the power loss due to rolling resistance W_{roll}, the mechanical power required for braking W_{break}, and the power loss due to wind resistance W_{wind}.

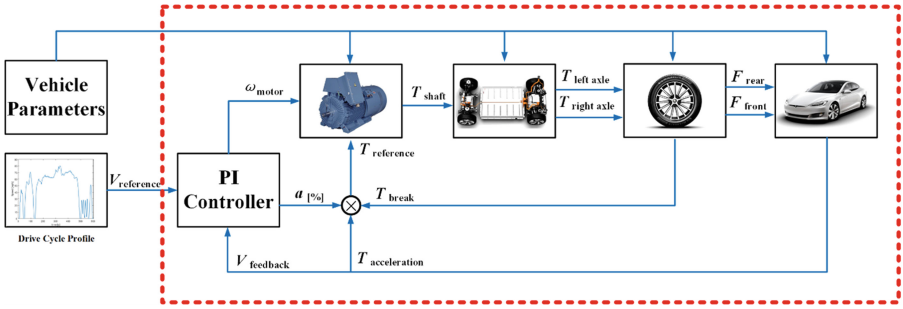


Fig. 2. Schematic of the EV model using Matlab/Simulink.

To summarize, these blocks are used to simulate the powertrain system and vehicle dynamics of an electric vehicle. A schematic of the EV model is designed and presented in Fig. 2. The mechanical power of the motor W_{motor}, power transferred through the differential W_{differ}, power transferred from axle to wheel W_{wheel}, power loss due to rolling resistance W_{roll}, power required for braking W_{break}, and power loss due to wind resistance W_{wind} will be output as a part of the vehicle data. These components are widely recognized and utilized as essential features within real-world automotive datasets.

2.2 Feature Analysis

In the previous section, we obtain various features of the electric vehicle model including W_{motor}, W_{differ}, W_{wheel}, W_{roll}, W_{break}, and W_{wind}. However, it is important to note that not all vehicle features are equally relevant to the battery SOC, and the inclusion of irrelevant features in the dataset may result in overfitting. Therefore, a feature analysis is necessary to identify the most relevant characteristics for accurate SOC prediction.

To analyze the features, we utilize the Spearman correlation coefficient [36], which is ideal for SOC data that displays a monotonic decay. The coefficient measures the strength and direction of the monotonic relationship between two variables, as defined by the formula:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\left(n^2 - 1\right)} \tag{1}$$

where d is the difference in ranks between the two variables, and n is the number of data points. This method does not assume a linear relationship or a specific distribution form, making it more suitable for evaluating the correlation between features and battery SOC, especially when dealing with monotonic data.

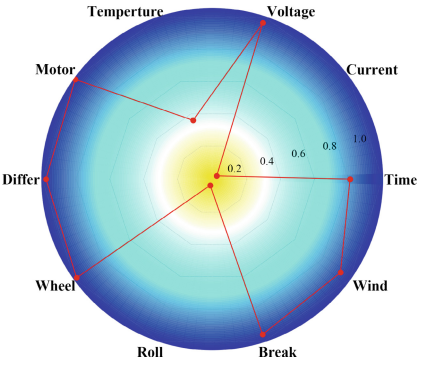


Fig. 3. Spearman correlation coefficient between different features and SOC.

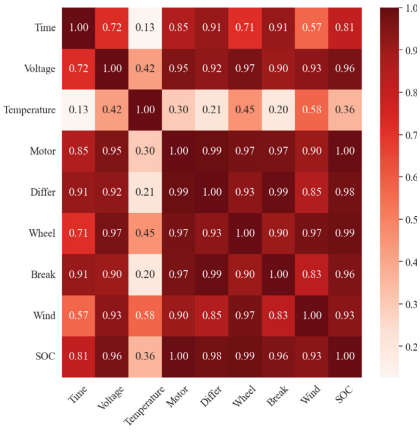


Fig. 4. Spearman correlation coefficient of selected features.

In Fig. 3, we observe that the Current in battery data and the power loss W_{roll} due to rolling resistance in vehicle data both have a low correlation coefficient of 0.02 and 0.03. Hence, we drop these two irrelevant features from our dataset. Then, we conduct a secondary correlation analysis on the remaining features, and as shown in Fig. 4, select eight relevant features for SOC prediction. This approach helps to eliminate irrelevant features and reduces the data size, thereby avoiding the overfitting of the model.

2.3 Data Fusion

To integrate the battery data and simulated external vehicle condition data, we employ a data fusion technique to match the spatial and temporal dimensions of the datasets.

For spatial matching, we use the vehicle parameters and driving cycle as the connection link. Specifically, we input the driving cycles and vehicle parameters used in the battery discharge tests (UDDS, US06 and LA92) into the simulation model to ensure the consistency of the data in the test environment.

For temporal matching, we first interpolate the missing battery data as it is collected in the laboratory using the sensor. We utilize the Lagrange interpolation method [37]. Assuming that the battery dataset consists of n-time data points. The Lagrange interpolation formula is:

$$l_i\left(x_{missing}\right) = \prod_{j=1, j\neq i}^{n} \frac{x_{missing} - x_j}{x_i - x_j} \tag{2}$$

where $l_i\left(x_{missing}\right)$ is the Lagrange interpolation basis function:

$$y_{missing} = \sum_{i=1}^{n} y_i \cdot l_i\left(x_{missing}\right) \tag{3}$$

By interpolating the missing data, we ensure that the battery data is a complete sample with a fixed sampling frequency of 10 Hz. We then set the simulation model's vehicle data sampling frequency to match the same rate of 10 Hz. This allows us to fuse the battery data and simulated vehicle data for SOC prediction.

2.4 Prediction Model

The KS-Autoformer module, as depicted in Fig. 5, is composed of three essential parts: the decomposition architecture, the auto-correlation mechanism, and the Kalman smoothing technique.

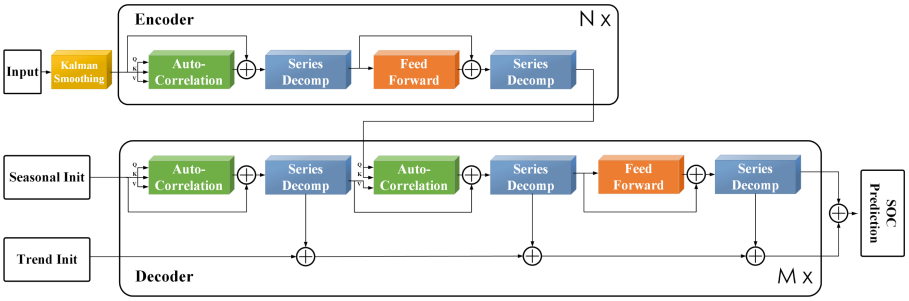


Fig. 5. Structure of KS-Autoformer.

Decomposition Block. Decomposition can help to separate the seasonal and trend components of our dataset. By doing so, we can better understand the stable and fluctuating patterns of the electric vehicle's behavior. Furthermore, it

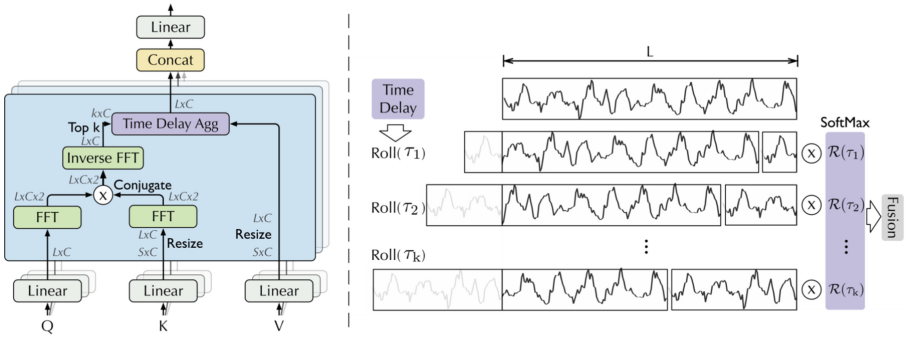


Fig. 6. Auto-Correlation (left) and Time delay aggregation (right).

simplifies the training process and reduces the risk of overfitting. For a Length L time series $X \in \mathbb{R}^{L \times d}$, the decomposition can be expressed as:

$$\begin{aligned} X_t &= AvgPool(Padding(X)) \\ X_s &= X - X_t \end{aligned} \tag{4}$$

where $X_s, X_t \in \mathbb{R}^{L \times d}$ represent the seasonal and trends part severally. AvgPool(X) combined with a padding method is exploited as the moving average operation. As a result, the whole process of decomposition can be expressed as:

$$X_s, X_t = SeriesDecomp(X) \tag{5}$$

Auto-correlation Mechanism. The Auto-Correlation mechanism [38] is taking the place of the canonical self-attention mechanism which is shown in Fig. 6. The historical sub-sequences of battery data often contain important information about the current battery state. To model the similarities between the subsequences, suppose a real-time discrete series $\{X_t\}$, the auto-correlation $R_{XX}(\tau)$ is proposed, which can be expressed as:

$$R_{XX}(\tau) = \lim_{L \to +\infty} \frac{1}{L} \sum_{t=1}^{L} X_t X_{t-\tau} \tag{6}$$

where $R_{XX}(\tau)$ measures the time lag similarities between $\{X_t\}$ and the lag τ series $\{X_{t-\tau}\}$. After the calculation process, the top k possible temporal series $\tau_1, \tau_2, \ldots, \tau_k$ are selected and can be expressed as:

$$\tau_1, \cdots, \tau_k = argTopk_{\tau \in 1, \cdots, L_{seq}} \left(R_{Q,K}(\tau) \right) \tag{7}$$

To achieve sequence-level joins, similar subsequence information needs to be aggregated. As shown in Fig. 6, according to the predicted cycle length, the Roll(·) operation is first used for information alignment, and then the information aggregation is carried out, and the form of query, key, and value are still used here.

$$\hat{R}_{Q,K}(\tau_1), \cdots, \hat{R}_{Q,K}(\tau_k) = SoftMax\left(\hat{R}_{Q,K}(\tau_1), \cdots, \hat{R}_{Q,K}(\tau_k)\right) \quad (8)$$

$$Auto - Correlation(Q, K, V) = \sum_{i=1}^{k} Roll(V, \tau_k)\, \hat{R}_{Q,K}(\tau_k) \quad (9)$$

Kalman Smoothing. To enhance the time delay aggregation technique described earlier, we employed Kalman smoothing, a mathematical method used to reduce noise and preserve the original patterns in laboratory battery data. The application of Kalman smoothing involves utilizing the following equation:

$$x_{k|k-1}\& = F_k x_{k-1|k-1} + B_k u_k \quad (10)$$

$$P_{k|k-1}\& = F_k P_{k-1|k-1} F_k^T + Q_k \quad (11)$$

$$K_k\& = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \quad (12)$$

$$x_{k|k}\& = x_{k|k-1} + K_k(y_k - H_k x_{k|k-1}) \quad (13)$$

$$P_{k|k}\& = (I - K_k H_k) P_{k|k-1} \quad (14)$$

where $x_{k|k}$ is the a predict of the state, F_k is the state transition matrix, B_k is the input matrix, u_k is the input vector, $P_{k|k-1}$ is the a priori predict error covariance matrix, Q_k is the process noise covariance matrix, K_k is the Kalman gain, y_k is the measurement vector. H_k is the observation matrix, R_k is the measurement noise covariance matrix.

In Fig. 7, Kalman smoothing is shown to have improved the quality of laboratory battery data by reducing noise while preserving temporal patterns. This enhanced data can be used to improve time delay aggregation, a technique introduced above to summarize temporal data. Kalman smoothing produces a more accurate representation of the process, which leads to more precise aggregated values. It also helps to reduce the impact of outliers and anomalies in the data, improving the robustness of time delay aggregation results.

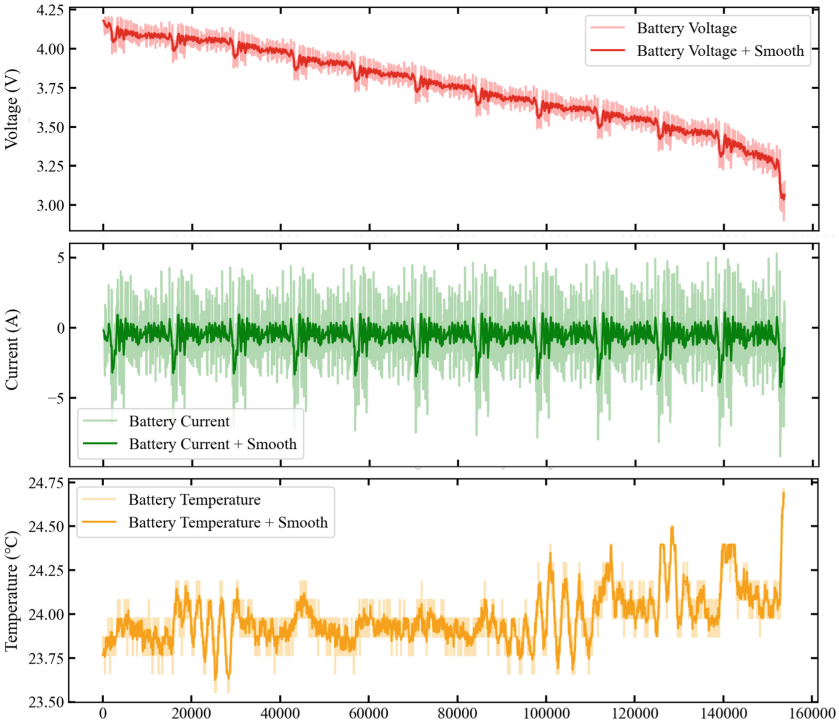


Fig. 7. Kalman smoothing on [a] Voltage; [b] Current; [c] Temperature.

3 Experimental Evaluation

Dataset. We employed the LG Open Battery Dataset [39] and applied it within the framework proposed in our research. Figure 8 displays the operational data of a 3Ah LG 18650 battery under the UDDS driving cycle, accompanied by the vehicle's external data obtained using our framework. In order to more effectively evaluate the extrapolation and adaptability of the model, we select specific driving cycles to form the training and validation sets and use the data from another DC for testing. In this research paper, we employ the UDDS and US06 driving cycles to form the training and validation sets. The UDDS and US06 driving cycles are chosen due to their ability to represent typical urban driving conditions, encompassing factors such as stop-and-go traffic and moderate acceleration patterns. Additionally, the US06 driving cycle specifically captures aggressive and high-speed/acceleration driving behavior, incorporating rapid speed fluctuations. To assess the model's performance on unseen data, we employ the LA92 driving cycle as the test dataset.

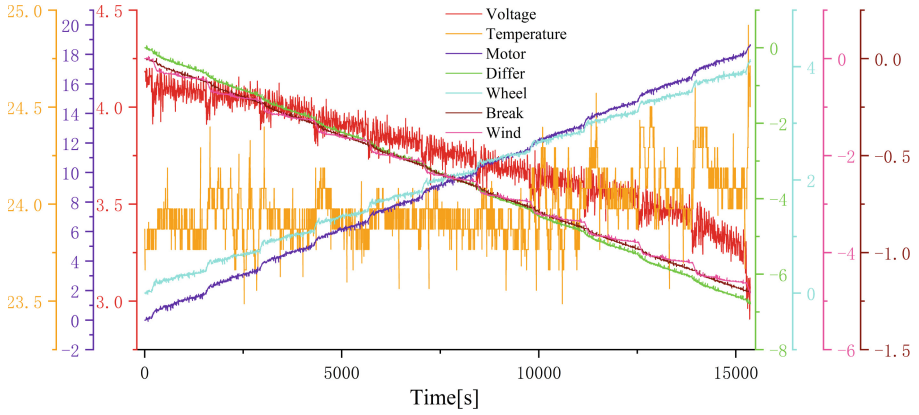


Fig. 8. Illustration of battery and vehicle dataset.

Evaluation Metrics. SOC prediction can be evaluated using various metrics, including MSE, RMSE, and MAE. These metrics provide quantitative measures of how well an prediction algorithm is performing compared to actual SOC values.

MSE (Mean Squared Error) formula is:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(SOC_{\mathrm{est}} - SOC_{\mathrm{act}})^2 \tag{15}$$

where SOC_{est} is the predicted value, SOC_{act} is the true value, and N is the number of samples. MSE provides a simple and straightforward expression for prediction error, but squaring the error value amplifies the impact of outliers.

RMSE (Root Mean Squared Error) formula is:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(SOCest - SOCact)^2} \tag{16}$$

RMSE is essentially the square root of MSE, which helps to ensure that the evaluation metric is consistent with the original value scale.

MAE (Mean Absolute Error) formula is:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|SOCest - SOCact| \tag{17}$$

MAE maintains consistency between the scale of the error metric and the original values, and does not suffer from the issue of error amplification due to squaring.

Implementation. To ensure fair and meaningful comparisons between the KS-Autoformer model and other models, we employed consistent hyperparameter

settings as foo: an initial learning rate of 0.0001, p = 0.05 dropout, and a batch size of 32. The size of each input time series segment is 96, and the prediction horizon is set to 24. The training process is terminated after six epochs, and we repeat each experiment five times to ensure the reliability of the results. Also, we compute the mean and standard deviation of the different datasets to normalize all the data points. In the end, we conduct experiments on a workstation equipped with a GeForce RTX 3070 GPU and an AMD Ryzen 7 5800 CPU.

Comparative Experiment. In our experiments, we conduct a comparative analysis between KS-Autoformer and several baseline models, including Autoformer, Informer, and Transformer. The results are displayed in Fig. 9. This experiment reveals that KS-Autoformer outperforms the baseline Autoformer model in predicting the state of charge of electric vehicle batteries since our model addresses sensor data noise and improves time delay aggregation in the Auto-correlation mechanism. Moreover, our proposed model performs better than other Transformer models, such as Informer and the original Transformer model.

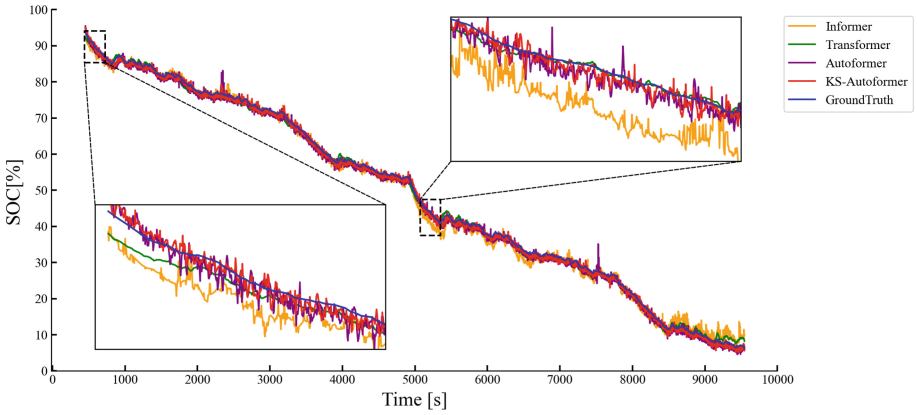


Fig. 9. Performance of different SOC prediction models.

Ablation Experiment. To demonstrate the validity of the proposed SOC prediction framework, we conducted another ablation experiment. Specifically, we compared our proposed model with three other baseline models under both the original SOC prediction framework and the proposed SOC prediction framework, as shown in Table 1 and Fig. 10. The results indicate that all four algorithms outperformed the original SOC prediction framework.

Table 1. Comparison of models in different frameworks.

Model	Framework	MSE [%]	RMSE [%]	MAE [%]
KS-Autoformer	Origin	0.0228	1.5099	0.7974
	Proposed	**0.0083**	**0.9149**	**0.7295**
Autoformer [38]	Origin	0.0257	1.6061	0.6290
	Proposed	**0.0157**	**1.2540**	**0.7876**
Informer [40]	Origin	0.0698	2.6426	2.1114
	Proposed	**0.0681**	**2.6096**	**1.9711**
Transformer [41]	Origin	0.0475	2.1802	1.1241
	Proposed	**0.0439**	**1.9227**	**1.0886**

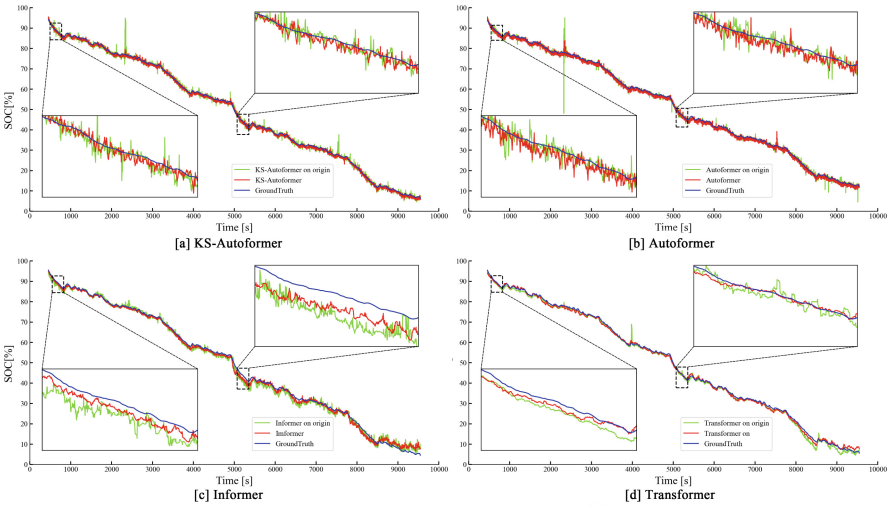


Fig. 10. Performance of different framework on [a] KS-Autoformer; [b] Autoformer; [c] Informer; [d] Transformer.

4 Conclusion

In this work, we propose a novel framework for predicting the state of charge of electric vehicle batteries, incorporating external vehicle data and laboratory battery data. After features analysis and data fusion, our high-frequency sampling dataset can capture sufficient temporal information under complex operating conditions. Furthermore, we propose a KS-Autoformer prediction model to address issues related to feature explosion and reduce sensor measurement errors. In order to more effectively evaluate the extrapolation and adaptability of our model, UDDS and US06 driving cycles are used for training and LA92 driving cycle for testing. Results show that our proposed framework achieves superior performance in predicting SOC, leading to more efficient battery utilization and improved safety performance.

References

1. Yang, F., Wang, D., Zhao, Y., Tsui, K.L., Bae, S.J.: A study of the relationship between coulombic efficiency and capacity degradation of commercial lithium-ion batteries. Energy **145**, 486–495 (2018). https://doi.org/10.1016/j.energy.2017.12.144. https://www.sciencedirect.com/science/article/pii/S0360544217321874
2. Xu, Y., et al.: Research on the impact of re-electrication in the transportation sector on carbon emission and pollutant emission in Yunnan province. In: 2021 Power System and Green Energy Conference (PSGEC), pp. 407–412. IEEE (2021)
3. Hannan, M.A., Lipu, M.H., Hussain, A., Mohamed, A.: A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: challenges and recommendations. Renew. Sustain. Energy Rev. **78**, 834–854 (2017)
4. Rahimi-Eichi, H., Ojha, U., Baronti, F., Chow, M.Y.: Battery management system: An overview of its application in the smart grid and electric vehicles. IEEE Ind. Electron. Mag. **7**(2), 4–16 (2013)
5. Zhang, D., Zhong, C., Xu, P., Tian, Y.: Deep learning in the state of charge estimation for li-ion batteries of electric vehicles: a review. Machines **10**(10), 912 (2022). https://doi.org/10.3390/machines10100912. https://www.mdpi.com/2075-1702/10/10/912
6. Zhou, W., Zheng, Y., Pan, Z., Lu, Q.: Review on the battery model and SOC estimation method. Processes **9**(9), 1685 (2021)
7. Zhang, X., Hou, J., Wang, Z., Jiang, Y.: Study of SOC estimation by the ampere-hour integral method with capacity correction based on LSTM. Batteries **8**(10), 170 (2022)
8. Deng, Y., Hu, Y., Cao, Y.: An improved algorithm of SOC testing based on open-circuit voltage-ampere hour method. In: Li, K., Xue, Y., Cui, S., Niu, Q. (eds.) LSMS/ICSEE 2014. CCIS, vol. 463, pp. 258–267. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45286-8_27
9. Tian, J., Xiong, R., Shen, W., Sun, F.: Electrode ageing estimation and open circuit voltage reconstruction for lithium ion batteries. Energy Storage Mater. **37**, 283–295 (2021)
10. Pillai, P., Sundaresan, S., Kumar, P., Pattipati, K.R., Balasingam, B.: Open-circuit voltage models for battery management systems: a review. Energies **15**(18), 6803 (2022)
11. Xile, D., Caiping, Z., Jiuchun, J.: Evaluation of SOC estimation method based on EKF/AEKF under noise interference. Energy Procedia **152**, 520–525 (2018)
12. Liu, X., Li, W., Zhou, A.: PNGV equivalent circuit model and SOC estimation algorithm for lithium battery pack adopted in AGV vehicle. IEEE Access **6**, 23639–23647 (2018)
13. Susanna, S., Dewangga, B.R., Wahyungoro, O., Cahyadi, A.I.: Comparison of simple battery model and thevenin battery model for SOC estimation based on OCV method. In: 2019 International Conference on Information and Communications Technology (ICOIACT), pp. 738–743. IEEE (2019)
14. Ji, Y., Qiu, S., Li, G.: Simulation of second-order RC equivalent circuit model of lithium battery based on variable resistance and capacitance. J. Cent. South Univ. **27**(9), 2606–2613 (2020)
15. Xu, Y., Hu, M., Fu, C., Cao, K., Su, Z., Yang, Z.: State of charge estimation for lithium-ion batteries based on temperature-dependent second-order RC model. Electronics **8**(9), 1012 (2019)

16. Burgos-Mellado, C., Orchard, M.E., Kazerani, M., Cárdenas, R., Sáez, D.: Particle-filtering-based estimation of maximum available power state in lithium-ion batteries. Appl. Energy **161**, 349–363 (2016)
17. Xiong, R., Zhang, Y., He, H., Zhou, X., Pecht, M.G.: A double-scale, particle-filtering, energy state prediction algorithm for lithium-ion batteries. IEEE Trans. Ind. Electron. **65**(2), 1526–1538 (2017)
18. Plett, G.L.: Dual and joint EKF for simultaneous SOC and SOH estimation. In: Proceedings of the 21st Electric Vehicle Symposium (EVS21), Monaco, pp. 1–12 (2005)
19. Li, M., Zhang, Y., Hu, Z., Zhang, Y., Zhang, J.: A battery SOC estimation method based on AFFRLS-EKF. Sensors **21**(17), 5698 (2021)
20. El Din, M.S., Hussein, A.A., Abdel-Hafez, M.F.: Improved battery SOC estimation accuracy using a modified UKF with an adaptive cell model under real EV operating conditions. IEEE Trans. Transp. Electrification **4**(2), 408–417 (2018)
21. He, H., Qin, H., Sun, X., Shui, Y.: Comparison study on the battery SoC estimation with EKF and UKF algorithms. Energies **6**(10), 5088–5100 (2013)
22. Tao, Z., Li, Z., Yangcheng, H.O.U., Wei, C.: SOC estimation of aging lithium battery based on adaptive CKF. Energy Storage Sci. Technol. **9**(4), 1193 (2020)
23. Luo, J., Peng, J., He, H.: Lithium-ion battery SOC estimation study based on Cubature Kalman filter. Energy Procedia **158**, 3421–3426 (2019)
24. Cai, L., Meng, J., Stroe, D.I., Peng, J., Luo, G., Teodorescu, R.: Multiobjective optimization of data-driven model for lithium-ion battery SOH estimation with short-term feature. IEEE Trans. Power Electron. **35**(11), 11855–11864 (2020)
25. How, D.N., Hannan, M.A., Lipu, M.H., Ker, P.J.: State of charge estimation for lithium-ion batteries using model-based and data-driven methods: a review. IEEE Access **7**, 136116–136136 (2019)
26. Antón, J., Nieto, P.G., de Cos Juez, F.J., Lasheras, F.S., Vega, M.G., Gutiérrez, M.R.: Battery state-of-charge estimator using the SVM technique. Appl. Math. Model. **37**(9), 6244–6253 (2013)
27. Ozcan, G., Pajovic, M., Sahinoglu, Z., Wang, Y., Orlik, P.V., Wada, T.: Online state of charge estimation for lithium-ion batteries using gaussian process regression. In: IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, pp. 998–1003. IEEE (2016)
28. Sahinoglu, G.O., Pajovic, M., Sahinoglu, Z., Wang, Y., Orlik, P.V., Wada, T.: Battery state-of-charge estimation based on regular/recurrent Gaussian process regression. IEEE Trans. Ind. Electron. **65**(5), 4311–4321 (2017)
29. Tian, J., Xiong, R., Shen, W., Lu, J.: State-of-charge estimation of LiFePO4 batteries in electric vehicles: a deep-learning enabled approach. Appl. Energy **291**, 116812 (2021)
30. Huang, Z., Yang, F., Xu, F., Song, X., Tsui, K.L.: Convolutional gated recurrent unit-recurrent neural network for state-of-charge estimation of lithium-ion batteries. IEEE Access **7**, 93139–93149 (2019)
31. Huang, S., He, Z., Li, X.: A method of SOC estimation for electric vehicle based on limited information. In: 2020 IEEE International Conference on Networking, Sensing and Control (ICNSC), pp. 1–4. IEEE (2020). https://doi.org/10.1109/ICNSC48988.2020.9238124. https://ieeexplore.ieee.org/document/9238124/
32. Li, R., Wang, H., Dai, H., Hong, J., Tong, G., Chen, X.: Accurate state of charge prediction for real-world battery systems using a novel dual-dropout-based neural network. Energy **250**, 123853 (2022). https://doi.org/10.1016/j.energy.2022.123853. https://linkinghub.elsevier.com/retrieve/pii/S0360544222007563

33. Kasera, J., Chaplot, A., Maherchandani, J.K.: Modeling and simulation of wind-PV hybrid power system using Matlab/Simulink. In: 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science, pp. 1–4. IEEE (2012)
34. Tran, M.K., Akinsanya, M., Panchal, S., Fraser, R., Fowler, M.: Design of a hybrid electric vehicle powertrain for performance optimization considering various powertrain components and configurations. Vehicles **3**(1), 20–32 (2020)
35. Ragone, M., Yurkiv, V., Ramasubramanian, A., Kashir, B., Mashayek, F.: Data driven estimation of electric vehicle battery state-of-charge informed by automotive simulations and multi-physics modeling. J. Power Sources **483**, 229108 (2021)
36. Myers, L., Sirois, M.J.: Spearman correlation coefficients, differences between **12** (2014)
37. Rashed, M.T.: Lagrange interpolation to compute the numerical solutions of differential, integral and integro-differential equations. Appl. Math. Comput. **151**(3), 869–878 (2004)
38. Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: decomposition transformers with auto-correlation for long-term series forecasting (2021). http://arxiv.org/abs/2106.13008
39. Kollmeyer, P., Vidal, C., Naguib, M., Skells, M.: LG 18650hg2 li-ion battery data and example deep neural network xEV SOC estimator script **3** (2020). https://doi.org/10.17632/cp3473x7xv.3. https://data.mendeley.com/datasets/cp3473x7xv/3
40. Zhou, H., et al.: Informer: Beyond efficient transformer for long sequence time-series forecasting (2021). http://arxiv.org/abs/2012.07436
41. Vaswani, A., et al.: Attention is all you need, vol. 30 (2017)



Deep Reinforcement Learning-Based Multi-node Collaborative Task Offloading Optimization in 6G Space-Air-Ground Integrated Networks

Wang Li, Xin Chen, Libo Jiao(✉), Wangzhong Ning, and Wenwu Zhu

Beijing Information Science and Technology University, Beijing, China
{wangli,chenxin,jiaolibo,ningwangzhong,zhuwenwu}@bistu.edu.cn

Abstract. With the explosion of communication data volume, 6G communication has gradually entered the vision of academia and industry. In addition, in order to support the execution of computationally intensive applications, the study of 6G space-air-ground integration network (SAGIN) is becoming more and more extensive, in which satellites, drones, and base stations can provide arithmetic support for mobile users (MUs) through multi-access edge computing (MEC). However, the variety of offloading modes, the mobility of nodes and the stochastic state of wireless networks make that selecting the optimal base station and allocating the appropriate computational resources become more challenging. In this paper, we first describe the offloading process and establish the SAGIN model. Then the Deep Reinforcement Learning-based Task Offloading Optimization Algorithm (DTOOA) is proposed to jointly optimize the problem of minimizing latency and energy consumption. The numerical results show that the DTOOA scheme significantly outperforms benchmark schemes and markedly improves the quality of experience (QoE) of MUs.

Keywords: 6G · Space-Air-Ground Networks · Multi-access Edge Computing · Task Offloading · Deep reinforcement learning

1 Introduction

According to the June 2023 Mobility Report from Ericsson, the number of global 5G subscribers will reach 1.5 billion and average global monthly usage per smartphone will exceed 20 GB by the end of 2023. In the era of unprecedented information explosion, huge data traffic puts higher demands on communication technology. 5G, which has already entered the commercial deployment phase, still has various limitations. In order to meet services such as global coverage, ultra-high-rate transmission, ultra-low latency and the Internet of Everything, industry,

Supported by the National Natural Science Foundation of China under Grant 62202059.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 313–332, 2024.
https://doi.org/10.1007/978-3-031-63989-0_16

academia and standards organizations have begun exploring and researching 6G [10]. 6G space-air-ground integration network (SAGIN) is considered a promising network architecture to address the limitations of 5G.

Mobile users (MUs) usually have limited computing resources and cannot handle computationally intensive tasks locally. The multi-access edge computing (MEC) paradigm was born to better meet new applications such as autonomous driving, ultra-high-definition video and telemedicine [6]. Compared to traditional cloud computing [4], MEC brings computing resources closer to the MUs and reduces processing latency and energy consumption of the MUs by offloading the tasks to the edge nodes. However, the terrestrial radio access network suffers from limited coverage and rigid network structure, which cannot timely handle the task offloading requirements in remote areas. MEC-enabled 6G SAGIN has a three-layer network that includes ground-based, air-based and space-based networks, providing MUs with all-time, all-domain services by linking multiple nodes such as ground base stations, drones and satellites [7].

MEC-enabled 6G SAGIN is a dynamic, flexible multi-node network architecture. Therefore, optimizing the task offloading problem in 6G SAGIN networks has many challenges. Firstly, due to the limited computational resources of the edge nodes and temporal variability of the wireless network environment, the MUs need to select one of the multiple offloading methods and identify a particular edge node for task offloading. Yang et al. [13] utilized a multi-task learning approach in MEC to select appropriate edge nodes for computational offloading. Shen et al. [8] proposed a collaborative task offloading framework based on slicing for space-air-ground integrated vehicular networks and optimized the allocation of slicing resources through a deep reinforcement learning (DRL) algorithm. Secondly, when multiple MUs select the same edge node, the allocation of computational resources to each MUs needs to be optimized. Waqar et al. [12] proposed a MEC-enabled integrated aerial-terrestrial vehicle network and optimized the computational resource allocation to reduce the computational and communication overheads. Cao et al. [1] studied the problem of minimum energy consumption in low Earth orbit (LEO) satellite edge-assisted multilayer MEC systems. Tang et al. [9] proposed a reinforcement learning based traffic offloading scheme in SAGIN. Finally, edge nodes in 6G SAGIN are not stationary. When the mobility of MUs and mobile base stations (MBSs) makes the network topology change, new offloading methods need to be determined and computational resources need to be reallocated, Liu et al. [5] consider the effect of frequent movement of MUs on the selection of optimal edge nodes in mobile edge computing.

Many studies have discussed task offloading strategies in SAGIN, however, the mobility of MUs and MBSs, the problem for offloading method selection in the overlapping coverage area of edge nodes, and the complexity of multiple communication links in different edge nodes have not been fully considered. In this paper, we propose a task offloading framework in 6G SAGIN and solve the optimization problem involved using a DRL algorithm. Specifically, our contribution is as follows:

– A MEC-enabled 6G SAGIN task offloading framework is proposed. Considering the mobility of MUs and MBSs, the diversity of communication methods, and the heterogeneity of edge nodes, we described the task offloading process and model the system in 6G SAGIN.
– In order to improve the quality of experience (QoE) of MUs, we propose a problem of minimizing the average cost of MUs based on the system model, and a DRL-based task offloading optimization algorithm (DTOOA) incorporating Markov processes is proposed to jointly optimize the average cost of MUs weighted by latency and energy consumption.
– A number of simulation experiments were conducted to evaluate the performance of the DTOOA scheme. In many different simulation environments, the numerical results show that the DTOOA scheme is more effective than other offloading optimization schemes for reducing the average cost of MUs.

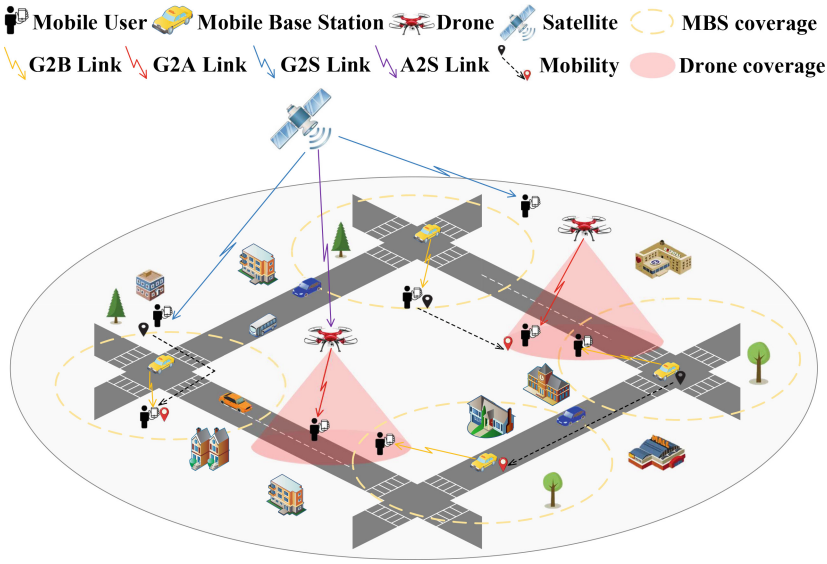


Fig. 1. MEC-enabled 6G Space-Air-Ground Integrated Networks.

2 System Model

Figure 1 shows the MEC-enabled 6G SAGIN scenario, which consists of LEO satellite, drones, and MBSs. In this paper, we optimize tasks offloading process within the satellite coverage. In addition to performing tasks locally, MUs can offload tasks to MBSs, drones or satellite via ground-to-MBS (G2B), ground-to-air (G2A) and ground-to-space (G2S) communication links. In particular, when

the path loss between the MU and the satellite is excessive due to the factors such as signal blocking and occlusion, the MU can first upload the task to the drone via G2A link, and then the drone offloads the task to the satellite via air-to-space (A2S) link.

In this section, we first present an overview of SAGIN, and then provide the detailed definitions of offloading design, mobility model, communication model, and computational model.

2.1 Overview

For Network Architecture. We deployed 6G SAGIN in remote areas without terrestrial fixed cellular base station coverage, thus utilize satellite, drones, and MBSs deployed in vehicles to support MUs for computationally intensive task offloading. In this scenario, we consider a LEO satellite, indexed as $\mathcal{O}$, multiple pre-deployed drones, indexed as $\mathcal{D} = \{1, 2, ..., d, ..., D\}$, multiple MBSs, indexed as $\mathcal{M} = \{1, 2, ..., m, ..., M\}$, and multiple MUs that may need to perform task offloading, indexed by $\mathcal{N} = \{1, 2, ..., n, ..., N\}$. LEO satellite provide seamless coverage of the entire network area, whereas both drones and MBSs have limited coverage. In addition, MBSs and MUs are moving within the network area, and task offloading for MUs can only be provided when the MUs enter the coverage area of the drones or MBSs.

For Time Slot and Task. We define the time slot model as $\mathbb{T} = \{1, 2, ..., t, ..., T\}$, in which the duration of each time slot is ψ, and assume that the network topology, node locations, and channel conditions remain constant over a time slot t. Each MU generates an atomic and non-redivisible computationally intensive task in each time slot t, and each computational task is described by a 3-tuple $\mathcal{K}_n(t) = (k_n^{\mathcal{I}}(t), k_n^{\mathcal{C}}(t), k_n^{\mathcal{L}}(t))$, where $k_n^{\mathcal{I}}(t)$[bits], $k_n^{\mathcal{C}}(t)$[cycles], and $k_n^{\mathcal{L}}(t)$[seconds] represent the input data size of the task, the amount of computation required for the task, and the maximum tolerable latency of the task, respectively.

2.2 Offloading Design

MUs are able to handle small scale tasks locally. However, in order to support compute-intensive applications such as AR/VR, autonomous driving, and ultra-high-definition video, MUs usually need to select edge nodes for task offloading. To minimize the long term average latency and energy consumption of MUs, we propose a three step offloading scheme including offloading request, offloading strategy, and task execution.

Offloading Request. At the beginning of each time slot, each MU sends an offload request to the edge node currently providing the service. If the current MUs is performing the task locally, the MUs will send an offloading request directly to the LEO satellite. The intelligent agent deployed in the satellite aggregate information such as offload requests and edge node computational resources and use the information to make intelligent decisions.

Offloading Strategy. The satellite is the center node in the 6G SAGIN topology, in which the deployed intelligent agent determine the offloading strategy for each time slot. The offloading strategy include determining how each MU task is to be performed and the computational resources allocated to the MUs by each edge node. There are five execution modes for each MU task, including local execution, offloading to a MBS, offloading to a drone, offloading to the LEO satellite, and offloading to the LEO satellite through a drone retransmission. Normally, the MU can offload the task to the satellite directly, however, when the ground-satellite communication signal fades excessively, drones with altitude advantage can be selected as a relay nodes to retransmit the MU task to the satellite for offloading. Of course, when the MU selects the edge node for offloading or retransmission, it needs to satisfy the precondition that the MU is within the coverage area of the edge node. In addition, the optimal choice of nodes needs to be considered when the MU is within the overlapping coverage of multiple edge nodes.

We define $\mathbf{U}_n(t) = \{u_n^{\mathcal{N}}(t), mu_n^{\mathcal{M}}(t), du_n^{\mathcal{D}}(t), u_n^{\mathcal{O}}(t), d'u_n^{\mathcal{R}}(t)\}, m \in \mathcal{M}, d \in \mathcal{D}, d' \in \mathcal{D}, \forall n \in \mathcal{N}$ as the 5-dimensional execution mode vector of the MU n task at time slot t, where $u_n^{\mathcal{N}}(t), u_n^{\mathcal{M}}(t), u_n^{\mathcal{D}}(t), u_n^{\mathcal{O}}(t), u_n^{\mathcal{R}}(t) \in \{0, 1\}$ denote whether the five execution modes are selected, in sequence corresponding to local execution, offloading to a MBS, offloading to a drone, offloading to satellite and offloading to satellite via a drone relay, respectively. The coefficients m and d indicate which specific MBS and drone the task is offloaded to, respectively. The coefficient d' indicates which specific drone is selected for task retransmission. MU n can only choose one task execution method, therefore the execution method needs to satisfy the constraint

$$u_n^{\mathcal{N}}(t) + u_n^{\mathcal{M}}(t) + u_n^{\mathcal{D}}(t) + u_n^{\mathcal{O}}(t) + u_n^{\mathcal{R}}(t) = 1, \forall n \in \mathcal{N}. \tag{1}$$

After the execution of the MU tasks has been determined, the intelligent agent need to determine the appropriate computational resources that edge nodes allocate to MUs. Let $\mathbf{C}_n(t) = \{c_n^{\mathcal{N}}(t), c_n^{\mathcal{M}}(t), c_n^{\mathcal{D}}(t), c_n^{\mathcal{O}}(t), c_n^{\mathcal{R}}(t)\}, \forall n \in \mathcal{N}$ denotes the 5-dimensional computational resource allocation vector, which represents the computational resources allocated to the MU n through local execution, offloading to a MBS, offloading to a drone, offloading to a satellite and offloading to satellite via a drone relay, respectively. To summarize, we define the offloading strategy of MU n at time slot t as

$$a_n(t) = (\mathbf{U}_n(t), \mathbf{C}_n(t)), \forall n \in \mathcal{N}. \tag{2}$$

Based on the definition above, the offloading strategy for all MUs at time slot t in 6G SAGIN scenario is given as

$$a(t) = \{\mathbf{U}(t), \mathbf{C}(t)\} = \{a_1(t), a_2(t), ..., a_n(t), ..., a_N(t)\}. \tag{3}$$

Task Execution. Based on the optimized offloading strategy made by the intelligent agent, the MUs execute the tasks according to the corresponding

execution method and the allocated computational resources. The task execution phase consists of three processes, including task upload, task computation and result return. When the task is executed locally, the task upload and result return processes are omitted.

2.3 Mobility Model

In the SAGIN scenario, the satellite provides global offloading strategy guidance and seamless coverage to support task offloading. Drones with smaller coverage areas are pre-deployed in areas of high computational demand to address the problem of localized computational intensity. MBSs with larger coverage range move randomly within the network range to avoid over-concentration of computing resources and extend the overall service range. The coordinates of MU n, MBS m, drone d, and satellite at time slot t are defined using 3-dimensional Cartesian coordinates $(x_n^{\mathcal{N}}(t), y_n^{\mathcal{N}}(t), 0)$, $(x_m^{\mathcal{M}}(t), y_m^{\mathcal{M}}(t), 0)$, $(x_d^{\mathcal{D}}(t), y_d^{\mathcal{D}}(t), z_d^{\mathcal{D}}(t))$ and $(x^{\mathcal{O}}(t), y^{\mathcal{O}}(t), z^{\mathcal{O}}(t))$ with units of meters, respectively. For simplicity, we assume that LEO satellites are semi-static over in a fixed time frame, and from a practical implementation perspective, it is necessary to federate multiple satellites to ensure the network range can be covered continuously and seamlessly. The drones fixed at pre-deployed locations, and the MUs and MBSs move randomly within the network range according to random vectors $\mathbf{v}_n^{\mathcal{N}}(t) = (\Delta x_n^{\mathcal{N}}(t), \Delta y_n^{\mathcal{N}}(t), 0)$ and $\mathbf{v}_m^{\mathcal{M}}(t) = (\Delta x_m^{\mathcal{M}}(t), \Delta y_m^{\mathcal{M}}(t), 0)$ following Gaussian distribution. Therefore, the coordinates of MU n and MBS m at time slot $t+1$ can be expressed as

$$\begin{aligned}
(x_n^{\mathcal{N}}(t+1), y_n^{\mathcal{N}}(t+1), 0) &= (x_n^{\mathcal{N}}(t), y_n^{\mathcal{N}}(t), 0) + \mathbf{v}_n^{\mathcal{N}}(t) \cdot \min(1, \frac{V_{max}^{\mathcal{N}}}{|\mathbf{v}_n^{\mathcal{N}}(t)|}), \\
(x_m^{\mathcal{M}}(t+1), y_m^{\mathcal{M}}(t+1), 0) &= (x_m^{\mathcal{M}}(t), y_m^{\mathcal{M}}(t), 0) + \mathbf{v}_m^{\mathcal{M}}(t) \cdot \min(1, \frac{V_{max}^{\mathcal{M}}}{|\mathbf{v}_m^{\mathcal{M}}(t)|}),
\end{aligned} \tag{4}$$

where $V_{max}^{\mathcal{N}}$ and $V_{max}^{\mathcal{M}}$ represent the maximum distance that MUs and MBSs can moving during a single time slot, respectively.

Based on the defined coordinate system, we can obtain the distances between MU n and MBS m, drone d, satellite at time slot t as $i_{n,m}^{\mathcal{M}}(t)$, $i_{n,d}^{\mathcal{D}}(t)$, $i_n^{\mathcal{O}}(t)$ respectively. The distance between drone d and satellite at time slot t is $i_m^{\mathcal{R}}(t)$. We define the topological relationship at time slot t between MU n and MBS m as $\alpha_{n,m}^{\mathcal{M}}(t)$ and denote as

$$\alpha_{n,m}^{\mathcal{M}}(t) = \begin{cases} 1, & i_{n,m}^{\mathcal{M}}(t) \le l_m^{\mathcal{M}} \\ 0, & else \end{cases}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \tag{5}$$

where $l_m^{\mathcal{M}}$ signifies the coverage radius of the MBS m. In a similar way, establish the topological association between MU n and drone d at time slot t as $\alpha_{n,d}^{\mathcal{D}}(t) \in \{0, 1\}$, then we have

$$\alpha_{n,d}^{\mathcal{D}}(t) = \begin{cases} 1, & i_{n,d}^{\mathcal{D}}(t) \le l_d^{\mathcal{D}} \\ 0, & else \end{cases}, \forall n \in \mathcal{N}, \forall d \in \mathcal{D}, \tag{6}$$

where $l_d^{\mathcal{D}}$ denotes the coverage radius of the drone d. Because of the mobility of MUs and MBSs, the set of edge nodes that each MU can request for task offloading may change at each time slot. We define the MBSs and drones vectors that can be requested by the MU n at time slot t as $\mathbf{M}_n(t)$ and $\mathbf{F}_n(t)$, respectively, then we have

$$\begin{aligned} \mathbf{M}_n(t) &= \{\alpha_{n,1}^{\mathcal{M}}(t), ..., \alpha_{n,M}^{\mathcal{M}}(t)\}, \forall n \in \mathcal{N}, \\ \mathbf{D}_n(t) &= \{\alpha_{n,1}^{\mathcal{D}}(t), ..., \alpha_{n,D}^{\mathcal{D}}(t)\}, \forall n \in \mathcal{N}. \end{aligned} \tag{7}$$

Based on the definition above, the optional task execution mode vector of MU n at time slot t can be defined as

$$\mathbf{O}_n(t) = \{o_n^{\mathcal{N}}(t), \mathbf{M}_n(t), \mathbf{D}_n(t), o_n^{\mathcal{O}}(t), \mathbf{D}_n(t)\}, \forall n \in \mathcal{N}, \tag{8}$$

where $o_n^{\mathcal{N}}, o_n^{\mathcal{O}} = 1$ means that the MU can always choose the two task execution modes of local execution or offloading to satellite. When MU n selects a task execution mode at time slot t, it must be satisfied that the selected execution mode is within the optional task execution mode vector. For a given execution mode vector $\mathbf{U}_n(t) = \{u_n^{\mathcal{N}}(t), mu_n^{\mathcal{M}}(t), du_n^{\mathcal{D}}(t), u_n^{\mathcal{O}}(t), d'u_n^{\mathcal{R}}(t)\}, m \in \mathcal{M}, d \in \mathcal{D}, d' \in \mathcal{D}, \forall n \in \mathcal{N}$, we define the execution mode index as $I_n^{\mathcal{U}}(t)$ and expressed as

$$\begin{aligned} I_n^{\mathcal{U}}(t) = u_n^{\mathcal{N}}(t) + (1+m)u_n^{\mathcal{M}}(t) + (1+M+d)u_n^{\mathcal{D}}(t) \\ + (2+M+D)u_n^{\mathcal{O}}(t) + (2+M+D+d')u_n^{\mathcal{R}}(t), \end{aligned} \tag{9}$$

where M, D denote the number of MBSs and drones, respectively. Considering the optional task execution mode vector is $(2+M+2D)$ dimensional and based on the execution mode index $I_n^{\mathcal{U}}(t)$, we can get the execution mode selection constraint $\mathbf{O}_n(t)[I_n^{\mathcal{U}}(t) - 1] = 1, \forall n \in \mathcal{N}$, and at this point, the index of the optional task execution mode vector starts from 0.

2.4 Communication Model

When MUs execute tasks by offloading to edge nodes, the task input data $k_n^{\mathcal{I}}(t)$ needs to be uploaded through the uplink. In SAGIN, we consider four types of communication links including G2B, G2A, G2S and A2S. Each MU is equipped with a single antenna and each edge node such as MBS, drone and satellite are equipped with multiple antennas. To better suppress the intra-cell interference, MUs use orthogonal frequency division multiple access (OFDMA) technology to access the edge nodes in the uplink.

G2B Communication Links. According to the OFDMA technology, the operating band $B^{\mathcal{M}}$ of each MBS is evenly partitioned into N sub-bands, and the size of each sub-bands is $W_n^{\mathcal{M}} = B^{\mathcal{M}}/N$ [MHz]. The non-overlapping sub-bands will be allocated to MUs connected to the same MBS, which avoids intra-cell interference of each MBS in the uplink. However, considering the random mobility of MBSs and MUs, it is essential to take into account the interference generated by different MUs communicating with different MBSs when coverage overlaps.

Based on the description above, the MU n to MBS m wireless uplink rate at time slot t in the G2B communication link can be expressed as

$$R_{n,m}^{\mathcal{M}}(t) = W_n^{\mathcal{M}} \log_2 \left(1 + \frac{p_n^{\mathcal{N}} |h_{n,m}^{\mathcal{M}}|^2 (i_{n,m}^{\mathcal{M}}(t))^{-\iota}}{\sigma_{G2B}^2 + \zeta_{n',m'}^{\mathcal{M}}(t)} \right), \tag{10}$$

where $p_n^{\mathcal{N}}$ is transmission power of MU n, $h_{n,m}^{\mathcal{M}}$ denotes the unity channel gain between MU n and MBS m at the reference distance of $i_{n,m}^{\mathcal{M}}(t) = 1$ m, ι represents the path loss exponent and σ_{G2B}^2 indicates the noise power in G2B communication link. $\zeta_{n',m'}^{\mathcal{M}}(t)$ represents the interference from other MUs employing the same sub-band as MU n within the same coverage area but connected to different MBSs. Then we have

$$\zeta_{n',m'}^{\mathcal{M}}(t) = \sum_{m' \in \mathcal{M}/\{m\}} \alpha_{n,m'}^{\mathcal{M}}(t) \sum_{n' \in \mathcal{N}/\{n\}} \alpha_{n',m'}^{\mathcal{M}}(t) \eta_{n,n'}(t) p_{n'}^{\mathcal{N}} |h_{n',m'}^{\mathcal{M}}|^2 (i_{n',m'}^{\mathcal{M}}(t))^{-\iota}, \tag{11}$$

where $\eta_{n,n'}(t) \in 0, 1$, $\eta_{n,n'}(t) = 0$ signifies that MU n and MU n' are operating on different sub-bands at time slot t, while $\eta_{n,n'}(t) = 1$ indicates that MU n and MU n' are utilizing the same sub-band at time slot t.

G2A Communication Links. In the SAGIN, drones are pre-deployed in hotspot areas and the coverage of the drones are non-overlapping, so there is no inter-cell interference between different drones in G2A communications. Utilizing the OFDMA technology, the transmission rate between MU n and drone d in G2A communication link at time slot t is calculated as

$$R_{n,d}^{\mathcal{D}}(t) = W_n^{\mathcal{D}} \log_2 \left(1 + \frac{p_n^{\mathcal{N}} |h_{n,d}^{\mathcal{D}}|^2 (i_{n,d}^{\mathcal{D}}(t))^{-\iota}}{\sigma_{G2A}^2} \right), \tag{12}$$

where $W_n^{\mathcal{D}} = B^{\mathcal{D}}/N$ is the channel sub-bandwidth of the drones, similar to the G2B communication link, $B^{\mathcal{D}}$ is the operating band of each drone and divided equally into N sub-bands. $h_{n,d}^{\mathcal{D}}$ denotes the unity channel gain between MU n and drone d at the reference distance of $i_{n,d}^{\mathcal{D}}(t) = 1$ m, σ_{G2A}^2 indicates the noise power in G2A communication link.

G2S and A2S Communication Links. In the G2S and A2S communication link, the distance from the satellite to MUs and drones is much larger than the moving distance of MUs and drones, therefore, we ignore the influence of mobility on the G2S and A2S communication rate. The communication rate from MU n and drone d to the satellite at time slot t can be calculated as

$$R_n^{\mathcal{O}}(t) = W_n^{\mathcal{O}} \log_2 \left(1 + \frac{p_n^{\mathcal{N}} 10^{\frac{-L_n^{\mathcal{O}}(t)}{10}}}{\sigma_{G2S}^2} \right), \tag{13}$$

$$R_d^{\mathcal{R}}(t) = W_d^{\mathcal{R}} \log_2 \left(1 + \frac{p_d^{\mathcal{D}} 10^{\frac{-L_d^{\mathcal{R}}(t)}{10}}}{\sigma_{A2S}^2} \right), \tag{14}$$

where $W_n^{\mathcal{O}}$ and $W_d^{\mathcal{R}}$ is the size of channel sub-bands allocated by the satellite to MU n and drone d, respectively. $p_d^{\mathcal{D}}$ is transmission power of drone d, $L_n^{\mathcal{O}}(t)$ and $L_d^{\mathcal{R}}(t)$ denotes the path loss from MUs and drones to the satellite at time slot t, σ_{G2S}^2 and σ_{A2S}^2 indicates the noise power in G2S and A2S communication link. It is worth noting that the rate of the G2S communication link is lower than that of the G2A and A2S communication links, due to the longer transmission distance and the higher path loss.

2.5 Computation Model

For the offloading strategy $a_n(t) = (\mathbf{U}_n(t), \mathbf{C}_n(t))$ of MU n at time slot t, task $\mathcal{K}_n(t) = (k_n^{\mathcal{I}}(t), k_n^{\mathcal{C}}(t), k_n^{\mathcal{L}}(t))$ has five execution modes, including local execution, offloading to a MBS, offloading to a drone, offloading to satellite and offloading to satellite via a drone relay.

Local Execution. We define the task latency of MU n local execution as $\tau_n^{\mathcal{N}}(t)$, and $\tau_n^{\mathcal{N}}(t)$ only consists of the task computation latency, which can be calculated as

$$\tau_n^{\mathcal{N}}(t) = k_n^{\mathcal{C}}(t) / c_n^{\mathcal{N}}(t), \tag{15}$$

where $c_n^{\mathcal{N}}(t)$ is the computational resource of MU n at time slot t. Following the widely used energy consumption model [13], the energy consumption for local execution by MU n at time slot t can be calculated as

$$e_n^{\mathcal{N}}(t) = \varsigma (c_n^{\mathcal{N}}(t))^2 k_n^{\mathcal{C}}(t), \tag{16}$$

where ς represent the energy efficiency parameter, which varies based on the chip architecture. According to (15) and (16), the weighted cost that combines latency and energy consumption for locally executed tasks $\mathcal{K}_n(t)$ can be expressed as

$$\Phi_n^{\mathcal{N}}(t) = \lambda \tau_n^{\mathcal{N}}(t) + (1 - \lambda) e_n^{\mathcal{N}}(t), \forall n \in \mathcal{N}, \tag{17}$$

where $\lambda \in [0, 1]$ denotes the preference of MU n for balancing latency and energy consumption. For instance, if an underpowered MU aims to conserve battery life, it may lower the value of λ. Conversely, the parameters are adjusted to prioritize the latency.

Offloading to MBS, Drone or Satellite. We assume that MBSs, drones, and satellite can concurrently provide task offloading for multiple MUs, and we quantify the computational resources as the number of CPU cycles per second. If MU n offloading the task $\mathcal{K}_n(t) = (k_n^{\mathcal{I}}(t), k_n^{\mathcal{C}}(t), k_n^{\mathcal{L}}(t))$ to MBSs, drones or satellite, there are normally three processes to consider, including task upload, task computation, and result return. In common, the size of the task results is much

smaller than the size of the task inputs $k_n^{\mathcal{I}}(t)$. Furthermore, the downlink rate significantly surpasses the uplink rate, so we can neglect the delay and energy consumption associated with the result return phase. Based on the description above, the total delay of MU n offloading task $\mathcal{K}_n(t)$ to MBS m, drone d and satellite at time slot t can be calculated as

$$\begin{aligned}
\tau_{n,m}^{\mathcal{M}}(t) &= k_n^{\mathcal{I}}(t)/R_{n,m}^{\mathcal{M}}(t) + k_n^{\mathcal{C}}(t)/c_n^{\mathcal{M}}(t), \\
\tau_{n,d}^{\mathcal{D}}(t) &= k_n^{\mathcal{I}}(t)/R_{n,d}^{\mathcal{D}}(t) + k_n^{\mathcal{C}}(t)/c_n^{\mathcal{D}}(t), \\
\tau_n^{\mathcal{O}}(t) &= k_n^{\mathcal{I}}(t)/R_n^{\mathcal{O}}(t) + k_n^{\mathcal{C}}(t)/c_n^{\mathcal{O}}(t),
\end{aligned} \tag{18}$$

where $c_n^{\mathcal{M}}(t)$, $c_n^{\mathcal{D}}(t)$ and $c_n^{\mathcal{O}}(t)$ are the computational resources allocated to the MU n by the MBS, drone and satellite based on the offloading strategy made by the intelligent agent. The total energy consumption of MU n offloading task $\mathcal{K}_n(t)$ to MBS m, drone d and satellite at time slot t is obtained as

$$\begin{aligned}
e_{n,m}^{\mathcal{M}}(t) &= p_n^{\mathcal{N}} k_n^{\mathcal{I}}(t)/R_{n,m}^{\mathcal{M}}(t) + p_n^{\mathcal{W}} k_n^{\mathcal{C}}(t)/c_n^{\mathcal{M}}(t), \\
e_{n,d}^{\mathcal{D}}(t) &= p_n^{\mathcal{N}} k_n^{\mathcal{I}}(t)/R_{n,d}^{\mathcal{D}}(t) + p_n^{\mathcal{W}} k_n^{\mathcal{C}}(t)/c_n^{\mathcal{D}}(t), \\
e_n^{\mathcal{O}}(t) &= p_n^{\mathcal{N}} k_n^{\mathcal{I}}(t)/R_n^{\mathcal{O}}(t) + p_n^{\mathcal{W}} k_n^{\mathcal{C}}(t)/c_n^{\mathcal{O}}(t),
\end{aligned} \tag{19}$$

where $p_n^{\mathcal{W}}$ represents the power of MU n during the waiting for result return phase. Based on (18) and (19), the delay and energy weighted-cost of the offloading task $\mathcal{K}_n(t)$ to the MBS m, drone f and satellite at time slot t can be calculated as

$$\begin{aligned}
\Phi_n^{\mathcal{M}}(t) &= \lambda \tau_{n,m}^{\mathcal{M}}(t) + (1-\lambda) e_{n,m}^{\mathcal{M}}(t), \forall n \in \mathcal{N}, \\
\Phi_n^{\mathcal{D}}(t) &= \lambda \tau_{n,d}^{\mathcal{D}}(t) + (1-\lambda) e_{n,d}^{\mathcal{D}}(t), \forall n \in \mathcal{N}, \\
\Phi_n^{\mathcal{O}}(t) &= \lambda \tau_n^{\mathcal{O}}(t) + (1-\lambda) e_n^{\mathcal{O}}(t), \forall n \in \mathcal{N},
\end{aligned} \tag{20}$$

in practical terms, when $\lambda = 1$, the objective is to minimize the delay for MU n, whereas $\lambda = 0$ is aimed at significantly enhancing the battery life of MU n.

Offloading to Satellite via Drone Relay. When the G2S communication link is obstructed or the path loss is excessive, MUs can offload the task to satellite via a drone relay. In this case, the MU needs to upload the task to the drone first, then the drone offloading the task to the satellite for execution. The result return phase is again omitted, and the total task execution latency for MU n is

$$\tau_{n,d}^{\mathcal{R}}(t) = k_n^{\mathcal{I}}(t)/R_{n,d}^{\mathcal{D}}(t) + k_n^{\mathcal{I}}(t)/R_d^{\mathcal{R}}(t) + k_n^{\mathcal{C}}(t)/c_n^{\mathcal{R}}(t), \tag{21}$$

where $c_n^{\mathcal{R}}(t)$ is the computational resources assigned to the MU n through offloading to satellite via a drone relay. In this case, the total energy consumption of MU n at time slot t can be obtained as

$$e_{n,d}^{\mathcal{R}}(t) = p_n^{\mathcal{N}} k_n^{\mathcal{I}}(t)/R_{n,d}^{\mathcal{D}}(t) + p_n^{\mathcal{W}} \left(k_n^{\mathcal{I}}(t)/R_d^{\mathcal{R}}(t) + k_n^{\mathcal{C}}(t)/c_n^{\mathcal{R}}(t) \right), \tag{22}$$

similar to the definitions previously, the weighted-cost of latency and energy consumption in this case can be calculated as

$$\Phi_n^{\mathcal{R}}(t) = \lambda \tau_{n,d}^{\mathcal{R}}(t) + (1-\lambda) e_{n,d}^{\mathcal{R}}(t), \forall n \in \mathcal{N}. \tag{23}$$

Let $C^{\mathcal{M}}$, $C^{\mathcal{D}}$ and $C^{\mathcal{O}}$ indicate the overall computational resources of MBSs, drones and the satellite. Based on the definition above, we can get the computational resource constraint at time slot t as

$$\sum_{n\in\mathcal{N}} c_n^{\mathcal{M}}(t) \le C^{\mathcal{M}}, \sum_{n\in\mathcal{N}} c_n^{\mathcal{D}}(t) \le C^{\mathcal{D}}, \sum_{n\in\mathcal{N}} \left(c_n^{\mathcal{O}}(t) + c_n^{\mathcal{R}}(t)\right) \le C^{\mathcal{O}}. \tag{24}$$

3 Problem Formulation

In this section, we formulate the multi-node collaborative task offloading optimization problem to jointly reduce the task computation latency and energy consumption of the MUs. Based on the models above and the offloading strategy $a(t)$ made by the intelligent agent for all MUs at time slot t, we can define the average cost of MUs weighted by delay and energy consumption at time slot t as

$$\overline{\Phi}(t) = \frac{1}{N}\sum_{n\in\mathcal{N}} \left(u_n^{\mathcal{N}}(t)\Phi_n^{\mathcal{N}}(t) + u_n^{\mathcal{M}}(t)\Phi_n^{\mathcal{M}}(t) + u_n^{\mathcal{D}}(t)\Phi_n^{\mathcal{D}}(t) + u_n^{\mathcal{O}}(t)\Phi_n^{\mathcal{O}}(t) + u_n^{\mathcal{R}}(t)\Phi_n^{\mathcal{R}}(t)\right). \tag{25}$$

Our goal is to minimize the average cost of MUs through optimizing the offloading strategies made by the intelligences at each time slot. Based on the description above, considering the constraints, including execution modes, tolerance delay, computational resources, coverage of edge nodes and mobility of MUs and MBSs. We formulate the problem of minimizing the average cost of MUs over a T-slot time horizon as

$$\begin{aligned}
&\mathbf{P1}: \min_{a(t)} \frac{1}{T}\sum_{t\in\mathbb{T}} \overline{\Phi}(t) \qquad (26)\\
\text{s.t. } &C1: u_n^{\mathcal{N}}(t), u_n^{\mathcal{M}}(t), u_n^{\mathcal{D}}(t), u_n^{\mathcal{O}}(t), u_n^{\mathcal{R}}(t) \in \{0,1\}, \forall n \in \mathcal{N},\\
&C2: u_n^{\mathcal{N}}(t) + u_n^{\mathcal{M}}(t) + u_n^{\mathcal{D}}(t) + u_n^{\mathcal{O}}(t) + u_n^{\mathcal{R}}(t) = 1, \forall n \in \mathcal{N},\\
&C3: |\mathbf{v}_n^{\mathcal{N}}(t)| \le V_{max}^{\mathcal{N}}, |\mathbf{v}_m^{\mathcal{M}}(t)| \le V_{max}^{\mathcal{M}}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M},\\
&C4: \mathbf{O}_n(t)[I_n^{\mathcal{U}}(t) - 1] = 1, \forall n \in \mathcal{N},\\
&C5: u_n^{\mathcal{N}}(t)\tau_n^{\mathcal{N}}(t) + u_n^{\mathcal{M}}(t)\tau_{n,m}^{\mathcal{M}}(t) + u_n^{\mathcal{D}}(t)\tau_{n,d}^{\mathcal{D}}(t)\\
&\qquad\qquad + u_n^{\mathcal{O}}(t)\tau_n^{\mathcal{O}}(t) + u_n^{\mathcal{R}}(t)\tau_{n,d}^{\mathcal{R}}(t) \le k_n^{\mathcal{L}}(t), \forall n \in \mathcal{N},\\
&C6: 0 \le c_n^{\mathcal{M}}(t) \le C^{\mathcal{M}}, 0 \le c_n^{\mathcal{D}}(t) \le C^{\mathcal{D}}, 0 \le c_n^{\mathcal{O}}(t), c_n^{\mathcal{R}}(t) \le C^{\mathcal{O}}, \forall n \in \mathcal{N},\\
&C7: \sum_{n\in\mathcal{N}} c_n^{\mathcal{M}}(t) \le C^{\mathcal{M}}, \sum_{n\in\mathcal{N}} c_n^{\mathcal{D}}(t) \le C^{\mathcal{D}}, \sum_{n\in\mathcal{N}} \left(c_n^{\mathcal{O}}(t) + c_n^{\mathcal{R}}(t)\right) \le C^{\mathcal{O}}, \forall n \in \mathcal{N},
\end{aligned}$$

where $C1$ and $C2$ show that MUs can only choose a single execution mode at time slot t. $C3$ ensures that the distance moved by MUs and MBSs does not exceed the maximum limit at time slot t. $C4$ indicates that MUs can only select optional execution methods to perform tasks. $C5$ represents that the execution delay of tasks does not exceed the maximum tolerable delay. $C6$ and $C7$ mean that the computational resources allocated to the MUs and the total used computational resources are not exceeding the resource limits of the edge nodes, respectively.

4 DRL-Based Task Offloading Optimization Scheme

P1 is a mixed-integer nonlinear programming problem involving binary and continuous variables, so it is challenging to obtain an optimal or suboptimal solution. DRL has strong learning ability and adaptivity to handle highly complex problems, thus we use MDP-based Proximal Policy Optimization (PPO) algorithm to solve **P1**. In this section, we introduce the MDP model and the PPO framework, then we propose the DTOOA scheme to optimize the cost of MUs.

4.1 MDP Model

An MDP model can be characterized by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ represents the set of system states, and $s(t)$ indicates the system state at time slot t. $\mathcal{A}$ show the set of system actions, and $a(t)$ denotes the system action at time slot t. $\mathcal{P}$ signifies the probability of transitioning from state $s(t)$ to $s(t+1)$ when action $a(t)$ is taken, and $\mathcal{R}$ represents the immediate reward associated with action $a(t)$.

State Space. At the offloading request phase, the intelligent agent summarizes the state space consisting of offload requests, wireless network state and computational resources of edge nodes. Then state $s(t) \in \mathcal{S}$ at time slot t can be expressed as

$$s(t) = \{\mathbf{K}(t), \mathbf{O}(t), \mathbf{R}(t), \mathbf{C}(t)\}, \tag{27}$$

where $\mathbf{K}(t) = (\mathbf{k}^{\mathcal{I}}(t), \mathbf{k}^{\mathcal{C}}(t), \mathbf{k}^{\mathcal{L}}(t))$ denotes the set of task information for all MUs at time slot t, including input data size, computing amount, and maximum acceptable latency. $\mathbf{O}(t)$ represents the set of optional task execution mode vector for all MUs. $\mathbf{R}(t)$ is the set of all communication link rates, including the G2B, G2A and G2S communication rates for all MUs and the A2S communication rate for all drones. $\mathbf{C}(t)$ is the set of computational resources available for all the edge nodes. It is worth noting that the changes in the coordinates of all nodes are embodied in $\mathbf{O}(t)$ and $\mathbf{R}(t)$, so the state space can omit the coordinates of the nodes.

Action Space. At the offloading strategy phase, the intelligent agent aggregate system state information and takes action $a(t) \in \mathcal{A}$, $a(t)$ was defined previously as

$$a(t) = \{\mathbf{U}(t), \mathbf{C}(t)\}, \tag{28}$$

where $\mathbf{U}(t)$ is execution mode vector of all MUs at time slot t, and $\mathbf{C}(t)$ denotes the computational resource allocation vector of all MUs.

System Reword P1. strives to minimize the average cost for MUs, whereas MDP is commonly employed to maximize the long-term rewards of the system.

Consequently, we can leverage MDP to maximize the negative value of the average cost of MUs. To encourage the intelligent agent to select actions aligning with the delay constraints $C5$ of MUs, we configure the reward as

$$r(t) = \begin{cases} -\overline{\Phi}(t), & the\ C5\ is\ satisfied, \\ -N^*(t)\overline{\Phi}(t), & else, \end{cases} \tag{29}$$

where $N^*(t)$ denotes the count of MUs that fall short of meeting constraint $C5$ at time slot t. If constraint $C5$ is satisfied, the immediate reward corresponds to the negative value of the average user cost. Conversely, if constraint $C5$ is not met, the immediate reward takes the form of a penalty that is calculated as the negative of the average user cost multiplied by the number of MUs failing to satisfy condition $C5$. As a result, the intelligent agent exhibits a preference for actions that ensure compliance with constraint $C5$ at time slot t.

4.2 PPO Framework

In the MDP model above, the computational resource allocation vectors in the action space are continuous, so we propose the DTOOA scheme based on the PPO framework to find the optimal task execution mode vector and computational resource allocation vector for MUs. The PPO is an on-policy, model-free DRL algorithm which improves the stability by controlling the magnitude of policy updates during training. PPO framework belongs to the actor-crtic network structure, which normally has three neural networks, including the actor network π_θ, the old actor network $\pi_{\theta_{old}}$ and the crtic network Q_φ, where θ, θ_{old} and φ denote the parameters of actor network, old actor network and crtic network, respectively.

The actor network is the policy network, which is responsible for outputting the probability distribution of actions in a given state $s(t)$. The old actor network is used to compute the ratio of new to old policies, thus limiting the changing magnitude of the new policy. The actor network is updated by gradient ascent method $\theta \leftarrow \theta + \xi \cdot \nabla_\theta \mathcal{L}(\theta)$, where ξ is the learning rate and $\mathcal{L}(\theta)$ is the objective function. In order to ensure the stability of training, PPO defines a clipped surrogate objective function, which is expressed as

$$\mathcal{L}(\theta) = \mathbb{E}\left[\min\left(\frac{\pi_\theta(a(t)|s(t))}{\pi_{\theta_{old}}(a(t))|s(t))} \cdot \hat{A}(t), clip\left(\frac{\pi_\theta(a(t)|s(t))}{\pi_{\theta_{old}}(a(t))|s(t))}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}(t)\right)\right], \tag{30}$$

where $\hat{A}(t)$ is the advantage function, which is used to evaluate whether the new strategy is better than the old one. Clip function is used to limit the ratio $\frac{\pi_\theta(a(t)|s(t))}{\pi_{\theta_{old}}(a(t))|s(t))}$ to the interval $[1-\epsilon, 1+\epsilon]$, and ϵ is a hyper-parameter that limits the ratios of the new to old policies. The advantage function is obtained from the critic network computation which defined as

$$\hat{A}(t) = \sum_{j=1}^{T-t} \gamma^j \cdot r(t+j) - Q_\varphi(s(t)), \tag{31}$$

where T denotes the maximum time slot, $r(t+j)$ represents the reward for future steps, and γ is the discount factor. $Q_\varphi(s(t))$ is the state-valued function output by the critic network. Critic network is updated by gradient descent method $\varphi \leftarrow \varphi - \xi \cdot \nabla_\varphi \mathcal{L}(\varphi)$ and loss function $\mathcal{L}(\varphi)$ can be expressed as

$$\mathcal{L}(\varphi) = \mathbb{E}\left[\left(\sum_{j=1}^{T-t} \gamma^j \cdot r(t+j) - Q_\varphi(s(t))\right)^2\right]. \tag{32}$$

4.3 DRL-Based Task Offloading Optimization Algorithm

The intelligent agent deployed in the satellite run the DTOOA algorithm, which collects environmental information from 6G SAGIN and makes offloading decisions. In the early stage of operation, the DTOOA algorithm needs to be trained for several iterations and reach convergence, after which the intelligent agent starts to make actual decisions.

Algorithm 1. DRL-Based Task Offloading Optimization Algorithm (DTOOA)

Initialize the actor network π_θ, the old actor network $\pi_{\theta_{old}}$ and the crtic network Q_φ with parameters θ, $\theta_{old} \leftarrow \theta$ and φ;
Initialize the replay memory buffer $\mathcal{B}$;
for *episode* $= 1$ *to* $MaxEpisode$ **do**
  Initialize the 6G SAGIN environment and obtain the initial state $s(1)$;
  for $t = 1$ *to* T **do**
    Execute action $a(t)$ generated by observing state $s(t)$ through the actor network π_θ;
    Calculate reward $r(t)$ based on (29) and get the next state $s(t+1)$;
    Save experience $(s(t), a(t), r(t), s(t+1))$ to the replay memory buffer $\mathcal{B}$;
  end for
  for each *experience* in replay memory buffer $\mathcal{B}$ **do**
    Update the actor network π_θ by computing the objective function (30) using a single experience;
    Update the critic network Q_φ by calculate the loss function (32) using a single experience;
  end for
  Update the old actor network $\pi_{\theta_{old}}$ with the parameter θ;
  Clear the experience replay buffer $\mathcal{B}$;
end for

At the start of time slot t, the intelligent agent makes an action $a(t)$ depending on the current state $s(t)$, gets the next state $s(t+1)$ and receives an immediate reward $r(t)$. Then, experience $(s(t), a(t), r(t), s(t+1))$ is saved to the replay memory buffer $\mathcal{B}$. After a episode, iterating through each experience in the replay memory buffer $\mathcal{B}$ to updates the actor and critic networks. Finally, the old actor network is updated and the replay memory buffer is cleared. For clarity, Algorithm 1 summarizes the details of DTOOA.

5 Numerical Results and Discussion

In this section, we utilize Python 3.9 and TensorFlow 2.0 to create a simulation platform and construct the 6G SAGIN model to evaluate the performance of our proposed DTOOA scheme. We first investigate the impact of internal parameters on the performance of the DTOOA scheme. Subsequently, we proceed to compare DTOOA against the other three schemes within various environments.

5.1 Simulation Setup

In the simulation scenario, we consider a 500 m radius circular network range with seamless satellite coverage, and the satellite $\mathcal{O}$ coordinate is $(0, 0, 300000)$ with units of meters. The coordinates of the $D = 2$ drones are $(-250, -250, 120)$ and $(250, 250, 120)$, and the coverage radius $l_d^{\mathcal{D}}$ is 100 m. The number of MUs in the range of $[10, 25]$ and $M \in \{2, 4, 6, 8, 10\}$ MBSs with $l_m^{\mathcal{M}} = 200$ m coverage radius stochastically distributed in the network range. The duration of each time slot ψ is 1 s, MBS and MU are moved within each time slot with maximum movement distances $V_{max}^{\mathcal{M}} = 10$ m and $V_{max}^{\mathcal{N}} = 3$ m, respectively. With reference to the communication and computational parameter settings in [1,2] and [3], Table 1 summarizes the default simulation parameters in our system model.

Table 1. Default Simulation Parameters.

Parameters	Values
Computational resources of each satellite/drone/MBS/MU	20/8/10/1 GHz
Operating band of each satellite/drone/MBS	24/5/10 MHz
Size of the task input data $k_n^{\mathcal{I}}(t)$	2–20 MB
Amount of computation required for the task $k_n^{\mathcal{C}}(t)$	0.5–3 Gigacycles
Maximum tolerable latency of the task $k_n^{\mathcal{L}}(t)$	0.8–1 s
Transmission power of each MU/drone $p_n^{\mathcal{N}}/p_d^{\mathcal{D}}$	0.2/1.6 W
Waiting power of each MU $p_n^{\mathcal{W}}$	0.05 W
Unity channel gain $h_{n,m}^{\mathcal{M}}/h_{n,d}^{\mathcal{D}}$	0/−50 dB
Path loss exponent ι	4
Background noise power $\sigma_{G2B}^2/\sigma_{G2A}^2$	−104/−104 dBm
Background noise power $\sigma_{G2S}^2/\sigma_{A2S}^2$	−114/−114 dBm
Path loss $L_n^{\mathcal{O}}(t)/L_d^{\mathcal{R}}(t)$	150–174/150 dB
Energy efficiency parameter ς	10^{-28}
Preference for latency and energy consumption λ	0.5

For DTOOA, both the actor and critic networks consist of four-layer neural architectures comprising an input layer, two hidden layers, and an output layer. The actor network features 512 neurons in both of its hidden layers, whereas

the critic network is composed of two hidden layers with 512 and 256 neurons, respectively. Besides, The $MaxEpisode$ is 1000 and the T is 20, the discount factor γ is 0.9, the ϵ used for clipping the surrogate objective is 0.2, and the mini-batch size is 64.

5.2 Parameter Analysis

In the simulated environment with $M = 6$ MBSs and $N = 10$ MUs, $k_n^{\mathcal{C}}(t)$, $k_n^{\mathcal{C}}(t)$ and $k_n^{\mathcal{L}}(t)$ obeying Gaussian distributions with mean values of 11 MB, 1.75 Gigacycles and 0.9 s, the path loss $L_n^{\mathcal{O}}(t)$ takes random values in the interval $[150, 174]$ dB, we evaluated the impact of learning rates on the convergence of the DTOOA scheme and conducted multiple experiments in the learning rate interval $[10^{-2}, 10^{-5}]$. We found that DTOOA converges only when the learning rate is near 1×10^{-4}. Figure 2 illustrates the convergence of the algorithm for learning rate is 5×10^{-4}, 1×10^{-4}, and 0.5×10^{-4}. With a learning rate set at 5×10^{-4}, the system learning rate becomes excessively close to 1, leading to an inability to learn the optimal solution and consequently resulting in a low system reward. When learning rate is 0.5×10^{-4}, the cumulative reward of the algorithm attains convergence after 700 iterations, exhibiting a slower convergence rate compared to the optimal scheme. When learning rate is 1×10^{-4}, the cumulative reward and convergence speed show notable improvements compared to the other cases, so we use 1×10^{-4} as the learning rate for subsequent simulation experiments.

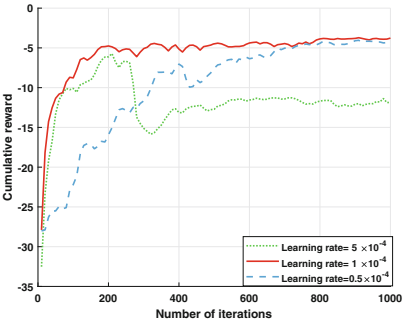


Fig. 2. Impact of different learning rate.

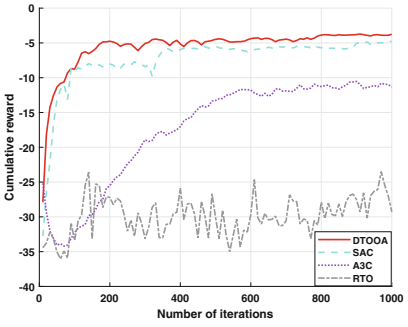


Fig. 3. Impact of various comparison algorithm.

5.3 Performance Analysis

To assess the algorithmic performance of DTOOA, we conduct a comparative evaluation against Soft Actor-Critic (SAC), Asynchronous Advantage Actor-Critic (A3C), and Random Task Offloading (RTO). Both SAC and A3C are

DRL algorithms based on actor-critic structure, SAC combines the features of off-policy and maximum entropy to improve the convergence [14] and A3C improves the training utility and training speed by asynchronous training [11]. In RTO, the actions are given randomly under the condition that the constraints are satisfied. In the performance analysis simulations, we set the default parameters as $M = 6$ MBSs and MUs, $N = 10$ MUs, $k_n^{\mathcal{C}}(t)$, $k_n^{\mathcal{C}}(t)$ and $k_n^{\mathcal{L}}(t)$ obeying Gaussian distributions with mean values of 11 MB, 1.75 Gigacycles and 0.9 s, the path loss $L_n^{\mathcal{O}}(t)$ takes random values in the interval $[150, 174]$ dB. In subsequent simulations, when analyzing the impact of a particular parameter on the long-term reward of the system, the default parameters were used for all other parameters.

System Reward Trend. With the simulation environment set as the default parameters, we evaluated the convergence trend of the cumulative rewards for the four schemes with the number of iterations. As shown in Fig. 3, our proposed DTOOA scheme has optimal convergence rewards and convergence speeds, and the cumulative rewards after convergence are 2.98% better than SAC and 21.62% better than A3C. Due to the clipping of the surrogate objective function in PPO, the DTOOA scheme has excellent stability and convergence, and can better cope with the situation that the state space changes abruptly, such as when the mobility of the MUs and MBSs leads to an abrupt change for the optional task execution mode vector $\mathbf{O}(t)$ of the MUs. The SAC scheme has higher cumulative rewards after convergence, but the convergence rate is inferior to the DTOOA scheme. The A3C scheme is inferior to the DTOOA and SAC schemes in terms of convergence speed and cumulative rewards, and the RTO scheme has no convergence trend.

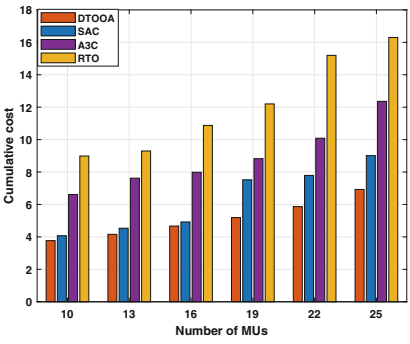


Fig. 4. Impact of different number of MUs.

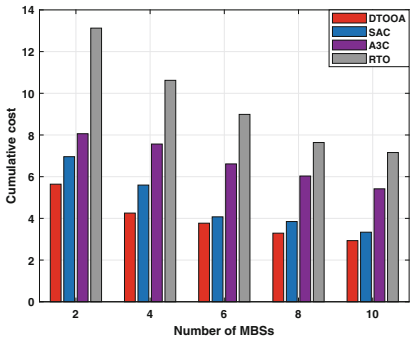


Fig. 5. Impact of different number of MBSs.

Number of MUs. In order to more visually represent the improvement in QoE of MUs by DTOOA, we compare the impact of the four schemes on the cumulative cost instead of the cumulative reward after convergence. As shown in Fig. 4, with the default parameter settings and setting the number of MUs to $[10, 25]$ with an increment of 3, we evaluate the impact of different numbers of MUs on the cumulative cost. The task computation requirements increase with the number of MUs, while the available wireless and computational resources are limited, thus the latency and energy cost for the MUs are increased. When the number of MUs increases to 25, the DTOOA scheme reduces the cumulative cost by 23.31% and 43.97% compared to the SAC and A3C schemes, respectively.

Number of MBSs. In the simulation environment with default parameter settings and with the numbers of MBSs set to $[2, 10]$ in increments of 2, Fig. 5 gives the trend of cumulative cost with different numbers of MBSs. The more the number of MBSs, the more edge nodes can be requested and the more computing resources can be allocated in the SAGIN system, thus reducing the cumulative cost and improving the QoE of MUs. When the number of MBSs is 2, compared to the SAC, A3C and RTO schemes, the DTOOA scheme reduces the cumulative cost by 18.97%, 30.02% and 56.97%, respectively.

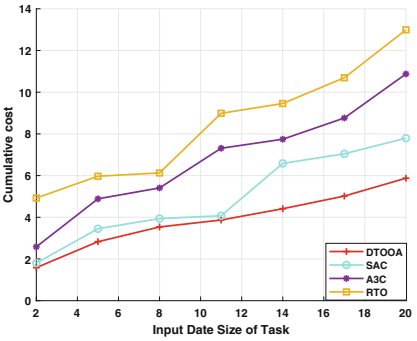


Fig. 6. Impact of different input data size of task.

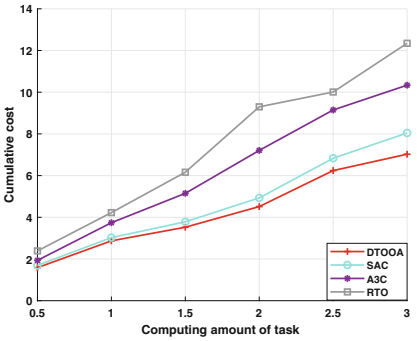


Fig. 7. Impact of different computing amount of task.

Input Data Size of Task. As shown in Fig. 6, in the simulation environment with default parameter settings and with the size of the task input data set to [2,20] in increments of 2, we analyze the impact of different task input data size on cumulative cost. The increase in the input data size of task leads to larger communication latency and energy consumption, which makes the cumulative cost increase. The curve of the DTOOA scheme is smooth, so the DTOOA scheme is insensitive to changes in the input data size of task and it is able to optimize task offloading in a stable manner. In addition, the DTOOA scheme reduces the

cumulative cost by 24.28% and 45.98% compared to the SAC and A3C schemes, respectively. Compared to the RTO scheme, it reduces the cumulative cost by 54.77%.

Computing Amount of Task. In the simulation environment where the computing amount of task is $[0.5, 3]$ Gigacycles in increments of 0.5 and other parameters are default parameters, The impact of different computing amount of task on the cumulative cost is shown in Fig. 7. Due to the restricted computing resources of the edge nodes, the larger the task computation volume, the higher the task computation latency and energy consumption, which makes the cumulative cost increase. The DTOOA scheme reduces cumulative costs by 12.66%, 32.02% and 41.21% over the SAC, A3C and RTO schemes, respectively.

6 Conclusion and Future Work

In this paper, we study the task offloading optimization problem in the 6G SAGIN scenario. In particular, we consider the modeling of multiple heterogeneous edge nodes and multiple offloading methods, meanwhile, the mobility of MUs and MBSs are considered. Temporally changing network environment and complex offloading scenario make the optimal task offloading problem challenging. First, we provide a detailed description of the task offloading process and establish various models in the SAGIN scenario. Then, we propose the multi-node collaborative task offloading optimization problem to jointly reduce the task computation latency and energy consumption of MUs. Finally, we propose the DTOOA scheme based on the Proximal Policy Optimization (PPO) framework to optimize the average cost of MUs, and conduct extensive simulation experiments. Numerical results show that the DTOOA scheme clearly outperforms the Soft Actor-Critic (SAC), Asynchronous Advantage Actor-Critic (A3C), and Random Task Offloading (RTO) schemes in a variety of different simulation environments, and can significantly reduce the average cost of MUs and improve the QoE of MUs.

In future work, considering a joint multi-satellite network to provide seamless assisted computation and caching for a region is a potential direction. In addition, modeling MBSs and drones for path planning to provide better service is also a viable option.

References

1. Cao, X., et al.: Edge-assisted multi-layer offloading optimization of LEO satellite-terrestrial integrated networks. IEEE J. Sel. Areas Commun. **41**(2), 381–398 (2023)
2. Gong, Y., Yao, H., Xiong, Z., Guo, S., Yu, F.R., Niyato, D.: Computation offloading and energy harvesting schemes for sum rate maximization in space-air-ground networks. In: GLOBECOM 2022 - 2022 IEEE Global Communications Conference, pp. 3941–3946 (2022)

3. He, L., Li, J., Wang, Y., Zheng, J., He, L.: Balancing total energy consumption and mean makespan in data offloading for space-air-ground integrated networks. IEEE Trans. Mob. Comput. **23**(1), 209–222 (2024)
4. Jin, X., Hua, W., Wang, Z., Chen, Y.: A survey of research on computation offloading in mobile cloud computing. Wireless Netw. **28**(4), 1563–1585 (2022)
5. Liu, H., Cao, G.: Deep reinforcement learning-based server selection for mobile edge computing. IEEE Trans. Veh. Technol. **70**(12), 13351–13363 (2021)
6. Ning, Z., Yang, Y., Wang, X., Guo, L., Gao, X.: Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing. IEEE Trans. Mob. Comput. **22**(5), 2628–2644 (2023)
7. Qin, P., Wang, M., Zhao, X., Geng, S.: Content service oriented resource allocation for space-air-ground integrated 6G networks: a three-sided cyclic matching approach. IEEE Internet Things J. **10**(1), 828–839 (2023)
8. Shen, H., Tian, Y., Wang, T., Bai, G.: Slicing-based task offloading in space-air-ground integrated vehicular networks. IEEE Trans. Mob. Comput. **23**(5), 4009–4024 (2024)
9. Tang, F., Hofner, H., Kato, N., Kaneko, K., Yamashita, Y., Hangai, M.: A deep reinforcement learning-based dynamic traffic offloading in space-air-ground integrated networks (SAGIN). IEEE J. Sel. Areas Commun. **40**(1), 276–289 (2022)
10. Wang, C.X., You, X., Gao, X., Zhu, X., Li, Z.: On the road to 6G: visions, requirements, key technologies, and testbeds. IEEE Commun. Surv. Tutor. **25**(2), 905–974 (2023)
11. Wang, Z., Li, M., Zhao, L., Zhou, H., Wang, N.: A3C-based computation offloading and service caching in cloud-edge computing networks. In: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1–2 (2022)
12. Waqar, N., Hassan, S.A., Mahmood, A., Dev, K., Do, D.T.: Computation offloading and resource allocation in MEC-enabled integrated aerial-terrestrial vehicular networks: a reinforcement learning approach. IEEE Trans. Intell. Transp. Syst. **23**(11), 21478–21491 (2022)
13. Yang, B., Cao, X., Bassey, J., Li, X.: Computation offloading in multi-access edge computing: a multi-task learning approach. IEEE Trans. Mob. Comput. **20**(9), 2745–2762 (2021)
14. Zhou, X., Huang, L., Ye, T., Sun, W.: Computation bits maximization in UAV-assisted MEC networks with fairness constraint. IEEE Internet Things J. **9**(21), 20997–21009 (2022)

Securing Wireless Communication in Critical Infrastructure: Challenges and Opportunities

Jörn Bodenhausen[1(✉)], Christian Sorgatz[2], Thomas Vogt[1], Kolja Grafflage[3], Sebastian Rötzel[4], Michael Rademacher[2], and Martin Henze[1,2]

[1] Security and Privacy in Industrial Cooperation, RWTH Aachen University, Aachen, Germany
{bodenhausen,vogt,henze}@spice.rwth-aachen.de
[2] Cyber Analysis & Defense, Fraunhofer FKIE, Wachtberg, Germany
{christian.sorgatz,michael.rademacher,martin.henze}@fkie.fraunhofer.de
[3] ATS Elektronik GmbH, Wunstorf, Germany
kolja.grafflage@atsonline.de
[4] Bonn-Netz GmbH, Bonn, Germany
sebastian.roetzel@bonn-netz.de

Abstract. Critical infrastructure constitutes the foundation of every society. While traditionally solely relying on dedicated cable-based communication, this infrastructure rapidly transforms to highly digitized and interconnected systems which increasingly rely on wireless communication. Besides providing tremendous benefits, especially affording the easy, cheap, and flexible interconnection of a large number of assets spread over larger geographic areas, wireless communication in critical infrastructure also raises unique security challenges. Most importantly, the shift from dedicated private wired networks to heterogeneous wireless communication over public and shared networks requires significantly more involved security measures. In this paper, we identify the most relevant challenges resulting from the use of wireless communication in critical infrastructure and use those to identify a comprehensive set of promising opportunities to preserve the high security standards of critical infrastructure even when switching from wired to wireless communication.

Keywords: Critical Infrastructure · Wireless Communication · Security

1 Introduction

Critical infrastructure encompasses all physical and cyber assets that are essential to maintain vital societal functions. Common examples are smart grids (i.e., electricity and water distribution) [43,67] and smart city services (i.e., intelligent traffic and transportation) [22,74]. Over the past decades, this infrastructure

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 333–352, 2024.
https://doi.org/10.1007/978-3-031-63989-0_17

has transformed towards highly digitized and interconnected intelligent systems [15,68,73] to capitalize on benefits such as efficiency improvements, better visibility into the network, and increased flexibility [6], while at the same time meet increasing complexity, e.g., resulting from distributed energy resources [5,36].

Traditionally, critical infrastructure relied on private wired communication infrastructure to interconnect assets (e.g., fiber optic networks) [36]. More recently, we observe an increasing incorporation of wireless communication in critical infrastructure, supplementing and partially replacing wired communication, fueled by various complementing developments [24]: ① Significant advances in the fundamental technology underlying wireless communication enable increased throughput and low latency [40,51], energy efficient transmissions over long-distances [49], and a significantly higher density of devices [34]. ② A fundamental shift to highly digitized and interconnected systems, where many existing facilities are not connected to a private wired network yet and new emerging entities that do not have an established dedicated wired network at all [36]. ③ The emergence of new usage scenarios where wired communication is technically or economically infeasible, e.g., in the context of smart cities and Internet of Things (IoT) deployments, where large numbers of mobile and resource-constrained devices are deployed over large areas [59].

While the resulting increasing use of *wireless* communication in critical infrastructure provides tremendous benefits, it also poses serious challenges for security. Most importantly, critical communication might now take place over publicly accessible or at least visible networks outside the physical control of the critical infrastructure operator, much unlike the traditional security assumptions of dedicated private networks with strong perimeter protection [36]. Consequently, to sustainably capitalize on the advantages of wireless communication in critical infrastructure, a thorough understanding of these resulting security challenges as well as opportunities to address them is required.

Related Work. Various works study and systematize the security challenges resulting from modern wireless communication (not specific to critical infrastructure) [52,61,66], (mostly) wired communication in distinct critical infrastructure [3] such as water systems [67] or power grids [36,68], as well as highly digitized and interconnected systems such as smart cities [22,43] and the (industrial) IoT [11,59]. While these works provide a comprehensive overview of the respective research fields, they do not generalize the unique challenges of securing wireless communication in critical infrastructure. Contrary, different works focus on isolated challenges and possible solutions to address these, e.g., by analyzing path-loss of electromagnetic waves in urban environments [49], studying the performance of Transport Layer Security (TLS) for IoT [50,54], evaluating low latency but secure communication for the IoT [28], and securing IoT devices with secure elements [58]. Still, these works do not focus on deriving a comprehensive picture of the challenge and solution space. This contrast results in a gap between an overview of relevant challenges for the particular case of wireless communication in critical infrastructure and the potential benefit from solutions for individual challenges.

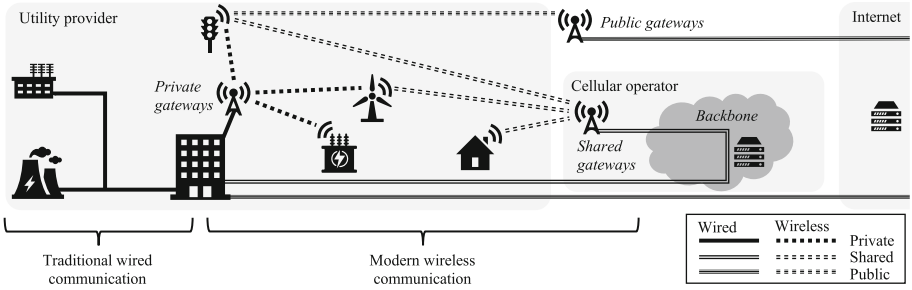


Fig. 1. Communication in critical infrastructure shifts from dedicated private wired networks to heterogeneous wireless communication, encompassing all means of private, public, and shared communication networks as well as dual-homed devices.

Contributions. In this paper, we fill this gap by comprehensively studying the relevant challenges resulting from the use of wireless communication in critical infrastructure and identifying opportunities to uphold a strong level of security even under these challenging conditions. More precisely, our contributions are:

1. We systematize the use of wireless communication in critical infrastructure and provide an overview of relevant technology and its properties (Sect. 2).
2. We categorize and summarize security challenges that are specifically applicable to wireless communication in critical infrastructure (Sect. 3).
3. We identify and discuss complementary opportunities to address the challenges of secure wireless communication in critical infrastructure (Sect. 4).

2 Wireless Communication in Critical Infrastructure

In recent years, critical infrastructure increasingly relies on wireless communication [3]. To lay the foundation to identify and address resulting security challenges, we first identify developments and motivations that fuel this trend (cf. Sect. 2.1), before we introduce and discuss various wireless communication technology that is especially relevant for critical infrastructure (cf. Sect. 2.2).

2.1 From Wired to Wireless Communication

For decades, critical infrastructure relied on private *wired* communication networks [36], as exemplified for a typical (municipal) utility provider in the left part of Fig. 1. With advancements in wireless communication technology and the push towards digitization, critical infrastructure increasingly relies on a confluence of heterogeneous *wireless* communication networks, which are no longer necessarily private, but potentially shared with other critical infrastructure or the public (right part of Fig. 1). While wireless communication will likely serve as a complement to existing wired communication, it mainly serves as an enabler for novel use cases and a remedy for the increasing need for communication capabilities of assets. Notably, the choice of a wireless communication technology

within one critical infrastructure will neither be homogeneous nor exclusive, i.e., multiple technologies will coexist and individual devices or assets might connect over different wireless networks [74].

Exemplarily focusing on a (municipal) utility provider as depicted in Fig. 1, we discuss and highlight the reason and motivation as well as different incarnations of the shift to wireless communication. Most notably, the shift towards an increasing use of wireless communication would not be possible without tremendous advancements in the underlying technologies [34,51]. These advancements have improved wireless networks, enhancing reliability, efficiency, and capacity, making their integration into critical infrastructure more feasible.

Complementing these developments, there has been a fundamental transformation towards highly digitized and interconnected systems, exemplified by the emergence of smart grids [13], where existing facilities that were previously devoid of communication requirements now need connectivity. Moreover, an increasing amount of entities such as heat pumps, tiny solar power installations, and charging stations have arisen that lack an established dedicated wired network altogether. This trend is carried to the extreme by completely new smart city scenarios (i.e., smart recycling and parking, intelligent transportation [74]) where wired communication may prove technically or economically unviable.

Overall, the shift from wired to wireless communication enables a multitude of new use cases and applications. To illustrate this, we will briefly reference the most prominent applications for secure wireless communication in critical infrastructure. In general, applications can be grouped into categories such as energy, environment, industry, living, and services [74]. Here, critical infrastructure mainly affects energy [13,62], industry [42], and services [62]. The dominant critical infrastructure application is smart grids [13,62], i.e., the monitoring of energy generation, transmission, and consumption [35,36,68]. In the field of services, applications arise due to the possibility of mobile sensors, which allows, e.g., for the monitoring of transportation services [62]. Moreover, the paradigm of Industry 4.0 leads to interesting use cases and demands for very dense wireless networks to interconnect machines and computational capabilities [20,42].

In summary, the incorporation of wireless communication within critical infrastructure has gained traction due to significant technological advancements, the shift towards interconnected systems such as smart grids, and the rise of new scenarios where wired communication is infeasible. As depicted in Fig. 1, in contrast to wired communication which predominantly relies on dedicated private fiber optic and copper wire networks, wireless communication in critical infrastructure can rely on a plethora of cellular radio technology in private, public, and shared deployments.

2.2 Heterogeneous Wireless Networks

The range of different cellular radio technologies potentially suitable for critical infrastructure is immense. The most common technologies are Long Term Evolution (LTE), Narrowband IoT (NB-IoT), Global System for Mobile Communications (GSM), Terrestrial Trunked Radio (TETRA), Long Range Wide

Area Network (LoRaWAN), Wi-Fi, 450 MHz LTE Machine Type Communication (LTE-M), 5G and in the future 6G. Choosing a suitable technology is challenging since they have widely differing characteristics and capabilities.

What all these wireless networks have in common is, that they are cellular radio networks, i.e., their topology can be summarized as interconnected base stations. Each base station typically serves a certain area, while overlaps are possible and desired. Base stations require an exposed location, a stable power supply, and a backhaul connection typically realized via wired networks.

The characteristics and capabilities of the different technologies are mainly determined by the maturity level of the technology, the frequency used, and the available bandwidth. At Sub-GHz bands (e.g., 450 MHz LTE-M or LoRaWAN), a higher range (cell size) and better penetration of structures can be achieved. However, the bandwidth is scarce which reduces throughput and scalability and therefore limits the potential applications [17,50,65]. At higher frequency bands (e.g., mmWave for 5G), the throughput is comparable or even superior to wired technologies but the range of each base station and device is limited [51].

In wireless communication, frequency bands are heavily regulated. There are frequency bands that are strictly assigned to certain entities, i.e., Mobile Network Operators (MNOs), while others are generally accessible. Receiving the exclusive right to use a certain band (e.g., 450 MHz LTE-M, 5G) requires significant investment costs, which need to be generated. However, due the exclusive control, the operator can also guarantee a certain Quality of Service (QoS). License-exempt bands reduce the operation costs (e.g., Wi-Fi, LoRaWAN). However, particularly in urban scenarios, these bands can become uncontrollably crowded which compromises their applicability for critical infrastructure [48].

Similar considerations hold for the operation model of the different wireless technologies. As visualized in Fig. 1, there are three possibilities: public, shared, and private. Interestingly, there are technology advancements such as 5G network slicing which blur the boundaries between public and private network infrastructures [38]. *Public* network infrastructure is operated by entities (MNOs) not directly associated with critical infrastructure. 3rd Generation Partnership Project (3GPP) technologies (LTE, 5G) are widely used and the network is shared with others (e.g., smartphones). Recently, a new type of public network infrastructure, driven by a shared community approach, emerged. Most prominently, The Things Network [64] builds a wide-range LoRaWAN network. *Shared* infrastructure is operated by entities directly associated to critical infrastructure. While building and operating a wireless network is not feasible for a single critical infrastructure operator, multiple operators can form an association and share costs, e.g., the nationwide 450 MHz LTE network for critical infrastructure in Germany [8]. Compared to public networks, the network is not shared outside the association. *Private* networks are directly operated by critical infrastructure operators. These networks are limited to certain areas where a huge device density renders a network economically feasible (e.g., plants, smart cities) or where there is a lack of other solutions (e.g., offshore wind farms). Technologies operating in license-exempt bands are preferred (e.g., Wi-Fi, LoRaWAN), however, private 5G campus networks provide an interesting alternative.

Overall, it can be stated that due to their versatility, heterogeneous wireless networks offer enormous potential to be utilized for a wide variety of applications. However, this versatility also leads to significant complexity that affects operators and their clients. The huge diversity in wireless technologies, frequency regulation, and operation models not only complicates the deployment and operation of wireless communication in critical infrastructure but also fundamentally challenges security.

3 Challenges of Securing Wireless Communication

Wireless communication in critical infrastructure places high demands on security. At the same time, various significant challenges must be addressed that differ depending on the particular setting and influences such as operation models, particular technologies, network topology, and utilized devices. Thus, each particular use case can shift the focus to certain challenges or create new ones.

Methodology. We strive to identify those challenges that are particularly relevant to wireless communication in critical infrastructure. To this end, we study various literature providing an overview of security challenges resulting from various aspects relevant to wireless communication in critical infrastructure: (i) modern wireless communication in general, i.e., not focusing on critical infrastructure [52,61,66] (ii) wired communication in specific, isolated branches of critical infrastructure [3,36,67,68], as well as (iii) fully interconnected and digitized intelligent (future) systems [10,11,22,43,59]. We complement these more holistic surveys with highly specialized works focusing on isolated challenges of securing wireless communication in critical infrastructure, especially with respect to (iv) network and device constraints alongside various dimensions [16,49,50,54], (v) a wide range of security challenges [14,43,47,55,60], as well as (vi) legal and deployment considerations [2,3].

Overall, using this methodology, we identify twelve overarching challenges of securing wireless communication in critical infrastructure, which we broadly categorize into four groups: ① constraints of the utilized network and devices (Sect. 3.1), ② security of networks (Sect. 3.2), ③ security of the utilized devices (Sect. 3.3), and ④ application-specific challenges (Sect. 3.4). To provide the foundation to adequately address these challenges, in the following, we further discuss and analyze them in more detail.

We intentionally chose this ordering and categorization to make the section structured and coherent. Our ordering does not necessarily resemble the difficulty or importance of the identified challenges, which mostly depends on the specific situation and perspective. Thus, some of those challenges are more fundamental and tangible while others might be considered rather conceptual. For instance, the challenges grouped in the categories ① and ③ could be considered fundamental, as these stem from the devices as well as the environment and are thus entirely rigid. The challenges in the categories ② and ④ on the other hand are not as rigorous and depend more on design decisions and requirements than on actual limitations. In any case, a detailed use case analysis of each particular

application and its environment is necessary to determine the magnitude and importance of each challenge and thus derive a specific prioritization.

3.1 Network and Device Constraints

Securing wireless communication is first and foremost challenged by constraints of the employed network and devices. Most importantly, wireless networks are challenged w.r.t. reliability and mobility. Furthermore, especially in IoT settings, both network and devices can be restricted with regard to available resources such as bandwidth, computing power, or energy.

C1 - Reliability is a crucial aspect for critical infrastructure, where seamless communication and real-time data transmission are paramount to prevent disruptions and ensure smooth operation and safety. Any interruptions or failures in the wireless network can have far-reaching consequences, including power outages, equipment malfunctions, and delays in emergency response. Therefore, building and maintaining a reliable mobile network infrastructure is essential to ensure the smooth operation of critical infrastructure systems. Due to the inherent characteristics of wireless communication, this is particularly difficult to achieve as wireless connections are susceptible to various environmental factors that can drastically affect signal strength and quality. Most importantly, obstacles such as terrain, buildings, or other structures as well as distance can reduce the reliability of the wireless connection by causing signal attenuation and propagation loss [49]. Furthermore, wireless networks are prone to interference caused by other radio technologies operating on the same frequencies or from inter-cell interference caused by the same technology [16]. These difficulties must be considered during the planning of any mobile network intended for use in critical infrastructure.

C2 - Mobility additionally leads to non-uniform connection quality that depends on the device position and corresponding local influences on reliability, influenced by factors such as distance from the base station, physical obstructions, device interference, and signal propagation characteristics [49]. This variability in network quality poses challenges for both mobile nodes and devices that are permanently deployed in unfavorable positions. Seamless roaming between different coverage areas is critical to maintaining an uninterrupted connection when devices move between cells. A challenge for devices in motion is the potentially increased power consumption when they remain connected to a cell at its edge [63], despite a better signal being available from another nearby cell [33]. The ability to switch between cells presents an opportunity for attackers to target User Equipments (UEs) by using fake/rogue base stations [31]. Another challenge with mobility is the availability of wireless technology in any location where the device may be present. Therefore, not only roaming between cells but also between wireless technologies is an important aspect to address. The security vulnerabilities of all supported technologies become critical when devices have the ability to switch between them, especially if they can be forced into switching from external sources. For instance, if 5G coverage is limited, then a device or UE can downgrade to an earlier generation of mobile communication,

such as LTE or GSM, so that the device can maintain uninterrupted access to the network.

C3 - Network Limitations imposed by the particular technology can further restrict the network beyond the challenges of wireless technologies in general. Most notably, the throughput is limited compared to a wired network since throughput requires bandwidth and spectrum is scarce. This limitation holds for licensed (e.g., 450 MHz LTE [41]) as well as unlicensed bands (e.g., LoRaWAN [50]) and in particular restricts the acceptable bandwidth overhead that can be added by security mechanisms. Furthermore, devices compete for this bandwidth, which becomes a problem when particularly many devices are part of the network, as the available bandwidth is shared among all devices and thus even more limited. This challenge was exemplarily illustrated for LoRaWAN, where numerous devices might even make a timely TLS connection establishment infeasible [50]. As a result, latency is also an issue, as it takes longer to transmit the full message, which can have significant implications for safety-related mechanisms such as an emergency shutdown. This is reinforced by the technology-specific maximum message size (e.g., 256 bytes for LoRaWAN), through which larger messages might be fragmented and delayed even if the wireless network is currently not busy. Furthermore, the communication range of different wireless technologies can differ significantly and thus restrict their usability (cf. Section 2.2).

C4 - Device Limitations further impose challenges beyond the limitations of the network. Especially in an IoT-like setting, device resources such as compute power, memory, storage, or energy can be rather restricted [7]. Such limitations can have a significant impact on the system. For instance, they can limit the applicability of authentication and encryption mechanisms, especially asymmetric ones, as these are commonly computationally expensive [54]. Memory constraints limit the possible complexity of a system and its ability to cache information. Moreover, especially for battery-powered devices, energy efficiency is of utmost importance. As transmission and reception are particularly energy-intensive, messages should be kept as short as possible and bandwidth efficiency is crucial. All those factors also limit the applicability of mechanisms that address security concerns.

3.2 Network Security Concerns

While the challenges discussed so far primarily affect the usability of wireless technology, concerns also arise for the security of transmitted data. Naturally, conventional security considerations, i.e., the CIA triad consisting of confidentiality, integrity, and availability, also apply here. Moreover, proper authentication is a fundamental challenge that needs to be addressed.

S1 - Confidentiality and Integrity protect data from unauthorized access (confidentiality) and unauthorized alteration (integrity). Such protection is particularly relevant for wireless communication in critical infrastructure, where sensitive data from both the customer and the critical infrastructure operator

is transmitted over a wireless medium that can be accessed by anyone in communication range. In some cases, such as the transmission of generic machine instructions, confidentiality might be optional, but integrity is of utmost importance to mitigate the manipulation and insertion of messages. However, assuring confidentiality and integrity can be challenging in constrained environments, as suitable security mechanisms add overhead [43]. Furthermore, it must be considered that hop-by-hop mechanisms, as usually integrated into wireless technology, do not suffice if the subsequently traversed infrastructure is not private, often calling for the use of end-to-end security.

S2 - Availability ensures that resources can be accessed when needed. Unlike wired networks, the wireless medium can be used easily and unrestricted by anyone in range, allowing for a multitude of attacks on availability. Malicious interference in the form of jamming can cause a disturbance in the wireless network such that legitimate communication is partially or completely disrupted [46]. Moreover, energy depletion of battery-powered devices is possible by preventing the system from sleeping [47]. Hence, protecting wireless networks from such attacks is exceptionally challenging, especially considering critical infrastructure.

S3 - Authentication ensures that only authorized devices participate in communication. While wired networks can be secured to some extent through their architecture and segmentation, cryptographic methods must inevitably be used in a wireless network to ensure that only authenticated devices can access it. Such cryptographic methods usually rely on asymmetric mechanisms, which are demanding for resource-constrained devices [36]. Furthermore, proper authentication is required in various contexts (e.g., network connection, end-to-end security, system updates [27]). Overall, such methods must therefore be designed in a resource-efficient way and devices must receive appropriate credentials.

3.3 Device Security and Challenges

The changes in security requirements and network environment due to wireless communication in critical infrastructure also impose challenges on individual devices. One such challenge is the more difficult deployment with cryptographic material. As credentials weaken over time or become otherwise insecure, revocation mechanisms must be utilized. Moreover, easier physical access of adversaries to devices deployed in more or less public spaces must be considered.

A1 - Deployability of devices is significantly impeded by the need for credentials for subsequent authentication. In particular root credentials are required to authenticate all subsidiary credentials and, moreover, all connections and firmware upgrades. Devices need to be equipped and configured with root credentials during production or deployment [14]. This deployment may be automated or require manual work, but in any case, makes the deployment process more difficult and expensive.

A2 - Device Exclusion allows to remove previously authorized devices from a network, e.g., when credentials get stolen or otherwise compromised as well as when underlying cryptographic primitives lose their promised security level. Especially credential theft could occur rather frequently in critical infrastructure

due to the easier physical access to devices that are increasingly deployed in public spaces. For such cases, there is a need for revocation mechanisms and by extension the exclusion of compromised devices from the network. While there are solutions that address this issue for subsidiary credentials, they usually require trust in the utilized root credentials. Protection and revocation of root credentials thus is an open challenge [25,26,60].

A3 - Physical Access to Devices refers to an adversary's ability to physically extract, e.g., keying material from a device. While devices were conventionally inaccessible on private premises, they are increasingly being deployed in openly accessible spaces [55], e.g., in the context of smart cities. An illustrative example would be smart metering systems, that are deployed in private homes and are thus easily accessible to the residents. Furthermore, the wireless characteristic allows for devices to be moved while still being connected to the network. This results in special requirements and challenges for the physical protection of the devices themselves and the critical information stored on them.

3.4 Application-Specific Requirements

In addition to these challenges that follow from the use of wireless communication in critical infrastructure in general, further use-case-specific requirements might need consideration. Most importantly, this concerns potential legal requirements as well as a certain flexibility to allow for seamless operation.

R1 - Legal Requirements are imposed on a wide range of domains and in particular critical infrastructure. For example, a guideline of the German Federal Office for Information Security (BSI) that is applicable for smart meter infrastructure [2] demands the use of TLS with mutual certificate-based authentication, limits session lifetime to 48 h, and restricts the choice of cryptographic algorithms and parameters. Such requirements can also cover aspects outside security, e.g., functional safety or interoperability, and significantly limit the possibility of specialized and optimized solutions.

R2 - Flexibility ensures the seamless operation of the system. For once, the interoperability of the system must be considered, since the infrastructure might not be controlled by a single entity. Moreover, the updatability of and access to the system, i.e., serviceability, must be taken into account [3]. As it might be impracticable or impossible to access devices that are deployed in the field, remote access and update solutions must be utilized. Furthermore, the use of several wireless technologies in parallel could enhance the redundancy and robustness of the system. Lastly, as the use case might not yet be precisely defined, the system should allow for adaptability to changing conditions.

4 Opportunities to Secure Wireless Communication

To address the identified crucial challenges for securing wireless communication in critical infrastructure, we move forward by identifying and examining potential opportunities to secure wireless communication but also point out their

Table 1. Various opportunities to secure wireless communication in critical infrastructure exist to address the identified fundamental security challenges (marked by ■).

Opportunities to secure wireless communication in critical infrastructure (Section 4) \ **Challenges** of securing wireless communication in critical infrastructure (Section 3)	**Netw./Dev. Constr.** C1 - Reliability	C2 - Mobility	C3 - Network Limit.	C4 - Device Limit.	**Netw. Sec. Concerns** S1 - Conf./Integrity	S2 - Availability	S3 - Authentication	**Dev. Security/Chall.** A1 - Deployability	A2 - Device Exclusion	A3 - Phy. Acc. to Dev.	**App.-Spec. Require.** R1 - Legal Require.	R2 - Flexibility
Network Security												
Lower Layer Security Mechanisms	■	□	■	□	□	□	□	□	□	□	□	□
Secure Communication Channel	□	□	■	■	■	□	□	□	□	□	□	□
Communication Security												
Resource-Conscious E2E Security	□	□	■	■	■	□	□	□	□	□	■	■
E2E Security Extensions	□	□	■	■	□	□	□	□	□	□	□	□
Cryptography Optimizations	□	□	■	■	■	□	□	□	□	□	□	□
Device Security												
Hardware Security Mechanisms	□	□	□	■	□	□	□	□	□	■	■	□
Authenticated Code and Secure Boot	□	□	□	□	□	□	■	□	□	■	□	□
Root of Trust	□	□	□	□	□	□	■	■	■	■	□	■
Network Knowledge												
Propagation Conditions	□	■	■	□	□	■	□	□	□	□	□	□
Context Awareness	□	■	■	■	□	□	□	□	□	■	□	□
Roaming and Redundancy	■	■	□	□	■	■	□	□	□	□	□	□

shortcomings and promising potential for further research. For easier reference, Table 1 gives an overview of the challenges that are addressed by the various discussed approaches, which we structured into four categories: ① security for the network technology (Sect. 4.1), ② security for the encapsulated communication (Sect. 4.2), ③ security of the devices themselves (Sect. 4.3), and ④ enhancing security using detailed knowledge of the network (Sect. 4.4).

4.1 Network Security

Wireless networks are built upon diverse technologies, each equipped with its own set of inherent security mechanisms. As critical infrastructure becomes more interconnected, wireless network security becomes increasingly important.

Lower Layer Security Mechanisms are particularly relevant for wireless communication, i.e., using a shared medium. Even if end-to-end security on a higher layer is utilized, the lower layer security of the particular communication technology has to be addressed, for which various mechanisms exist. With LTE and 5G, etc., radio traffic typically takes place in licensed frequencies and heavy regulations ensure reduced interference from other radio technologies. To further mitigate unintended interference, frequency planning [16] can be used. To combat jamming, employing advanced modulation schemes is an effective countermeasure [32]. Thus, reliability (**C1**) can be improved [3] while addressing network limitations (**C3**).

Secure Communication Channels ensure that data is protected while traversing a network. To secure data transport from the MNO to the customer, a Virtual Private Network (VPN) or a dedicated physical connection can be used. This can be combined with private Access Point Names (APNs) or network slicing [38] on the access part to isolate data from different classes of users [21]. IoT devices typically only require communication with specific servers, allowing to restrict communication, which mitigates the risks of integrating compromised UEs into botnets as it would require communication with unauthorized servers. Such mechanisms address confidentiality and integrity (**S1**) on the network layer by establishing secure channels for data transmission, ensuring that data remains confidential and intact during network traversal. However, on this layer, it is particularly important to address network and device limitations (**C3, C4**). IoT devices often send little data, making it inefficient to send it as an Internet Protocol (IP) packet. Cellular mobile technologies such as LTE-M or NB-IoT provide Non-IP Data Delivery (NIDD) as a security mechanism, where data packets are sent without IP address and other headers [21], eliminating typical IP-based attack scenarios. However, for NIDD and similar approaches, existing implementations for higher-layer protocols need to be adapted to work with changes to the layer directly below [39].

4.2 Communication Security

Security mechanisms and authentication provided by utilized communication technology commonly operate in a hop-by-hop fashion. However, especially for public networks, further protection mechanisms are required to secure the communication between nodes in an end-to-end manner [12]. The commonly used approach to realize this is TLS [53], which is a secure and application-independent protocol that provides confidentiality, integrity protection, and moreover authentication, if a proper root of trust is used for both peers, for end-to-end communication. However, such approaches cause additional overhead, especially with respect to bandwidth and computing power.

Resource-Conscious End-to-End Security aims to provide confidentiality (**S1**) while respecting limitations of devices (**C4**) and the network (**C3**). In critical infrastructure, TLS is often without alternatives due to legal requirements (**R1**) and the requirement for interoperability with other systems (**R2**). However, the overhead of TLS is not fixed and varies depending on the configuration and utilized cryptographic algorithms [50,54], allowing for optimization through proper configuration and selection of parameters [50]. Still, further research is required to determine the most beneficial options in a certain scenario and verify the effectiveness of resulting optimizations. In this context, it has to be considered that some options for optimization come at the cost of reducing the interoperability and updatability.

End-to-End Security Extensions provide further optimizations beyond what is possible through parametrization and algorithm choice alone. For instance, the optional TLS session resumption mechanism makes subsequent connections more efficient by cryptographically linking them to the initial connection [53]. Session

sharing between applications allows to make better use of session resumption, conserve system resources (**C4**), and improve latency [29]. Likewise, latency can be improved by pre-computing certain cryptographic operations [28]. Lastly, caching of information allows to reduce the amount of transmitted data, further countering network limitations (**C3**). For TLS, e.g., an extension allows to cache parts of the handshake, which can then be omitted in subsequent handshakes [56]. Still, further research is required to assess the practical application and interoperability of these mechanisms, in particular for critical infrastructure depending on the device and the network properties like bandwidth. This is required as especially interoperability is low for multiple of the extensions proposed for constrained devices.

Cryptography Optimizations allow to further tailor security when communicating only with a limited set of services (at the cost of flexibility). However, such solutions must be adequate for and tailored to the particular setting and requirements. For instance, if confidentiality is not required, a solution focused solely on integrity protection (**S1**) could significantly reduce resource requirements. To this end, aggregation of message authentication codes [72], especially using progressive message authentication [4,69], improves the bandwidth overhead (**C3**) of integrity protection (**S1**) through cleverly designed shorter integrity tags, where the reduced security of shorter tags is improved upon over time. BP-MAC [71], on the other hand, precomputes integrity protection tags to better utilize computing resources (**C4**) and significantly improve latency. From a different perspective, RePeL [70] leverages unused header fields of (industrial) communication protocols to embed integrity protection tags to retrofit integrity protection (**S1**) in the face of network limitations (**C3**). However, such approaches generally do not allow for communication with arbitrary systems, as is the case for standardized solutions such as TLS. To enable severely resource-constrained devices to still communicate securely using TLS, delegation approaches offload the particularly resource-intensive connection establishment to a more powerful, trusted device [30]. The established connection state is then handed to the constrained device and it can thus benefit from the high level of security. Overall, if the given communication scenario allows for optimizing cryptography, especially resource constraints of devices and networks can be addressed. However, since this might impact interoperability, it should be further examined how those can be integrated into established protocols.

4.3 Device Security

Besides securing the network and communication, also individual devices need to be secured (cf. Sect. 3.3). In particular, the potentially easier physical access to devices in critical infrastructure (**A3**) poses a significant threat. The device security must be considered on various levels to ensure trust in the system. This includes hardware and software security as well as a proper root of trust.

Hardware Security Mechanisms are inevitable for trust into systems that are deployed in hostile environments to protect against manipulation of systems through direct physical access (**A3**). One prominent approach are secure

elements [58], which implement security-related mechanisms such as tamper-proof memory to store keys and certificates as well as secure random number generation in hardware. Furthermore, secure elements can also improve performance through hardware acceleration and thus relieve resource-constrained devices (**C4**). Most notably, such optimizations concern rather resource-intensive algorithms such as those used in TLS, making it easier to meet the legal requirement (**R1**) to use this protocol. However, it is important to consider that embedded secure elements can reduce updatability as only current security functions are implemented in hardware. Further research in the area of hardware security mechanisms could improve on yet unsolved aspects such as the detection and prevention of hardware trojans and the protection against intellectual property theft through reverse engineering [55].

Authenticated Code and Secure Boot are further fundamental building blocks for the security of the overall system. They are used to protect the software integrity of systems and thus, by extension, assure the authenticity (**S3**) of any interaction with that system and mitigate its manipulation through remote or direct physical interaction (**A3**). For this, any code is authenticated via digital signatures or similar, which are then used during the boot process of the system to ensure that only authenticated code is executed [18]. However, a recent paper on secure boot in an automotive setting outlines various challenges that still need to be addressed [57]. In particular, an unbroken and properly rooted chain of trust throughout the entire boot process is of utmost importance.

A Root of Trust is inevitable for authentication (**S3**) across various levels and to enable further secure communication with the system. When relying on root credentials, challenges during deployment (**A1**) and exclusion (**A2**) of devices have to be considered. Certificates, as extensively used in TLS [53], cryptographically link an identity to a public key, which can then be used for authentication and key agreement, but revocation is non-trivial. Subscriber Identity Module (SIM) and embedded SIM (eSIM) cards, are commonly used for authentication in mobile networks, but several security issues arise from a lack of authentication and protection of the communication between SIM and modem [75]. Physical Unclonable Functions (PUF) promise to be an invaluable tool to securely bind keys to a device. Here, developments in the area of nanotechnology-based PUFs and asymmetric keys derived from PUFs are particularly interesting [19]. As PUFs might also provide tamper-resistance, such mechanisms further address the challenge of easier access to the system (**A3**). Moreover, recent open-source projects such as OpenTitan [1], promise to close the remaining gaps between hardware and software security. Here, updatability (**R2**) is also considered on the lowest layers and built into the architecture [18].

4.4 Network Knowledge

Finally, detailed knowledge about a wireless network, especially w.r.t. topology and capabilities, provides ample opportunities to strengthen its secure and reliable use. Beneficial knowledge ranges from general propagation conditions over context awareness of devices to redundancy considerations for applications.

Propagation Conditions lead to an irregular availability (**S2**) of a wireless network. In addition, reaching a desired throughput to tackle network limitations (**C3**) requires sufficient signal power. Therefore, detailed network planning is mandatory for secure wireless communication. For public networks, this planning process is conducted by an MNO and the users are confronted with the results. For shared and private networks, the critical infrastructure operator can optimize the receptions based on its applications and needs. Conducting such an optimization is challenging and requires knowledge about the expected reception at different locations or areas. Propagation models are well researched and can assist in this task [51]. However, conducting a targeted measurement campaign is often desirable [49]. The resulting data (from models or campaigns) can be used to tackle the challenges of availability (**S2**) and mobility (**C2**).

Context Awareness of a device in a wireless network plays a fundamental role in adjusting security mechanisms to current requirements and conditions. The most prominent context is the position (geographical coordinates) of a device, which is particularly relevant when considering mobility (**C2**). Coordinates can be obtained either manually during roll-out (for stationary devices), automatically via technologies such as Global Positioning System (GPS), or even with the communication technology itself [9]. For 6G, the combination of communication and sensing is regarded as one major innovation [37]. Other relevant context informations are the criticality of the application the device is used for, if the device has been recently moved, or even other environmental data such as the temperature. In particular, such context information is useful to detect manipulation or unwanted movement of the devices, which addresses the challenge of easier physical access to devices (**A3**). In addition, context information provides the opportunity to dynamically influence security parameters such as cryptographic algorithms based on the current context [23] to adjust to varying network limitations (**C3**) and available system resources (**C4**).

Roaming and Redundancy through the utilization of multiple wireless network technologies can be highly beneficial for critical infrastructures to improve reliability (**C1**) and availability (**S2**) in case of network failures or targeted attacks such as jamming. To this end, a self-configuring multi-access gateway can be employed to meet the strict requirements in critical infrastructure [45]. In general, there are two possible scenarios for such a gateway. First, multiple technologies are used simultaneously, which is particularly beneficial for mobility (**C2**) but also enhances confidentiality (**S1**). However, a mechanism is needed which distributes communication over the different technologies which increases the system complexity [44]. When this mechanism is aware of the application context, delay-critical applications may use technologies such as 5G while low-priority data is transferred via LoRaWAN. Second, one technology is preferred and the others function purely as backup, thereby increasing the availability (**S2**) without additional system complexity.

5 Discussion and Conclusion

With the increasing trend towards wireless communication in general, a similar tendency and corresponding benefits can be observed for critical infrastructure. Most importantly, wireless communication affords the cost-effective interconnection of a multitude of devices without the need for rigid infrastructure. Moreover, it enables the mobility of devices and their deployment in remote areas. However, the diverse set of available wireless technologies and their different characteristics challenge the application of wireless communication in critical infrastructure. Most importantly, easy access to the transmission medium and the use of shared or public infrastructure give rise to serious security challenges.

In this paper, we categorized the relevant challenges that need to be addressed to enable the use of wireless communication in critical infrastructure alongside various dimensions. Network and device constraints arise from characteristics of wireless communication as well as limitations of the network and the utilized devices. Moreover, network security challenges must be addressed with a focus on (end-to-end) confidentiality and integrity protection, particularly critical availability, and authentication towards the network and communication peers. Novel device security challenges arise especially through the easier physical access to devices but also extend to the deployability and exclusion of devices. Lastly, application-specific requirements cover the need for flexibility of the system w.r.t. interoperability, updatability, and the consideration of future requirements to which a too-inflexible system cannot adapt. This includes the stringent legal and regulatory requirements for critical infrastructure which could further tighten over time.

Current approaches and research trends already provide a multitude of opportunities to secure the wireless communication of critical infrastructure by addressing the identified challenges. Network security mechanisms address reliability and limitations to some extent, but confidentiality and integrity protection are limited to hop-by-hop security. For communication security on a higher layer, several approaches promise to enhance efficiency and address network and device limitations through beneficial configuration, extension, or optimization. Within the scope of device security, several hardware and software security mechanisms can be identified as promising building blocks for the overall security. By combining them in a secure architecture with trust rooted in hardware, an unbroken chain of trust could be achieved. Moreover, detailed network knowledge is particularly promising for addressing challenges such as reliability and mobility.

Our work highlights the opportunities for securing wireless communication in critical infrastructure and identifies avenues for further research to fully capitalize on the benefits of wireless communication in critical infrastructure while guaranteeing strong security. Overall, no single approach is sufficient to overcome all the challenges nor is this necessary. Rather, an interplay of multiple solutions must be strived for to achieve a resilient and secure system. To this end, this paper aims to spark and guide further research efforts toward coherent yet adaptable solutions that address the unique challenges of securing wireless communication in critical infrastructure.

Acknowledgments. This work has been funded by the German Federal Office for Information Security (BSI) under project funding reference numbers 01MO23003A, 01MO23003B, 01MO23003C and 01MO23003D (PlusMoSmart). The responsibility for the content of this publication lies with the authors.

References

1. OpenTitan project. https://opentitan.org
2. Kryptographische Vorgaben für Projekte der Bundesregierung Teil 3: Intelligente Messsyteme. TR 03116-3, Federal Office for Information Security (BSI) (2023)
3. Alcaraz, C., et al.: Critical infrastructure protection: requirements and challenges for the 21st century. Int. J. Crit. Inf. Prot. **8**, 53–66 (2015)
4. Armknecht, F., et al.: ProMACs: progressive and resynchronizing MACs for continuous efficient authentication of message streams. In: ACM CCS (2020)
5. Bader, L., et al.: Comprehensively analyzing the impact of cyberattacks on power grids. In: IEEE EuroS&P (2023)
6. Bader, L., et al.: METRICS: a methodology for evaluating and testing the resilience of industrial control systems to cyberattacks. In: CyberICPS (2023)
7. Bormann, C., et al.: Terminology for constrained-node networks. In: RFC 7228 (2014)
8. Bundesnetzagentur: 450 MHz. www.bundesnetzagentur.de/450mhz
9. Conti, A., et al.: Location awareness in beyond 5G networks. IEEE Commun. Mag. **59**(11), 22–27 (2021)
10. Dahlmanns, M., et al.: Easing the conscience with OPC UA: an internet-wide study on insecure deployments. In: IMC (2020)
11. Dahlmanns, M., et al.: Missed opportunities: measuring the untapped TLS support in the industrial internet of things. In: ACM ASIA CCS (2022)
12. Dahlmanns, M., et al.: Transparent end-to-end security for publish/subscribe communication in cyber-physical systems. In: ACM SaT-CPS (2021)
13. Dileep, G.: A survey on smart grid technologies and applications. Renew. Energy **146**, 2589–2625 (2020)
14. Echeverría, S., et al.: Authentication and authorization for IoT devices in disadvantaged environments. In: IEEE 5th WF-IoT (2019)
15. Eggert, M., et al.: SensorCloud: towards the interdisciplinary development of a trustworthy platform for globally interconnected sensors and actuators. In: Trusted Cloud Computing (2014)
16. Elayoubi, S.E., et al.: Performance evaluation of frequency planning schemes in OFDMA-based networks. IEEE Trans. Wirel. Commun. **7**(5), 1623–1633 (2008)
17. Falanji, R., et al.: Range and capacity of LoRa 2.4 GHz. In: MobiQuitous (2022)
18. Frazelle, J.: Securing the boot process: the hardware root of trust. In: Queue (2020)
19. Gao, Y., et al.: Physical unclonable functions. Nat. Electron. **3**(2), 81–91 (2020)
20. Glebke, R., et al.: A . In: case for integrated data processing in large-scale cyber-physical systemsHICSS (2019)
21. GSMA: security features of LTE-M and NB-IoT networks. Technical Report, GSM Association (GSMA) (2019)
22. Hamid, B., et al.: Cyber security issues and challenges for smart cities: a survey. In: MACS (2019)
23. Hellaoui, H., et al.: Energy efficiency in security of 5G-based IoT: an end-to-end adaptive approach. IEEE Internet Things J. **7**(7), 6589–6602 (2020)

24. Henze, M., et al.: Network security and privacy for cyber-physical systems. In: Security and Privacy in Cyber-Physical Systems: Foundations, Principles, and Applications (2017)
25. Henze, M., et al.: Maintaining user control while storing and processing sensor data in the cloud. IJGHPC **5**(4), 97–112 (2013)
26. Henze, M., et al.: A trust point-based security architecture for sensor data in the cloud. In: Trusted Cloud Computing (2014)
27. Henze, M., et al.: Distributed configuration, authorization and management in the cloud-based internet of things. In: IEEE TrustCom (2017)
28. Hiller, J., et al.: Secure low latency communication for constrained industrial IoT scenarios. In: IEEE LCN (2018)
29. Hiller, J., et al.: The case for session sharing: relieving clients from TLS handshake overheads. In: IEEE LCN Symposium (2019)
30. Hummen, R., et al.: Delegation-based authentication and authorization for the IP-based internet of things. In: IEEE SECON (2014)
31. Hussain, S., et al.: LTEInspector: a systematic approach for adversarial testing of 4G LTE. In: NDSS Symposium (2018)
32. Jaitly, S., et al.: Security vulnerabilities and countermeasures against jamming attacks in wireless sensor networks: a survey. In: Comptelix (2017)
33. Kanwal, K., et al.: Reduced early handover for energy saving in LTE networks. IEEE Commun. Lett. **20**(1), 153–156 (2015)
34. Khalifeh, A., et al.: A survey of 5G emerging wireless technologies featuring LoRaWAN, Sigfox, NB-IoT and LTE-M. In: WiSPNET (2019)
35. Klaer, B., et al.: Graph-based model of smart grid architectures. In: SEST (2020)
36. Krause, T., et al.: Cybersecurity in power grids: challenges and opportunities. Sensors **21**(18), 6225 (2021)
37. Liu, F., et al.: Integrated sensing and communications: towards dual-functional wireless networks for 6G and beyond. In: IEEE J-SAC (2022)
38. Liu, R., et al.: Application of 5G network slicing technology in smart grid. In: ICBAIE (2021)
39. Lukić, M., et al.: Data flow in low-power wide-area IoT applications. In: 2018 26th Telecommunications Forum (TELFOR), pp. 1–4 (2018)
40. Luo, K., et al.: An empirical analysis of the progress in wireless communication generations. In: MobiQuitous (2020)
41. Lutze, L., et al.: A modular framework for evaluating smart grid communication protocols over mobile networks. In: IEEE SmartGridComm (2023)
42. Lyczkowski, E., et al.: Wireless communication in industrial applications. In: ETFA (2019)
43. Mecheva, T., et al.: Cybersecurity in intelligent transportation systems. Computers **9**(4), 83 (2020)
44. Mogensen, R.S., et al.: Selective redundant MP-QUIC for 5G mission critical wireless applications. In: VTC2019-Spring (2019)
45. Mogensen, R.S., et al.: Implementation and trial evaluation of a wireless manufacturing execution system for industry 4.0. In: VTC2019-Fall (2019)
46. Mpitziopoulos, A., et al.: A survey on jamming attacks and countermeasures in WSNs. IEEE Commun. Surv. Tutorials **11**(4), 42–56 (2009)
47. Osanaiye, O.A., et al.: Denial of service defence for resource availability in wireless sensor networks. IEEE Access **6**, 6975–7004 (2018)
48. Rademacher, M., et al.: Quantifying the spectrum occupancy in an outdoor 5 GHz WiFi network with directional antennas. In: WCNC (2018)

49. Rademacher, M., et al.: Path loss in urban LoRa networks: a large-scale measurement study. In: VTC2021-Fall (2021)
50. Rademacher, M., et al.: Bounds for the scalability of TLS over LoRaWAN. In: ITG-Symposium Mobile Communication (2022)
51. Rappaport, T.S., et al.: Overview of millimeter wave communications for fifth-generation (5G) wireless networks-with a focus on propagation models. IEEE Trans. Antennas Propag. **65**(12), 6213–6230 (2017)
52. Rathore, H., et al.: A review of security challenges, attacks and resolutions for wireless medical devices. In: IWCMC (2017)
53. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. RFC 8446 (2018)
54. Restuccia, G., et al.: Low-power IoT communication security: on the performance of DTLS and TLS 1.3. In: PEMWN (2020)
55. Roy, A., et al.: Device-specific security challenges and solution in IoT edge computing: a review. J. Supercomput. **79**, 20790–20825 (2023). https://doi.org/10.1007/s11227-023-05450-6
56. Santesson, S., et al.: Transport layer security (TLS) cached information extension. In: RFC 7924 (2016)
57. Sanwald, S., et al.: Secure Boot Revisited: Challenges for Secure Implementations in the Automotive Domain. Embedded Security in Cars, Escar Europe (2020)
58. Schläpfer, T., et al.: Security on IoT devices with secure elements. In: Embedded World Conference (2019)
59. Serror, M., et al.: Challenges and opportunities in securing the industrial internet of things. IEEE Trans. Ind. Inf. **17**(5), 2985–2996 (2021)
60. Sheffer, Y., et al.: Recommendations for secure use of transport layer security (TLS) and datagram transport layer security (DTLS). In: RFC 9325 (2022)
61. Somasundaram, R., Thirugnanam, M.: Review of security challenges in healthcare internet of things. Wireless Netw. **27**(8), 5503–5509 (2020). https://doi.org/10.1007/s11276-020-02340-0
62. Ta-Shma, P., et al.: An ingestion and analytics architecture for IoT applied to smart city use cases. IEEE Internet Things J. **5**(2), 765–774 (2018)
63. Tayyab, M., et al.: A survey on handover management: from LTE to NR. IEEE Access **7**, 118907–118930 (2019)
64. The Things Industries B.V.: The Things Network. https://www.thethingsnetwork.org/
65. Thomassen, M.S.E., et al.: A study of extensive LoRaWAN downlink communication in a mobility scenario. In: MobiQuitous (2022)
66. Touqeer, H., et al.: Smart home security: challenges, issues and solutions at different IoT layers. J. Supercomput. **77**(12) 14053–14089 (2021). https://doi.org/10.1007/s11227-021-03825-1
67. Tuptuk, N., et al.: A systematic review of the state of cyber-security in water systems. Water **13**(1), 81 (2021)
68. van der Velde, D., et al.: Methods for actors in the electric power system to prevent, detect and react to ICT attacks and failures. In: IEEE ENERGYCon (2020)
69. Wagner, E., et al.: Take a bite of the reality sandwich: revisiting the security of progressive message authentication codes. In: ACM WiSec (2022)
70. Wagner, E., et al.: Retrofitting integrity protection into unused header fields of legacy industrial protocols. In: IEEE LCN (2023)
71. Wagner, E., et al.: BP-MAC: Fast authentication for short messages. In: ACM WiSec (2022)

72. Wagner, E., et al.: When and how to aggregate message authentication codes on lossy channels? In: ACNS (2024)
73. Wolsing, K., et al.: IPAL: Breaking up silos of protocol-dependent and domain-specific industrial intrusion detection systems. In: RAID (2022)
74. Zhang, K., et al.: Security and privacy in smart city applications: challenges and solutions. IEEE Commun. Mag. **55**(1), 122–129 (2017)
75. Zhao, J., et al.: SecureSIM: rethinking authentication and access control for SIM/ESIM. In: MobiCom (2021)

Activity Recognition

Cross-User Activity Recognition via Temporal Relation Optimal Transport

Xiaozhou Ye(✉) and Kevin I-Kai Wang

Department of Electrical, Computer, and Software Engineering,
The University of Auckland, Auckland, New Zealand
xye685@aucklanduni.ac.nz, kevin.wang@auckland.ac.nz

Abstract. Current research on human activity recognition (HAR) mainly assumes that training and testing data are drawn from the same distribution to achieve a generalised model, which means all the data are considered to be independent and identically distributed (*i.i.d.*). In many real-world applications, this assumption does not hold, and collected training and target testing datasets have non-uniform distribution, such as in the case of cross-user HAR. Domain adaptation is a promising approach for cross-user HAR tasks. Existing domain adaptation works based on the assumption that samples in each domain are *i.i.d.* and do not consider the knowledge of temporal relation hidden in time series data for aligning data distribution. This strong assumption of *i.i.d.* may not be suitable for time series-related domain adaptation methods because the samples formed by time series segmentation and feature extraction techniques are only coarse approximations to *i.i.d.* assumption in each domain. In this paper, we propose the temporal relation optimal transport (TROT) method to utilise temporal relation and relax the *i.i.d.* assumption for the samples in each domain for accurate and efficient knowledge transfer. We obtain the temporal relation representation and implement temporal relation alignment of activities via the Hidden Markov model (HMM) and optimal transport (OT) techniques. Besides, a new regularisation term that preserves temporal relation order information for an improved optimal transport mapping is proposed to enhance the domain adaptation performance. Comprehensive experiments are conducted on three public activity recognition datasets (i.e. OPPT, PAMAP2 and DSADS), demonstrating that TROT outperforms other state-of-the-art methods.

Keywords: Human activity recognition · Out-of-distribution · Domain adaptation · Transfer learning · Time series classification

1 Introduction

Human activity recognition (HAR) is a crucial research area in Human-Computer Interaction, ubiquitous computing [3] and the Internet of Things [1], which aims to correctly classify the current activity of a human or a group of

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 355–374, 2024.
https://doi.org/10.1007/978-3-031-63989-0_18

humans performs based on physical sensor observations and context information [18]. HAR is applied to a variety of applications, including medical treatment, ambient assisted living, fitness and sports, rehabilitation, security surveillance, health monitoring, automatic security and home automation. Existing state-of-the-art HAR methods for processing time series sensor data are under the assumption that training and testing data are drawn from the same distribution, which means all the data are independent and identically distributed *i.i.d.* [38]. This assumption implies that a generalised model is achieved across the source training data and the target testing data. In other words, the training and testing data are taken from the same domain where the feature space and data distribution characteristics are the same [23]. However, in many real-world applications, this assumption does not hold, and the collected training and target testing datasets have non-uniform distribution due to the issue of data heterogeneity or sometimes referred to as the out-of-distribution (*o.o.d.*) problem. The performance of a model that is trained on the source domain will likely decline when tested on the target domain [24]. Therefore, it is necessary to consider a more practical scenario where the data distributions of the training and testing set can be different.

Transfer learning (or more specifically domain adaptation) [23] is a promising method for handling data heterogeneity. In general, the principle and key idea of transfer learning is to find some common knowledge and reduce the data distribution differences between the source domain and target domain. Most of the current domain adaptation research focus on static data area such as computer vision with the assumption that each sample(image) is *i.i.d.* in its corresponding domain [24]. The same assumption is also applied to time series transfer learning research, such as sensor-based HAR, which means each sliding window is *i.i.d.* in its corresponding domain [8,11,30,40]. However, time series data typically have inherent temporal relation, which means each sliding window is not independent of each other [13]. For an intuitive example of activity recognition, walking is made up of three sub-activities: raising the leg, thrusting forward and feet to the ground. These three sub-activities have temporal dependency relations. So, existing domain adaptation approaches do not fully utilize the characteristics of time series data effectively because temporal relation knowledge is overlooked in the process of aligning data distribution.

In this paper, we propose a novel temporal relation optimal transport (TROT) method targeting time series domain adaptation for cross-user activity recognition, which considers temporal relations across time. This method captures the data distributions of the common sub-activities while preserving the same temporal relation between the sub-activities across source and target users. Temporal relation preservation acts like a regularization for reducing the data distribution differences of the sub-activities across users. In this case, the data distribution mapping process can achieve better domain adaptation performance. Specifically, it is a combination of optimal transport and the Hidden Markov Model (HMM) method that performs cross-user HAR considering temporal relation knowledge. Moreover, we propose a new regularizer for the optimal transport domain adaptation problem that preserves temporal relation

order information to ensure correct data distribution alignment. Comprehensive experiments on three public activity recognition datasets demonstrate the better performance of TROT in time series domain adaptation applications (e.g. cross-user HAR). The rest of the paper is organized as follows: Sect. 2 presents the related work, delving into cross-user HAR and the concepts of transfer learning and domain adaptation. Section 3 describes the proposed method, including problem formulation, the motivation behind it, and the specifics of the TROT method-focusing on temporal relation representation and OT-based data distribution alignment. In Sect. 4, we detail the experimental setup and provide a comparison between our method and existing methods. Finally, Sect. 5 offers conclusions drawn from the study and suggests potential avenues for future research.

2 Related Work

2.1 Cross-User Human Activity Recognition

HAR is one crucial part of ubiquitous computing because of its important role in supporting daily human life. It aims at recognizing and analysing human behaviours from learned high-level knowledge from multi-modal sensor observations and context information. Various types of sensors are utilized to implement activity recognition in different scenarios. From the sensor modality view, HAR can be classified into five types: Smartphones/wearable sensors-based HAR, ambient sensors-based HAR, device-free sensors-based HAR, vision-based sensors HAR and other modality sensors-based HAR [7,19,35]. In this paper, we focus on wearable sensors-based HAR.

Sensor-based HAR is considered a problem of time series classification in machine learning research [3]. A great number of classification models, such as ensemble learning [29], SVM [5], and HMM [2] are proposed to solve the HAR problem. With the recent research development, deep learning has achieved many state-of-the-art results for a wide variety of tasks. Deep learning-based HAR approaches [12] are able to learn high-level features and automatically extract features from massive amounts of data [33]. However, these approaches mostly depend on the assumption that training and testing data are drawn from the same distribution to ensure the generalization ability of the model, which means all the data are *i.i.d.* [38]. In many real-world applications, this assumption does not hold, and the collected training and testing datasets are *o.o.d.*. In this paper, we focus on the sensor-based HAR *o.o.d.* problem.

There are various categories of sensor-based HAR *o.o.d.*: First, data is generated from different sensors. Different sensor types, platforms, manufacturers and modalities may cause different data formats and distributions [39]. Second, the data pattern may change over time, which is also called concept drift [21]. For example, the walking pattern of the same person may be impacted by health status. Third, the behaviour differentiation between different people may also be significant [28]. For instance, different people may walk at different paces. Forth, physical sensors may be installed in different body positions [27] or environmental layouts in smart homes [31], which may lead to different data distribution.

Our current research focuses on the sensor-based HAR *o.o.d.* of behaviour differentiation between different people.

2.2 Transfer Learning and Domain Adaptation

Transfer learning is a process that enables a model to be trained in one or more source domains and applied to one or more related target domains that have no label or few labels. During the process, one of the key tasks is to solve the *o.o.d.* problem by reducing the distribution difference between source and target domains. Domain adaptation [24,38] is a special type of transfer learning. It still focuses on solving *o.o.d.* problem between source and target domains but adds an additional constraint of assuming the same task of source and target domains. This paper focuses on the unsupervised domain adaptation problem, which means the availability of labelled data from the source domain and unlabelled data from the target domain [38].

Domain adaptation has been developed for many years, and most of the approaches are feature-based transfer learning [23]. The Large Margin Nearest Neighbour (LMNN) approach, as presented in [37], utilizes Mahalanobis distance metric learning. The goal of this method is to learn a linear transformation, such that data from different classes are distinctly separated from each other within the feature space. Subspace Alignment(SA) [14] applied PCA to generate subspace for the source domain and target domain. Then, SA is achieved by minimizing Bregman matrix divergence to learn a linear transformation matrix. CORrelation Alignment(CORAL) [32] learns a linear transformation by minimizing the distance between the second-order statistics of the source domain and target domain features. Optimal transport(OT) theory is a promising approach and is also applied to solve the domain adaptation problem [9] via Sinkhorn distances. There are several works extending the research of optimal transport domain adaptation. Optimal Transport for Domain Adaptation(OTDA) [15] is proposed to learn the coupling between two probability density functions. The source domain is then transformed into the target domain via barycentric mapping. Substructural optimal transport(SOT) [22] explores the substructure of domains to complete substructure-level mapping to achieve a balance between coarse-grained mapping and fine-grained mapping.

These feature-based domain adaptation methods focus on static data such as images, whereas time series data is often treated directly by the same domain adaptation framework [20,22]. However, these existing domain adaptation approaches may not work well with time series data because the temporal relation knowledge from the time series data is overlooked in the process of aligning data distribution. Temporal relation is an essential characteristic of time series data. Existing approaches that only make use of temporal invariant knowledge lead to models with unreliable performance. This is especially the case in real-world cross-user HAR applications, where a model is trained by one (or more) user(s) and later applied to other user(s). Furthermore, temporal

relation knowledge may be useful for finding the common knowledge between users for better sensor-based HAR domain adaptation. The focus of this paper is the exploration of temporal relation knowledge.

3 Method

3.1 Problem Formulation

In a cross-user HAR problem, a labelled source user $S^{train} = \left\{\left(x_i^{train}, y_i^{train}\right)\right\}_{i=1}^{n^{train}}$ drawn from a joint probability distribution $P^{Source}(X, y)$ and a target user $S^{test} = \left\{\left(x_i^{test}, y_i^{test}\right)\right\}_{i=1}^{n^{test}}$ drawn from a joint probability distribution $P^{Target}(X, y)$, where n^{train} and n^{test} are the number of source and target samples respectively. S^{train} and S^{test} have the same feature space and label space. In real-world applications, a source user and a target user are highly likely to have different data distributions, i.e., $P^{Source}(X, y) \neq P^{Target}(X, y)$. Given source user data $\left\{\left(x_i^{train}, y_i^{train}\right)\right\}_{i=1}^{n^{train}}$ and target user data $\left\{\left(x_i^{test}\right)\right\}_{i=1}^{n^{test}}$, the goal is to achieve correct activity classification for unlabeled target user data.

3.2 Motivation

In general, the principle and key idea of domain adaptation are to reduce the difference between the source domain(s) and the target domain(s) and bridge the gap in the data distributions between the domains. Existing cross-user HAR domain adaptation approaches are mainly based on sample level, user level, activity level and sub-activity level as shown in Fig. 1. However, these methods ignore the temporal relation across chronological time series samples and treat samples as *i.i.d.* in different domains. In fact, temporal relation knowledge is an important property hidden in time series data that can be used as a bridge for cross-user domain adaptation. Different users' time series activity data follow the same temporal relation in various activities. For example, the walking activity includes three sub-activities, namely raising the leg, thrusting forward and feet to the ground, regardless of different users. These temporal relations are valuable knowledge that can better support data distribution alignment across different users. Therefore, we propose a new method called TROT that introduces temporal relation knowledge for solving domain adaptation.

3.3 Temporal Relation Optimal Transport

In this section, we propose an OT implementation that utilizes temporal relation knowledge called TROT. In TROT, HMM is utilized to capture temporal relation knowledge, and OT is used to align the data distributions between the source user and the target user. In addition, we propose to add a new regularizer which preserves temporal order information to enhance OT optimization for better domain adaptation performance. TROT method mainly includes four

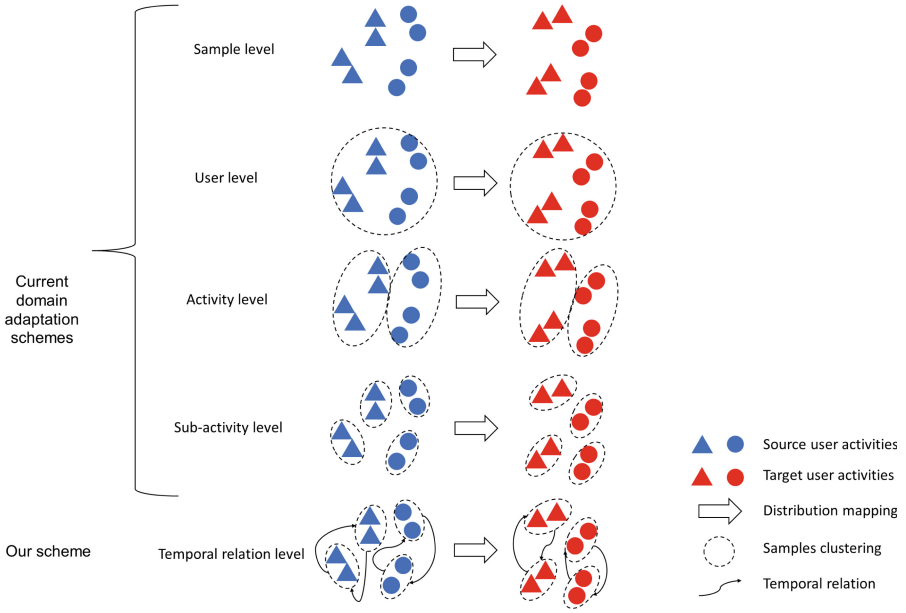


Fig. 1. Comparison of different domain adaptation schemes in cross-user HAR.

steps as listed below, and detailed explanations and discussions are provided in the following subsections. Figure 2 shows the overall TROT learning process.

1. **Data pre-processing**: Data segmentation and feature extraction for both source and target users is implemented as in Fig. 2 from figure (a) to figure (b). In Fig. 2, the red colour is associated with the source user, while the blue colour is associated with the target user. The different shapes, such as circle and triangle, represents different activity classes. Sensor-based HAR data consists of observations taken sequentially over time. When working with time series data, the sliding window technique is often employed for data segmentation. Then, for each window of data, the time and frequency domain features are extracted. Time domain features capture raw temporal patterns in the data, while frequency domain features provide insight into the rhythmic or periodic components of the data. These features are crucial in cross-user HAR because they can capture general patterns of human movement that are less dependent on individual idiosyncrasies, thereby improving the model's ability to generalize across different users. In this step, the raw data are transformed into samples with features.
2. **Temporal relation extraction**: Temporal state information is extracted via HMM for source and target users as in Fig. 2 from figure (b) to figure (c). The number in each shape indicates the corresponding temporal state information hidden in the time series data. For example, the numbers one, two and three in triangles may correspond to raising the leg, thrusting forward

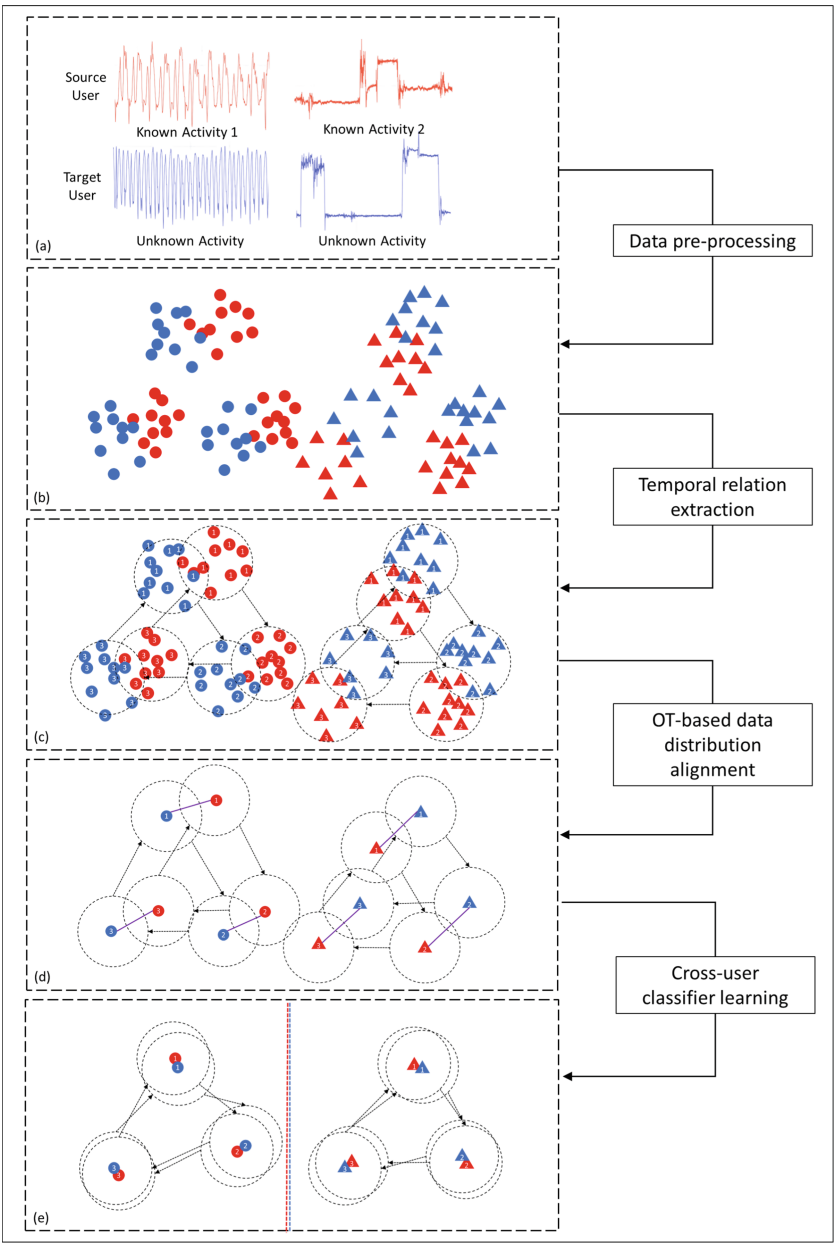


Fig. 2. TROT Learning Process Figure.

and feet to the ground of walking activity. The black dash arrow between the numbers means the temporal relation. The learned temporal relation knowledge is the common knowledge across users that is useful to bridge the gap between source and target users for better domain adaptation. In this step, the samples are clustered and form several clusters based on their temporal relation information.

3. **OT-based data distribution alignment**: The data distribution alignment task is achieved by transforming the samples from source user to target user as in Fig. 2 from figure (c) to figure (d). The cluster centres are used to represent the corresponding cluster, and then the centres of the source user are projected to the centres of the target user considering the temporal order information for learning the mapping plan. The purple solid line in Fig. 2 means the data distribution transformation from the source user to the target user. In this step, the samples from the source user are transformed based on the learned mapping plan.
4. **Cross-user classifier learning**: After the above data distribution alignment step between source and target users, the decision boundary of activity classes of the source and target users should be relatively consistent. Therefore, the transformed source user samples with labels are utilized for training a classifier for recognizing target user samples as in Fig. 2 from figure (d) to figure (e). The red dashed line is the decision boundary of activity classes of the source user, while the blue dashed line is the decision boundary of the target user. In this step, the learned classifier from the transformed source user is also suitable for the target user.

The highlight of the TROT method is the temporal relation extraction step and the OT-based data distribution alignment step. Discussion is focused on these two steps.

3.3.1 Temporal Relation Extraction

As in Fig. 2, after the data pre-processing step, there is implicit temporal relation information that needs to be captured. Here, we consider HMM [25] learning for temporal relation representation. HMM is the directed temporal probability graph model that describes the dependency of two related sequences. These two related sequences are called state sequence $S = (s_1, s_2, s_3, \ldots, s_T)$ and observation sequence $X = (x_1, x_2, x_3, \ldots, x_T)$. $Q = (q_1, q_2, q_3, \ldots, q_N)$ is the state set of all the possible states. The value of the state sequence at t time point s_t is only related to the value of the state sequence of $t-1$ time point s_{t-1}, and the value of the observation sequence at t time point x_t is only related to the value of the state sequence at t time point s_t. Here, there are three types of probability that HMM needs to learn: 1) $p(S_1)$ initial state probability, 2) $p(S_t|S_{t-1})$ state transition probability and 3) $p(X_t|S_t)$ emission probability.

For the cross-user HAR scenario, each sub-activity can be seen as a state. The projection of the observed samples and the corresponding sub-activity can be learned via HMM emission probability. The temporal relation between the

sub-activities can be learned via HMM state transition probability. In this way, the hidden temporal information can be captured and represented.

In detail, initial state probability is the possibility of the first sub-activity and is assumed to be the same across users in our study. Therefore, the first captured state is set as the initial set for the source user and the target user. Emission probability is the probability of a sample occurring, given a specified sub-activities (i.e. state) at time point t. The observation sequence, X_t, can be expressed as a function of S_t based on the learning of emission probability, which means $X_t = f(S_t)$. For the emission probability, we assume it conforms to Gaussian distribution. The emission probability in each class can be expressed as $B = [b_k]_{N\times 1},\ b_k = \mathcal{N}(\mu_k,\ \sigma_k),\ X_k \sim b_k,\ k = 1, 2, ..., N$, N is the number of sub-activities in each class. Here, μ_k means the centre value of k^{th} state's Gaussian distribution, σ_k means the covariance value of k^{th} state's Gaussian distribution, and X_k means the samples belong to the k^{th} state. Assuming there are C activity classes, then there are a total of $N \times C$ Gaussian distributions need to be learned for each user, i.e., $B^{Source} = [\mathbf{b}^{Source}]_{N\times C}$ and $B^{Target} = [\mathbf{b}^{Target}]_{N\times C}$. In this way, we abstract the observation samples to a higher level of sub-activities, and then the temporal relation extraction happens at the sub-activities level.

The state transition probability is the probability of a sub-activities at time point $t-1$ change to another sub-activity at time point t. For the state transition probability, we reduce the difference between the source user and the target user via common temporal relation knowledge of the human activity. The common knowledge is that the state transition follows the sub-activities order because of the continuity of human activities, which means the current state can only change to the following state in a future time point. For example, walking activity should first raise the leg and then thrust forward in the next state, but not jump to the last state of feet to the ground step. Therefore, $p(S_t|S_{t-1})$ state transition probability can be expressed as a $N \times N$ matrix $A = [a_{i,j}]_{N\times N},\ a_{i,j} = P(S_t = q_j|S_{t-1} = q_i),\ i, j = 0, 1, 2, ..., N, i \neq j$. $a_{i,j}$ means the transition probability from state q_i in $t-1$ time point to state q_j in t time point. There are total N sub-activities. q_0 is the initial state and q_N is the ending state. If an activity such as running or going upstairs has periodic cycle characteristics, then $q_0 = q_N$. The form of the state transition matrix for HMM is given below. Here, we set $\forall i \quad a_{i,i+1} = 1$ to make the state transition probability a common temporal relation knowledge in both the source and target users.

$$\begin{bmatrix} a_{0,1} & 0 & \cdots & \cdots & 0 & 0 \\ 0 & a_{1,2} & 0 & \cdots\cdots & & 0 \\ 0 & 0 & a_{2,3} & 0 & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots\cdots & & 0 \\ 0 & \cdots & \cdots & 0 & 0 & a_{N-1,N} \end{bmatrix}$$

With the observation sequence X, our goal is to calculate the parameters of emission probability in each activity for each user. The process of parameter learning can be achieved by Expectation Maximum (EM) [10] algorithm. After

the HMM parameter learning, the source user data distribution can be expressed as B^{Source}, and the target user can be expressed as B^{Target}. As Fig. 2 (c) shows, now each class is represented by multiple temporally-related Gaussian distributions. Our next task is the alignment of sub-activities in each class between source and target users.

3.3.2 Optimal Transport Based Data Distribution Alignment

Once the temporal-related distribution is extracted by HMM, the data distribution alignment is achieved via an OT-based method that maps the Gaussian distributions between the source and target users. OT has drawn attentions in recent years for domain adaptation, and it aims at moving source domain data distribution on top of target domain data distribution with an optimization transportation plan [17].

Based on the OT-based method, the source user and target user can be expressed as data distributions respectively:

$$P^{Source} = \sum_{i=1}^{i=k^{Source}} w_i^{Source} \delta_{\mu_i^{Source}},$$

$$P^{Target} = \sum_{j=1}^{j=k^{Target}} w_j^{Target} \delta_{\mu_j^{Target}}.$$

Here, $\sum_{i=1}^{i=k^{Source}} w_i^{Source} = 1$, and $\sum_{j=1}^{j=k^{Target}} w_j^{Target} = 1$. $\boldsymbol{\mu}$ is the center value of Gaussian distributions in B learned from HMM, and $\boldsymbol{\delta_\mu}$ is the Dirac function corresponding to the location of $\boldsymbol{\mu}$. The probabilities $\boldsymbol{w}$ are the probability masses associated with $\boldsymbol{\mu}$. In our case, $k^{Source} = k^{Target} = C \times N$, where C is the number of activity classes, and N is the number of states. We set $\forall i, j\ w_i^{Source} = w_j^{Target} = \frac{1}{C \times N}$.

Similar to [15], we also apply regularization approach to OT to prevent overfitting and keep structural risk minimization. Entropy regularization [9] $H(\gamma)$ is introduced with the idea of lowering the sparsity of coupling matrix γ via increasing its entropy to find a smoother optimal transport plan. Here, the coupling matrix represents the transport plan between two probability distributions as source user and target user. The smoother optimal transport plan often translates to being more generalizable, which is beneficial for domain adaptation, instead of having very specific, point-to-point transport, which might overfit the source user. Group-sparse regularization [15] $\Omega(\gamma)$ considers the use of label distribution knowledge that makes each sample in the target domain receive masses only from the samples in the source domain with the same label. By preserving the inherent label structure of the data, group-sparse regularization promotes the correct transfer of class structures between domains and makes models more generalizable to the target user.

Besides, we propose to add a new regularization term to the regularized optimal transport that preserves temporal order information. The main intuition underlying this temporal order preservation regularization is that even two

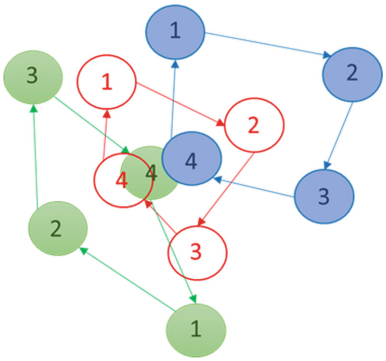


Fig. 3. OT-based Temporal Order Preservation Regularization Figure.

Gaussian distribution centres may be closer between the source user and target user; however, different temporal relation order may lead to wrong data distribution alignment. As Fig. 3 shows, the red is an activity from the source user, and the blue and green are two different activities from the target user. In this example, both green and blue are very close to red in terms of the overall distance between the centres of sub-activities distributions. However, blue is the only suitable one after considering the temporal order of the state Gaussian distribution.

Table 1. Temporal Order Mapping Between Source and Target Users Table.

		1	2	3	4	5	6	7	8
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
1	**1**	✓				✓			
2	**2**		✓				✓		
3	**3**			✓				✓	
4	**4**				✓				✓
5	**1**	✓				✓			
6	**2**		✓				✓		
7	**3**			✓				✓	
8	**4**				✓				✓

This temporal order preservation regularizer is defined as:

$$T(\gamma) = \sum_i \left\| \gamma(i, I_i) \right\|_2$$

The coupling matrix γ has the indices of the column, each of which corresponds to a Gaussian distribution from the target user, while the indices of the row correspond to a Gaussian distribution from the source user. For example, in Table 1, the indices of the coupling matrix are from 1 to 8 for both the column and the row. I_i is a set of indices that tell which columns in γ correspond to the

same temporal order as the i^{th} Gaussian distribution of the source user. In other words, I_i helps to find the matching temporal order in the target user for each Gaussian distribution in the source user. As in Table 1, the red and yellow are two activities from the source user, while the blue and green are two activities from the target user. All the activities have four temporal states (i.e. Gaussian distributions). The number from 1 to 4 in each activity represents the temporal order. For example, the temporal order 1 in red activity of the source user matches the temporal order 1 in blue and green activities of the target user, as the checkmark symbol shows. $\gamma(i, I_i)$ is a vector that is achieved by picking out the coefficients from the i^{th} row of γ that correspond to these matched temporal orders from the column. These coefficients are the transport plan from the i^{th} Gaussian distribution of the source user to the matched Gaussian distributions in the target user.

In summary, this regularizer encourages the Gaussian distributions of the source user to match the Gaussian distributions with the same temporal order of the target user. For example, in Fig. 3, order 1 of red is encouraged to map to order 1 of blue instead of mapping to order 3 of green.

With this temporal order preservation regularization, the masses corresponding to each Gaussian distribution of specified temporal order value match samples of the source and target users with the same temporal order value. So, the overall goal of TROT is:

$$\gamma^* = \arg\min_{\gamma} \langle \gamma, \mathcal{C} \rangle_F + \lambda H(\gamma) + \eta \Omega(\gamma) + \tau T(\gamma)$$

$$\text{s.t.}$$

$$\gamma \mathbf{1}_{k^{\text{Target}}} = P^{\text{Source}}$$

$$\gamma^T \mathbf{1}_{k^{\text{Source}}} = P^{\text{Target}}$$

$\gamma \in \mathbb{R}^{k^{\text{Source}} \times k^{\text{Target}}}$ is the coupling matrix between the probability distributions of the source domain and target domain, $\mathbf{1}_k$ is k-dimensional vectors of ones, $\langle ., . \rangle_F$ is the Frobenius dot product. $\mathcal{C}$ is the cost matrix of the squared Euclidean distance between two locations, $\mathcal{C}(i, j) = \left\| \mu_i^{Source} - \mu_j^{Target} \right\|_2^2$, denotes the cost of pushing forward a probability mass from μ_i^{Source} to μ_j^{Target}. λ, η, τ are coefficients.

The generalized conditional gradient (GCG) approach is used to solve the regularized optimization problem [15]. GCG algorithm addresses the general case of constrained minimization of composite functions defined as

$$\min_{\gamma} f(\gamma) + g(\gamma)$$

$f(\gamma)$ is a differentiable and possibly non-convex function, $g(\gamma)$ is a convex, possibly non-differentiable function. Here, we set

$$f(\gamma) = \langle \gamma, \mathcal{C} \rangle_F + \lambda H(\gamma)$$
$$g(\gamma) = \eta \Omega(\gamma) + \tau T(\gamma)$$
$$\text{s.t.} \quad H'(\gamma)$$

Now, the TROT task is

$$\gamma^* = \arg\min_{\gamma} \langle \gamma, \mathcal{C} + \lambda \nabla H(\gamma^{iter}) \rangle_F + (\eta \Omega(\gamma) + \tau T(\gamma))$$

iter is the number of iterations. After getting the optimal coupling matrix γ^*, barycentric mapping of the source domain is implemented for mapping the source domain to the target domain as

$$\widehat{P^{Target}} = diag(\gamma^* \mathbf{1}_{k^{Target}})^{-1} \gamma^* P^{Target}$$

In the final step, transformed source user data is utilized to train a classifier for classifying target user samples. Here, any classifier can be applied to learn a model with $\widehat{P^{Target}}$ and the labels of the source domain y^{Source}. We use the 1-Nearest Neighbour classification method to get labels $\{y_j^{Target}\}_{j=1}^{n^{Target}}$. 1-Nearest Neighbour is a simple instance of the k-Nearest Neighbours algorithm [16], which is a type of instance-based learning. Given a sample from the target user, the algorithm searches the entire samples from the source user to find the single sample that is closest to the sample from the target user. Here, the distance metric is the Euclidean metric. The label of the nearest sample from the source user is then assigned to the sample from the target user.

4 Experiments

4.1 Datasets and Pre-processing

We used three common sensor-based HAR public datasets as Table 2 shows for validating our cross-user transfer learning method. Here, we only use the sensor values of the accelerometer and gyroscope in the position of the right lower arm to explore the practical scenario of using the smartwatch. In the following, the information of each dataset is introduced briefly.

OPPORTUNITY (OPPT) [6] dataset contains recordings of subjects in a daily living scenario performing morning activities. During the recordings, subjects perform the activities without any restriction by following a loose description of the overall actions to perform. The sampling frequency is 30 Hz. Physical activity monitoring (PAMAP2) dataset [26] is collected following a protocol of specified activities for each subject. Over 10 h of data were collected altogether. The sampling frequency is 100 Hz. Daily and Sports Activities Data Set (DSADS) [4] is collected by asking the subjects to perform the activities in their own fashion. Without any given instructions, this is likely to lead to greater inter-subject variations, and would be more similar to real-life situations. Each activity

Table 2. Three sensor-based HAR datasets information Table.

Dataset & Ref.	Subjects	#Activities	Common Activities
OPPT	S1, S2, S3	4	lying, sitting, standing, walking
PAMAP2	1, 5, 6	11	lying, sitting, standing, walking, running, cycling, Nordic walking, ascending stairs, descending stairs, vacuum cleaning, ironing
DSADS	2, 4, 7	19	sitting, standing, lying on back, lying on right, ascending stairs, descending stairs, standing in an elevator still, moving around in an elevator, walking in a parking lot, walking on a treadmill in flat, walking on a treadmill inclined positions, running on a treadmill in flat, exercising on a stepper, exercising on a cross trainer, cycling on an exercise bike in horizontal positions, cycling on an exercise bike in vertical positions, rowing, jumping, playing basketball

is performed by each subject for 5 min. Sensor units are calibrated to acquire data at 25 Hz sampling frequency. The three datasets are selected in order of increasing domain adaptation task difficulty (DSADS 19 activities > PAMAP2 11 activities> OPPT 4 activities). In this case, DSADS dataset contains more activity classes and more similar activities (3 different walking activities), which make it harder to distinguish.

The data pre-processing step for all the methods follows the same setup as the TROT data segmentation and feature extraction step. Time series data is continuous sequence data. The sliding window technique is a common data segmentation method. For the purpose of temporal relation extraction, each window is set as a fixed time interval. Each window is set as 3 s, and overlapping is 50% as the common setting in sensor-based HAR task [34]. Following a similar feature extraction setup as [36], the x, y, z axis data in each sensor of accelerometer and gyroscope are combined as $\sqrt{x^2 + y^2 + z^2}$, 19 features from both time and frequency domains are extracted for a single sensor as in Table 3. Therefore, in each window, there are total of 38 features because of two sensors of accelerometer and gyroscope. Besides, standardization of maximum absolute value scaler is applied to scales and translates each feature individually in the range of $[-1, 1]$ to prevent the side effect of the features with different numerical order.

4.2 Experiments and Methods Comparison

Here are the categories of methods for comparison, including no domain adaptation methods and domain adaptation methods.

Table 3. Features Extraction Per Sensor

ID	Feature	Description
1	Mean	Average value of samples in window
2	Var	Variance
3	STD	Standard deviation
4	Mode	The value with the largest frequency
5	Maximum	Maximum
6	Minimum	Minimum
7	Mean crossing rate	Rate of times signal crossing mean value
8	Range	Maximum minus minimum
9	DC	Direct component
10–14	Five amplitude features	Mean, variance, STD, skewness, kurtosis
15–19	FFT variance	Mean, variance, STD, skewness, kurtosis

1. No domain adaptation:
   (a) NA: No Adaptation
   (b) TD: Target domain 1NN
2. Domain adaptation:
   (a) The usage of temporal relation: TROT
   (b) No usage of temporal relation:
      i. LMNN: Large Margin Nearest Neighbour [37]
      ii. SA: Subspace Alignment [14]
      iii. OT: Optimal transport [9]
      iv. CORAL: CORrelation Alignment [32]
      v. OTDA: Optimal transport for domain adaptation [15]
      vi. SOT: Substructural optimal transport [22]

The tables provide a comprehensive comparison of different methods applied to the OPPT, PAMAP2, and DSADS datasets. The performance of each method is evaluated based on the transition from one user to another user. Three users are randomly selected from all the users in each dataset. Then, a one-to-one cross-user HAR task is implemented, encompassing all possible pairings between users in each dataset.

For the NA method, the 1-Nearest Neighbour model is trained by source user samples and is applied directly to target user samples without any domain adaptation mapping. For the TD method, the 1-Nearest Neighbour model is trained and predicted by target user samples, which is the optimal baseline that we aim to approach. For the domain adaptation methods, hyper-parameters tuning is performed, and we follow a similar process as suggested in [15] to prevent overfitting on the testing set. The target user is partitioned into two equal datasets the validation set and the test set. For the TROT method that captures the temporal relation, it is particularly crucial to assess its performance on unseen future time-series data that exhibit a distinction from the data utilized for aligning the

distribution. Therefore, the target domain is partitioned in equal time duration of two parts that keep temporal order. The validation set is used to obtain the combination of hyper-parameters that achieve the highest accuracy from the hyper-parameter space. Then, the performance of the testing set is evaluated with the selected combination of hyper-parameters. Classification accuracy on the target domain is the evaluation metric that we used.

Table 4. OPPT Dataset Methods Comparison Table

Method	S1 → S2	S1 → S3	S2 → S1	S2 → S3	S3 → S1	S3 → S2
NA	44.18	54.11	50.36	53.22	47.94	46.53
TD	100	100	100	100	100	100
LMNN	77.77	73.10	75.04	69.94	65.96	71.06
SA	79.50	84.11	83.39	83.22	78.18	76.53
OT	80.37	81.33	81.69	79.05	81.69	78.89
CORAL	74.18	84.74	80.36	84.74	78.67	75.92
OTDA	83.96	80.44	86.42	78.04	86.66	80.62
SOT	75.60	79.94	82.76	73.62	74.09	69.53
TROT	**100**	**100**	**100**	**100**	**100**	**100**

In general, all the domain adaptation methods can achieve increased performance than the baseline method of NA in the OPPT dataset as shown in Table 4. TROT method achieved the highest accuracy across all the six tasks, with a perfect score of 100%. This suggests that the method is extremely efficient and reliable for the OPPT dataset. OTDA also showed promising results, especially in the tasks from S2 to S1 and from S3 to S1. SA and OT methods were relatively consistent in their performance across different tasks. The SA method, in particular, performed quite well, especially in the task from S1 to S3 and S2 to S1.

When applied to the PAMAP2 dataset in Table 5, all the domain adaptation methods except for LMNN achieved better performance than NA in most of the cross-user tasks. Negative transfer learning (i.e., worse performance) happened on LMNN for all the tasks of the PAMAP2 dataset. TROT maintained the best performance compared to other methods. Moreover, SOT and OTDA showed notable performance, with SOT showing particularly high efficiency in the tasks from 1 to 5 and 5 to 1. OTDA showed a good performance in the tasks from 1 to 6 and from 6 to 1.

On the DSADS dataset in Table 6, CORAL, SOT, OTDA and TROT achieved higher performance than NA in more than half of the tasks in the DSADS dataset. TROT maintained its superior performance. Among other methods, TROT performed best, especially in the task from 2 to 4. SOT showed reasonable performance, especially in the tasks from 2 to 7 and from 4 to 2.

Table 5. PAMAP2 Dataset Methods Comparison Table

Method	$1 \rightarrow 5$	$1 \rightarrow 6$	$5 \rightarrow 1$	$5 \rightarrow 6$	$6 \rightarrow 1$	$6 \rightarrow 5$
NA	28.72	42.02	47.91	55.46	60.43	35.75
TD	100	100	100	100	100	100
LMNN	22.25	20.17	26.04	26.41	36.03	21.34
SA	31.33	43.58	49.81	58.82	60.18	40.07
OT	53.69	61.70	57.77	61.22	57.14	57.32
CORAL	32.69	42.62	50.95	57.14	59.04	38.25
OTDA	64.47	65.23	63.08	65.19	66.12	64.47
SOT	69.82	57.80	64.54	61.95	62.48	56.14
TROT	**84.80**	**75.67**	**78.28**	**83.05**	**82.46**	**73.76**

Table 6. DSADS Dataset Methods Comparison Table

Method	$2 \rightarrow 4$	$2 \rightarrow 7$	$4 \rightarrow 2$	$4 \rightarrow 7$	$7 \rightarrow 2$	$7 \rightarrow 4$
NA	48.57	41.12	39.76	38.61	44.24	37.88
TD	100	100	100	100	100	100
LMNN	18.45	16.47	13.13	15.63	11.78	17.67
SA	48.83	40.91	39.86	38.30	43.04	38.82
OT	47.26	35.64	46.59	38.09	36.89	36.63
CORAL	49.35	41.27	40.70	39.24	43.30	39.76
OTDA	48.10	38.25	45.23	45.28	35.28	43.30
SOT	43.23	46.72	48.47	45.64	43.00	41.11
TROT	**53.95**	**52.47**	**49.27**	**47.21**	**53.68**	**46.37**

OTDA also performed relatively well but had a decrease in performance compared to the PAMAP2 dataset. LMNN had the lowest performance among all the methods, much like in the other two datasets. This suggests that LMNN might not be the best choice for cross-user HAR task.

TROT method gets the highest accuracy among all the three datasets of 18 cross-user HAR tasks, the performance of other methods varies based on the dataset and the specific tasks. The results showed that the methods without considering temporal relation knowledge have difficulties (i.e. unstable performance) when performing time series transfer learning tasks, because the design rationale assumes the samples follow *i.i.d.* in each domain. In contrast, TROT

relaxes the constraint condition of domain adaptation without the *i.i.d.* assumption of samples. Besides, TROT can capture temporal relation knowledge in each domain and reduce the difference by taking into account the temporal relation between the source and target users, which showed superior performance in the time series domain adaptation problem.

5 Conclusion

Current works in domain adaptation are mainly based on the assumption that samples in each domain are *i.i.d.* drawn from a joint probability distribution. However, existing research works overlook the temporal relation among the samples in the process of aligning data distribution and considering the temporal relation knowledge embedded in time series data. In this paper, we propose the TROT method to utilize temporal relation and relax the *i.i.d.* assumption for the source and target users and an OT and HMM based implementation for cross-user HAR. A new regularization term that preserves temporal order information for the regularized optimal transport mapping is proposed to extend the OT capability. Compared to existing domain adaptation, TROT obtains the temporal relation representation and completes the feature-based data distribution alignment to solve domain adaptation problem. Comprehensive experiments applied to three HAR public datasets demonstrate the superiority of the TROT method over other state-of-the-art methods.

In the future, we plan to remove the Markov assumption and extend HAR to capture and align more complex temporal relations in sensor-based HAR cross-user domain adaptation. Moreover, in the pursuit of refining and enhancing the applicability of the TROT, it is imperative to delineate the specific scenarios under which this approach may exhibit suboptimal performance. A critical aspect warranting investigation is the conceptualization and quantification of the "distance" between users or their corresponding data. This "distance" metric could potentially encapsulate the dissimilarity in types, positions, and other attributes of sensors utilized by different users. Establishing a threshold for this "distance" could provide a criterion to ascertain whether the application of TROT would yield satisfactory results. Such an exploration could illuminate the constraints of TROT, and guide the optimization of its algorithm to cater to a broader spectrum of user scenarios. Through a comprehensive analysis of these facets, the endeavor is to elevate the robustness and versatility of TROT, thus propelling it closer to being a universally applicable solution.

References

1. Abdallah, Z.S., Gaber, M.M., Srinivasan, B., Krishnaswamy, S.: Activity recognition with evolving data streams: a review. ACM Comput. Surv. (CSUR) **51**(4), 1–36 (2018)
2. Amft, O., Junker, H., Troster, G.: Detection of eating and drinking arm gestures using inertial body-worn sensors. In: Ninth IEEE International Symposium on Wearable Computers (ISWC 2005), pp. 160–163. IEEE (2005)

3. Andreas, B., Blanke, U., Schiele, B.: A tutorial on human activity recognition using body-worn inertial sensors. ACM Comput. Surv. (CSUR) **46**(3), 33 (2014)
4. Barshan, B., Yüksek, M.C.: Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. Comput. J. **57**(11), 1649–1667 (2014)
5. Bulling, A., Ward, J.A., Gellersen, H.: Multimodal recognition of reading activity in transit using body-worn sensors. ACM Trans. Appl. Percept. (TAP) **9**(1), 1–21 (2012)
6. Chavarriaga, R., et al.: The opportunity challenge: a benchmark database for on-body sensor-based activity recognition. Pattern Recogn. Lett. **34**(15), 2033–2042 (2013)
7. Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., Liu, Y.: Deep learning for sensor-based human activity recognition: overview, challenges, and opportunities. ACM Comput. Surv. (CSUR) **54**(4), 1–40 (2021)
8. Chen, Y., Wang, J., Huang, M., Yu, H.: Cross-position activity recognition with stratified transfer learning. Pervasive Mob. Comput. **57**, 1–13 (2019)
9. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Advances in Neural Information Processing Systems, vol. **26** (2013)
10. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol.) **39**(1), 1–22 (1977)
11. Deng, W.Y., Zheng, Q.H., Wang, Z.M.: Cross-person activity recognition using reduced kernel extreme learning machine. Neural Netw. **53**, 1–7 (2014)
12. Ding, R., et al.: Empirical study and improvement on deep transfer learning for human activity recognition. Sensors **19**(1), 57 (2018)
13. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. (CSUR) **45**(1), 1–34 (2012)
14. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2960–2967 (2013)
15. Flamary, R., Courty, N., Tuia, D., Rakotomamonjy, A.: Optimal transport for domain adaptation. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 1–40 (2016)
16. Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K.: KNN model-based approach in classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) OTM 2003. LNCS, vol. 2888, pp. 986–996. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39964-3_62
17. Kerdoncuff, T., Emonet, R., Sebban, M.: Metric learning in optimal transport for domain adaptation. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 2162–2168 (2021)
18. Lara, O.D., Labrador, M.A.: A survey on human activity recognition using wearable sensors. IEEE Commun. Surv. Tutorials **15**(3), 1192–1209 (2012)
19. Lentzas, A., Vrakas, D.: Non-intrusive human activity recognition and abnormal behavior detection on elderly people: a review. Artif. Intell. Rev. **53**(3), 1975–2021 (2020)
20. Li, S., Xie, B., Wu, J., Zhao, Y., Liu, C.H., Ding, Z.: Simultaneous semantic alignment network for heterogeneous domain adaptation. In: Proceedings of the 28th ACM International Conference on Multimedia, pp. 3866–3874 (2020)
21. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2018)
22. Lu, W., Chen, Y., Wang, J., Qin, X.: Cross-domain activity recognition via substructural optimal transport. Neurocomputing **454**, 65–75 (2021)

23. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2009)
24. Patel, V.M., Gopalan, R., Li, R., Chellappa, R.: Visual domain adaptation: a survey of recent advances. IEEE Signal Process. Mag. **32**(3), 53–69 (2015)
25. Rabiner, L., Juang, B.: An introduction to hidden markov models. IEEE ASSP Mag. **3**(1), 4–16 (1986)
26. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: 2012 16th International Symposium on Wearable Computers, pp. 108–109. IEEE (2012)
27. Rokni, S.A., Ghasemzadeh, H.: Autonomous training of activity recognition algorithms in mobile sensors: a transfer learning approach in context-invariant views. IEEE Trans. Mob. Comput. **17**(8), 1764–1777 (2018)
28. Saeedi, R., Sasani, K., Norgaard, S., Gebremedhin, A.H.: Personalized human activity recognition using wearables: a manifold learning-based knowledge transfer. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1193–1196. IEEE (2018)
29. Sekiguchi, R., Abe, K., Yokoyama, T., Kumano, M., Kawakatsu, M.: Ensemble learning for human activity recognition. In: Adjunct proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, pp. 335–339 (2020)
30. Soleimani, E., Nazerfard, E.: Cross-subject transfer learning in human activity recognition systems using generative adversarial networks. Neurocomputing **426**, 26–34 (2021)
31. Sukhija, S., Krishnan, N.C.: Supervised heterogeneous feature transfer via random forests. Artif. Intell. **268**, 30–53 (2019)
32. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
33. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018, Part III. LNCS, vol. 11141, pp. 270–279. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01424-7_27
34. Wang, G., Li, Q., Wang, L., Wang, W., Wu, M., Liu, T.: Impact of sliding window length in indoor human motion modes and pose pattern recognition based on smartphone sensors. Sensors **18**(6), 1965 (2018)
35. Wang, J., Gao, Q., Pan, M., Fang, Y.: Device-free wireless sensing: challenges, opportunities, and applications. IEEE Network **32**(2), 132–137 (2018)
36. Wang, J., Chen, Y., Hu, L., Peng, X., Philip, S.Y.: Stratified transfer learning for cross-domain activity recognition. In: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 1–10. IEEE (2018)
37. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**(2), 207–244 (2009)
38. Wilson, G., Cook, D.J.: A survey of unsupervised deep domain adaptation. ACM Trans. Intell. Syst. Technol. (TIST) **11**(5), 1–46 (2020)
39. Xing, T., Sandha, S.S., Balaji, B., Chakraborty, S., Srivastava, M.: Enabling edge devices that learn from each other: cross modal training for activity recognition. In: Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking, pp. 37–42 (2018)
40. Zhao, Z., Chen, Y., Liu, J., Shen, Z., Liu, M.: Cross-people mobile-phone based activity recognition. In: Twenty-Second International Joint Conference on Artificial Intelligence. Citeseer (2011)



SelfAct: Personalized Activity Recognition Based on Self-Supervised and Active Learning

Luca Arrotta(✉), Gabriele Civitarese, and Claudio Bettini

EveryWare Lab, Department of Computer Science, University of Milan, Milan, Italy
{Luca.Arrotta,Gabriele.Civitarese,Claudio.Bettini}@unimi.it

Abstract. Supervised Deep Learning (DL) models are currently the leading approach for sensor-based Human Activity Recognition (HAR) on wearable and mobile devices. However, training them requires large amounts of labeled data, whose collection is often time-consuming, expensive, and error-prone. At the same time, due to the intra- and inter-variability of activity execution, activity models should be personalized for each user. In this work, we propose *SelfAct*: a novel framework for HAR that combines self-supervised and active learning to mitigate these problems. *SelfAct* leverages a large pool of unlabeled data collected from many users to pre-train through self-supervision a DL model, with the goal of learning a meaningful and efficient latent representation of sensor data. The resulting pre-trained model can be locally used by new users, which will fine-tune it thanks to a novel unsupervised active learning strategy. Our experiments on two publicly available HAR datasets demonstrate that *SelfAct* achieves results that are close to or even better than those reached by fully supervised approaches with only a few active learning queries.

Keywords: Human Activity Recognition · Self-supervised Learning · Active Learning

1 Introduction

The recent research on sensor-based Human Activity Recognition (HAR) on mobile/wearable devices is dominated by solutions based on Deep Learning (DL) approaches [36]. Supervised learning is the most commonly proposed approach in this area since is capable of reaching significantly high recognition rates [29].

However, such approaches require a huge amount of labeled data to create models capable of generalizing. Indeed, the high intra- and inter-variability of activity execution is a significant challenge in this domain, and personalized models (i.e., models trained only with data from the target users) are the ones performing the best [37].

However, annotation is costly, time-consuming, intrusive, and thus often prohibitive. For this reason, several research groups are investigating solutions for HAR requiring limited labeled datasets [28].

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 375–391, 2024.
https://doi.org/10.1007/978-3-031-63989-0_19

Among these approaches, self-supervised learning methods are emerging [15, 19,20]. The goal of self-supervised learning is to leverage large corpora of unlabeled data to learn to extract effective features from sensor data. While self-supervised approaches showed their effectiveness on NLP and computer vision tasks [8], their application to sensor-based HAR in real-world scenarios is still an open research question.

One of the major drawbacks of existing approaches is that they assume that a small labeled dataset for fine-tuning is available. However, those works do not investigate how this dataset can be obtained in practice.

In the literature, active learning has been widely investigated to obtain annotated data samples in HAR data scarcity scenarios [1,18,28]. However, those approaches assume that a pre-training labeled dataset is available to initialize the recognition model. Indeed, a query is triggered based on the confidence level of the classifier. Applying the concept of active learning to a self-supervised setting is not trivial, since it requires deciding when to trigger queries by analyzing an unlabeled data stream.

In this work, we propose *SelfAct*: a novel framework bridging the gap between self-supervised learning and active learning. *SelfAct* relies on self-supervised learning to learn, from a large number of users, an effective feature representation of sensor data. Then, each user who wants to use *SelfAct* downloads this pre-trained model on personal mobile/wearable devices. After the first phase where the personal device accumulates unlabeled sensor data, a clustering algorithm on the embeddings generated by the pre-trained model is performed. After the accumulation phase, *SelfAct* analyzes in real-time each data sample in the stream and matches it with the closest cluster. A novel unsupervised active learning strategy, based on cluster density, is adopted to decide whether to trigger a query or not. Labeled data samples obtained through active learning are finally used to fine-tune the pre-trained self-supervised model and personalize it on the specific user.

The contributions of this work are the following:

- We propose *SelfAct*: a framework for personalized sensor-based HAR on mobile/wearable devices based on self-supervised learning.
- We design and integrate into *SelfAct* a novel unsupervised active learning strategy to obtain labeled data samples. These samples are used to fine-tune a self-supervised model.
- Our experiments on two public datasets show that *SelfAct* achieves similar or even better recognition rates than fully supervised approaches with a very limited amount of active learning queries.

2 Related Work

Most of the existing works in the literature tackling sensor-based HAR on mobile/wearable devices are based on supervised Deep Learning (DL) models [36]. However, during their learning process, such models require large amounts of labeled data that are often prohibitive to acquire [11]. Moreover,

activity models should be personalized for each user to capture her peculiarities in activity execution [11]. Several research groups are indeed working hard to mitigate the data scarcity problem, by proposing several categories of solutions.

Some research efforts proposed solutions based on automatic data annotation. Some of these works assume the existence of previously annotated datasets to pre-train an activity model, that is used for annotation [16]. Other approaches require auxiliary data sources in the environment (e.g., acoustic data from microphones) to automatically annotate activity data [10]. Finally, other works proposed knowledge-based heuristic approaches to generate weak labels (e.g., combining step count and GPS data) [13]. However, the scalability of such an approach is questionable since the heuristic was evaluated considering data collected by monitoring a single user. Moreover, the proposed method can not be used for data annotation when multiple activities share similar patterns in terms of step count and GPS data (e.g., standing and sitting).

Some research groups focused on solutions based on transfer learning [31, 32]. However, such methods rely on models that are pre-trained with significant amounts of labeled data available in a source domain and then fine-tuned in the target domain with a few annotated samples.

Another category of solutions to mitigate the data scarcity problem is semi-supervised learning [2]. However, these approaches still assume the existence of a small labeled dataset to initialize an activity classifier, that is then incrementally updated with samples of the unlabeled data stream annotated through techniques like self-learning, co-learning, active learning [4,14,18], and label propagation [28].

Unsupervised approaches have also been proposed to tackle data scarcity. For instance, some research groups directly relied on clustering techniques to distinguish the different user's activities [3,5,21,35]. However, they did not consider how to match clusters with the actual activity classes the user performs in realistic scenarios.

More recently, self-supervised learning has been studied to efficiently learn (without supervision) meaningful features from sensor data for both mobile [7, 30,34] and smart home [9] activity recognition. The majority of existing solutions assume the availability of a labeled dataset to fine-tune the self-supervised model and obtain the final activity classifier. A closely related work [22] uses a labeled validation dataset to match activities with clusters identified by k-means on self-supervised embeddings. Different from this work, *SelfAct* does not assume the knowledge of the number of activities. Also, *SelfAct* includes an unsupervised active learning strategy to select the samples representing each cluster to label only a few data samples through active learning. Overall, the major drawback of these works based on self-supervised learning is that they do not clarify how fine-tuning datasets can be obtained in realistic settings.

Finally, the work in [17] considered BERT to extract from smart home sensor data a latent representation of the action units performed by the resident (e.g., a movement in the kitchen) that are then clustered. Hence, frequently occurring patterns of action units are found through a motif discovery process. The most

frequent motifs are finally labeled by the resident thanks to active learning. However, NLP-based techniques like BERT and motif discovery work well for discrete environmental sensors, but they cannot be directly applied to the raw inertial measurements collected by mobile and wearable devices.

3 Methodology

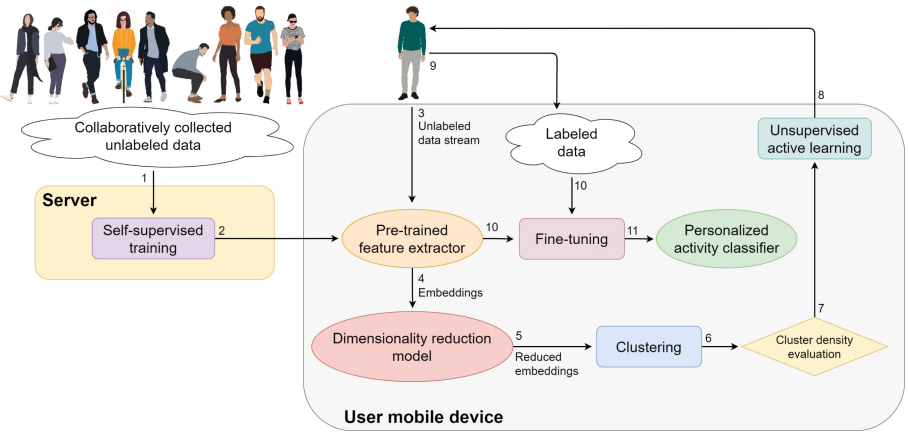


Fig. 1. The overall architecture of *SelfAct*

In this section, we present *SelfAct*: our framework that generates a personalized activity classifier by combining self-supervised learning and unsupervised active learning.

As depicted in Fig. 1, in the first phase (called *self-supervised phase*), *SelfAct* requires huge amounts of unlabeled inertial sensor data automatically and collaboratively collected from the mobile/wearable devices of multiple users (e.g., volunteers). This unlabeled dataset is used to train, in a self-supervised fashion, a deep neural network capable of extracting effective latent representations (i.e., embeddings) from sensor data. This process is described in detail in Sect. 3.1.

In a second phase (called *accumulation phase*), new users start using *SelfAct* by downloading and deploying the pre-trained self-supervised model on their devices. Each user locally accumulates unlabeled inertial sensor data for a certain period. At the end of the *accumulation phase*, such unlabeled data are transformed into embeddings with the pre-trained model and then used to train a personal dimensionality reduction model. Finally, the reduced embeddings are clustered. Intuitively, each cluster would likely represent a personal activity pattern. Since each activity may be performed with several different patterns (e.g., walking indoors or walking outdoors), we expect that each activity is captured by different clusters, each one representing a specific execution pattern. More details about this *accumulation phase* are presented in Sect. 3.2.

After the *accumulation phase*, *SelfAct* starts the *active learning phase* (presented in detail in Sect. 3.3). This phase analyzes, in real-time, each new sensor data sample in the stream to decide whether to ask the user for feedback about the corresponding label (i.e., the activity that the user is actually performing). Intuitively, *SelfAct* aims at asking for a label only whenever a sample is recognized as representative of a cluster.

Thanks to active learning, *SelfAct* automatically collects a small amount of labeled data specific to the subject and this data is used during the *fine-tuning phase* (presented in Sect. 3.4) to obtain a personalized activity classifier.

The *SelfAct*'s algorithm running locally on clients' devices is summarized in Algorithm 1. In the following, we will describe each phase of *SelfAct* in detail.

Algorithm 1. Client algorithm of *SelfAct*

Input: the *Feature Extraction Layers* fe of the pre-trained self-supervised model, the accumulation threshold ACC_TH

$fth \leftarrow$ randomly initialized layers of the *Fine-Tuning Head*
$dr \leftarrow$ NIL
$samplesNumber \leftarrow 0$
$storage \leftarrow \emptyset$
$reducedStorage \leftarrow \emptyset$
$labeledSamples \leftarrow \emptyset$
$clusters \leftarrow \emptyset$
for s data sample in the stream **do**
  $samplesNumber \leftarrow samplesNumber + 1$
  $emb_s \leftarrow$ obtain embeddings using fe
  $storage \leftarrow storage \bigcup \{emb_s\}$
  if $samplesNumber == ACC_TH$ **then**
    $dr \leftarrow$ data reduction model trained using $storage$
    $reducedStorage \leftarrow$ embeddings in $storage$ reduced using dr
    $clusters \leftarrow$ apply clustering algorithm on $reducedStorage$
  else
    if $samplesNumber > ACC_TH$ **then**
      $emb'_s \leftarrow$ dimensionally reduced emb_s using dr model
      $matchingClust \leftarrow$ closest cluster to emb'_s in $clusters$
      if $ActiveLearningNeeded(emb'_s, matchingClust)$ **then**
        $l \leftarrow$ feedback from user on the activity associated with s
        $labeledSamples \leftarrow labeledSamples \bigcup \{(s, l)\}$
      end if
      $matchingClust \leftarrow matchingClust \bigcup \{emb'_s\}$
      if Fine-tuning conditions are verified **then**
        adjust fth to cover all the classes in $labeledSamples$
        fine-tune $fe + fth$ using $labeledSamples$
        $fe + fth$ can be used to classify unlabeled samples
      end if
    end if
  end if
end for

3.1 Self-supervised Phase

The goal of the *self-supervised phase* is to train, in a self-supervised fashion, a deep learning model capable of transforming raw sensor data samples into an efficient latent representation (i.e., embeddings). Intuitively, data samples representing the same activity pattern should be close in the latent space, while data samples from different activity patterns should be distant.

In this phase, *SelfAct* relies on unlabeled raw sensor data that can be collaboratively and automatically collected from the mobile/wearable devices of several users. For the sake of this work, we do not focus on how such data are actually collected. For instance, such users can be volunteers involved in a large data collection campaign. Alternatively, unlabeled data could be collected by combining existing publicly available datasets for HAR.

SelfAct pre-processes and segments the available unlabeled data into fixed-length segmentation windows that are used to train the self-supervised model. During the *self-supervised phase*, this model is composed of two parts: i) *Feature Extraction Layers* that learns how to extract embeddings from sensor data, and ii) a *Self-Supervised Head*, necessary to perform the self-supervised training process. *SelfAct* does not impose a specific self-supervised approach. As we will discuss in Sect. 4, in our experiments we implemented *SelfAct* with a version of the SimCLR [12] approach specifically adapted for the HAR domain [34].

The output of the *self-supervised phase* is a pre-trained deep-learning model only composed of the *Feature Extraction Layers*, that is able to extract embeddings from unlabeled sensor data. This model can be locally deployed on the mobile devices of the new users that start using *SelfAct*.

3.2 Accumulation Phase

The *accumulation phase* starts when a new user downloads and locally deploys on her device the model pre-trained in the *self-supervised phase* (i.e., *Feature Extraction Layers*). The goal of the *accumulation phase* is to accumulate from the user sufficient amounts of unlabeled data to reliably discover personal activity patterns through clustering.

As depicted in Fig. 2,

the user's mobile device starts collecting the stream of unlabeled inertial sensor data that is segmented into fixed-length windows until an arbitrary number of ACC_TH (accumulation threshold) samples is acquired. This parameter should be tuned based on the specific application, and, in real-world scenarios, its value should be high in order to collect unlabeled data over a sufficiently long time period (e.g., a day, or a week).

Once unlabeled data accumulation is completed, each sample is provided to the pre-trained feature extractor to obtain embeddings. Then, to efficiently perform clustering to find activity patterns, we leverage all the generated embeddings to train a personalized dimensionality reduction model. Indeed, recent works demonstrated the impact of dimensionality reduction in improving the performance of clustering techniques [6]. Finally, the dimensionally reduced accumulated embeddings are provided to a clustering algorithm that does not make any prior assumption on the number of clusters (e.g., DBSCAN). The goal is to group in the same cluster those embeddings that have been generated while the user was performing the same activity pattern. We expect our method to obtain more than one cluster for each activity, since users may perform the same activity in different ways (e.g., walking indoors and walking outdoors).

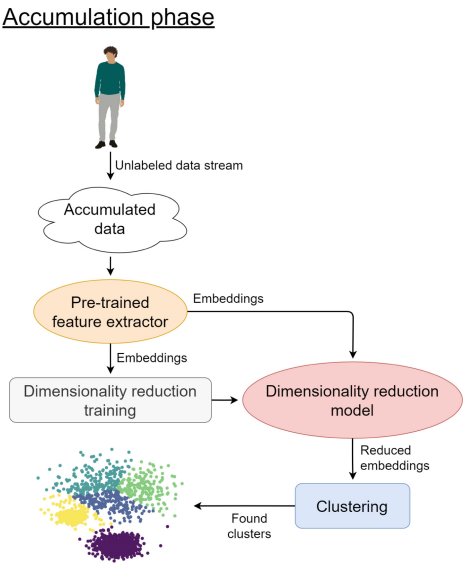


Fig. 2. The *accumulation phase* of *SelfAct*

3.3 Active Learning Phase

The goal of the *active learning phase* is to select, in real-time, data samples that should be labeled by the user. These labeled samples are then used by *SelfAct* to fine-tune a personalized activity classifier.

As shown in Fig. 3, in the *active learning phase*, each new unlabeled sample collected by the user's mobile device is provided to the pre-trained model, and the result is dimensionally reduced through the personalized dimensionality reduction model. Each new reduced embedding is then mapped to the closest cluster among the ones derived during the *accumulation phase* (e.g., the cluster with the closest centroid). The active learning strategy of *SelfAct* asks for a label only for those samples that can be considered as highly representative of the closest cluster (i.e., a specific activity pattern) in order to minimize the number of queries that are prompted to the user.

More specifically, an active learning process is started by evaluating whether a new reduced embeddings sample s improves the density of its cluster or not. *SelfAct* assigns to each cluster c a threshold T_c, which reflects the density level of the cluster. This threshold is automatically computed as the average distance between each pair of points within the cluster.

Intuitively, a new sample s is highly-representative of a cluster c if adding s to c increases the density of the cluster. In particular, we evaluate the average distance between each pair of points in c after adding s to the cluster. If this average distance is lower than T_c, then s is considered highly representative of c, and an active learning query is requested for the user.

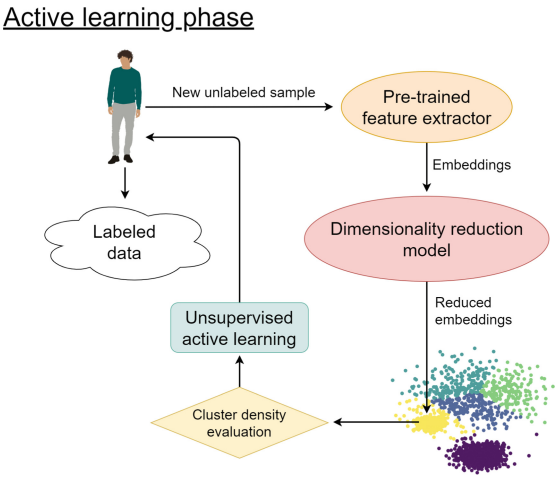


Fig. 3. The *active learning phase* of *SelfAct*

Algorithm 2 shows the pseudo-code of the strategy adopted by *SelfAct* to decide if an active learning query is needed.

Algorithm 2. ActiveLearningNeeded

Input: a new sample emb'_s, and its closest cluster $matchingClust$
Output: a boolean value

1: $avg \leftarrow$ average distance between points in $matchingClust$
2: $avg' \leftarrow$ average distance between points in $matchingClust \bigcup \{emb'_s\}$
3: **if** $avg' < avg$ **then**
4: **return** True
5: **end if**
6: **return** False

Regardless of whether the active learning query is triggered or not, a new sample s is always added to its closest cluster c, and T_c is updated accordingly. The active learning strategy of *SelfAct* assumes that the clusters found at the end of the *accumulation phase* are reliable. For this reason, as we will describe in Sect. 4, we empirically determined ACC_TH, by balancing the trade-off between recognition rate and a number of triggered questions.

3.4 Fine-Tuning Phase

Finally, during the *fine-tuning phase*, *SelfAct* relies on the data labeled through active learning at the previous phase to fine-tune the deep-learning model pre-trained during the *self-supervised phase*. In this way, *SelfAct* builds an activity

classifier that is personalized for the specific user using the framework. Specifically, *SelfAct* dynamically adds to the *Feature Extraction Layers* of the deep-learning model pre-trained during the *self-supervised phase* a *Fine-Tuning Head* that includes classification layers. Since it is not possible to know in advance how many activities (and which) the user actually performed, the *Fine-Tuning Head* is created dynamically based on the number of activity classes in the labeled data obtained through active learning. During the fine-tuning process, the layers of both the *Feature Extraction Module* and the *Fine-Tuning Head* are unfrozen.

The *fine-tuning phase* should be triggered by considering specific conditions. For instance, it can be a periodic task (e.g., with a time periodicity or each time a certain amount of labels is collected). However, defining when fine-tuning should be started is out of the scope of this paper and it will be investigated in future work. Indeed, periodic fine-tuning would also require strategies to mitigate catastrophic forgetting.

4 Experimental Evaluation

This section describes the experiments we carried out to evaluate *SelfAct*. We first introduce the two publicly available datasets we considered for our experiments. Then, we discuss details about our experimental setup: the data preprocessing, the specific models we used to implement *SelfAct*, and the evaluation methodology we adopted. Finally, we present the results of our evaluation.

4.1 Datasets

HHAR. The Heterogeneity Human Activity Recognition (HHAR) dataset [33] includes accelerometer and gyroscope data from 9 users wearing a smartphone (in a pouch carried on the waist) and a smartwatch. The dataset was collected considering 6 different activities: *biking*, *sitting*, *standing*, *walking*, *stairs up*, and *stairs down*. HHAR includes heterogeneous sampling rates, based on the specific devices worn by the user. In particular, the dataset includes 2 different smartwatch devices: LG Watch and Samsung Galaxy Gears with maximum sampling rates of 200 Hz and 100 Hz, respectively. Moreover, HHAR contains 4 different smartphone models: Samsung Galaxy S3 mini (maximum sampling rate of 100 Hz), Samsung Galaxy S3 (150 Hz), LG Nexus 4 (200 Hz), and Samsung Galaxy S+ (50 Hz).

MotionSense. MotionSense [24] is a publicly available HAR dataset collected from 24 subjects carrying an iPhone 6 s in their trousers' front pocket, and performing 6 different activities: *walking downstairs*, *walking upstairs*, *walking*, *jogging*, *sitting*, and *standing*. The dataset includes both accelerometer and gyroscope data sampled at 50 Hz.

4.2 Experimental Setup

In the following, we describe our experimental setup.

Machine Learning Models. As a self-supervised learning model, in our experiments, we used a version of the *SimCLR* technique [12] adapted for HAR in previous works [15,34]. Consistently with the previous work, the *Feature Extraction Layers* consist of three 1D convolutional layers with 32, 64, and 96 filters and 24, 16, and 8 kernel sizes, respectively, followed by a global maximum pooling layer. Each couple of consecutive layers is separated by a dropout layer with a dropout rate of 0.1. The *Self-Supervised Head* consists of three fully connected layers with 256, 128, and 50 neurons, respectively. The self-supervised model is pre-trained for 50 epochs and a batch size of 512 with the SGD optimizer, considering a cosine decay for the learning rate. Moreover, we empirically determined the optimal transformations to apply to the unlabeled data during SimCLR pre-training, finding that, consistently with [34], the application of the rotation transformation only was the best option.

The *Fine-Tuning Head* consists of a fully connected layer containing 1024 neurons and a softmax layer for the final classification. Fine-tuning is performed by running 50 training epochs with a batch size of 1, using an early stopping technique that stopped the training process when validation loss did not improve for 5 consecutive epochs, and adopting the Adam optimizer.

As a dimensionality reduction model, we adopted the Uniform Manifold Approximation and Projection (UMAP) technique [26] since previous work demonstrated how its use is effective in considerably improving the performance of clustering algorithms [6]. In preliminary experiments, we observed that other dimensionality reduction approaches were less effective. For instance, we obtained very negative results with PCA, probably due to the fact that it is a linear dimensionality reduction that is not effective in capturing the non-linear nature of sensor data. On the other hand, UMAP is effective in preserving global data relationships and capturing non-linearity [26].

Moreover, since *SelfAct* considers a real-time scenario for active learning, it is crucial to train a dimensionality reduction model only once at the end of the accumulation phase, and then to map new data samples in the reduced space. These real-time constraints can be addressed by using UMAP, while t-SNE [23], for example, would not be appropriate. During our evaluation, we experimentally determined through a grid search approach the best hyperparameters for UMAP: 15 as the number of neighbors, and an euclidean minimum distance equal to 0.4.

Realistic and real-time constraints mainly guided the choice of the clustering algorithm of *SelfAct*. First of all, it is not realistic to assume knowing the exact number of different activity patterns that the user will perform while using *SelfAct*. Thus, it would not be appropriate to consider clustering algorithms that require knowing the number of clusters in advance (e.g., K-means). Moreover, in real-time scenarios, we need to rely on an algorithm that (i) clusters the reduced embeddings accumulated during the *accumulation phase*, and then (ii) directly maps new data into the existing clusters during the *active learning phase*. For

these reasons, in our experiments, we relied on the HDBSCAN clustering algorithm [25]. In particular, we empirically determined with a grid search process its hyperparameters: a minimum cluster size of 6, and a minimum number of samples of 10.

Pre-processing. Since the version of SimCLR we are using only considers data from a single tri-axial sensor as input, we focus on the accelerometer data collected through the users' smartphones. Such data are segmented into fixed-length windows with a 50% of overlap. In particular, following the evaluation protocol proposed in [34], we considered windows of 400 samples for MotionSense. On the other hand, since HHAR includes data collected with different sampling rates, we experimentally determined the size of the windows, obtaining a window size of 800 samples. Due to unsatisfying recognition rates during preliminary experiments, in MotionSense we grouped the labels *walking downstairs* and *walking upstairs* into the label *walking stairs*, while in HHAR we grouped *stairs down* and *stairs up* into *stairs*. Indeed, these activities are particularly challenging to discriminate.

Baseline. In our evaluation, we compared *SelfAct* with a *fully supervised* approach: a deep neural network with the same architecture of the network used by *SelfAct* during the *fine-tuning phase*, trained in a supervised way considering full availability of labeled training data.

Evaluation Methodology. The *SelfAct* framework is designed for personalizing the HAR model for each user by leveraging a limited amount of labeled data. Hence, we performed our experiments considering the leave-one-subject-out cross-validation technique. At each iteration of the cross-validation process, data from one user is used as the test set (i.e., accumulation, active learning, and fine-tuning phases), while the remaining folds are used as the training set (i,e., self-supervised phase). At the end of the process, we averaged the results obtained at each iteration in terms of weighted F1 score.

Considering the *fully supervised* baseline, at each fold, 10% of the training set was used as the validation set. On the other hand, in order to simulate the *accumulation* and *active learning* phases of *SelfAct*, a certain percentage (i.e. *accumulation threshold*) of the test set was used to simulate the samples that are accumulated by *SelfAct* during the *accumulation phase*. The remaining samples of the test set were used to simulate the data stream processed during the *active learning phase*[1].

During fine-tuning, we used 10% of labeled data as the validation set. Finally, all the samples in the test set that were not labeled (i.e., the ones collected in the *accumulation phase* and the samples not labeled during the *active learning*

[1] This setting resembles a scenario in which practitioners must allocate both *accumulation* and the *active learning* phases within a given timeframe (e.g., one month) that is constrained by deployment or project requirements.

phase) were used to assess the recognition rate of the fine-tuned model. We also compute the *active learning rate*, i.e., the ratio between the amount of triggered active learning questions and the total number of samples available during the *active learning phase*.

While it was straightforward to apply the leave-one-out cross-validation on HHAR, it was more challenging on MotionSense. Indeed, users in this dataset do not have enough data samples to obtain meaningful results from the *accumulation* and *active learning* phases. Hence, we created pseudo-users by grouping actual MotionSense users performing activities in a similar way (based on available meta-data like age, height, weight, etcetera). This resulted in 5 pseudo-users (with each pseudo-users including data from 4 or 5 MotionSense users), that we used to perform the leave-one-out cross-validation.

4.3 Results

In the following, we describe the results we obtained during our experimental evaluation.

Recognition Rate. Table 1 compares *SelfAct* and the *fully supervised* baseline in terms of F1 score and the average number of necessary labeled data to deploy such solutions on both datasets.

Table 1. Main results of *SelfAct*. The *accumulation phase* was performed on 75% of the test set. The table reports the average number of active learning queries prompted to the user by *SelfAct* (i.e., number of samples for fine-tuning), and the average number of labeled samples in the training set used to train *fully supervised*

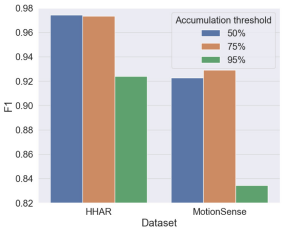
Dataset	Average number of active learning queries (fine-tuning samples)	*SelfAct* F1	Average supervised training set size	*Fully supervised* F1
HHAR	369	0.9734	25053	0.8792
MotionSense	143	0.9291	5304	0.9483

Such results were obtained by accumulating during the *accumulation phase* a number of samples that corresponds to 75% of the data available in the test set. Considering HHAR, *SelfAct* dramatically outperforms *fully supervised* in terms of F1 score (i.e., $\approx +10\%$). Moreover, this result is obtained considering only 369 labeled samples per user, which is a reduction of approximately 98% compared to the amount of labeled data required by the *fully supervised* baseline (i.e., $\approx 25k$ labeled samples on average at each iteration of the cross-validation).

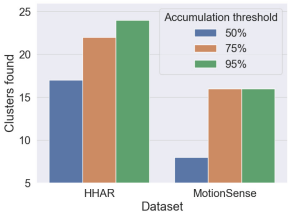
On the other hand, considering MotionSense, our framework did not improve the recognition rates of *fully supervised*. This happens since the *active learning phase* of *SelfAct* is actually computed on the data of more users (see Sect. 4.2).

Hence, with a few fine-tuning samples labeled with active learning, it is hard to cover all the possible ways in which an activity can be performed by different users. Nonetheless, the F1 score reached by *SelfAct* is only $\approx 2\%$ behind *fully supervised*, but relying only on 143 labeled samples per group of users included in the test set, instead of considering on average more than $5k$ labeled training samples. Hence, we believe that these results still highlight the effectiveness of *SelfAct*, which reaches recognition rates close to supervised approaches with a limited amount of labeled samples.

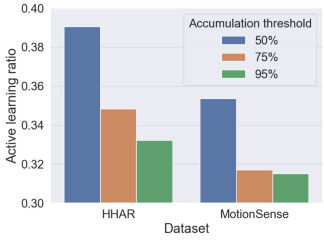
Impact of Accumulation Threshold. Figure 4 depicts how the *accumulation threshold* affects the performance of *SelfAct*.



(a) Recognition rates of *SelfAct* by varying the *accumulation threshold*



(b) Number of clusters derived by *SelfAct* at the end of the *accumulation phase* by varying the *accumulation threshold*



(c) Active learning rate during the *active learning phase* of *SelfAct* by varying *accumulation threshold*

Fig. 4. The impact of *accumulation threshold* on recognition rate, number of clusters, and active learning rate

By setting the *accumulation threshold* at 50% (i.e., 50% of data used for accumulation and 50% of data in the active learning phase) we get a high recognition rate, at the cost of having the highest active learning rate. Indeed, with this threshold, *SelfAct* finds a lower number of sparse clusters, thus requiring

a higher number of samples during the active learning phase to increase their densities. This also implies a higher number of active learning queries. It is important to recall that the number of clusters should not necessarily coincide with the number of activity classes. Indeed, in HAR, it is well-known that the same activity can be performed in very different ways by the subject (e.g., walking can be performed indoors and outdoors). Hence, each activity class can be potentially represented by multiple activity clusters.

On the other hand, when the accumulation threshold is 95% (i.e., 95% of data used for accumulation and 5% of data in the active learning phase) the recognition rate is the lowest. This is due to the fact that this setting does not provide a sufficient amount of data in the active learning phase for properly fine-tuning the model.

Hence, we observed that setting the threshold to 75% (i.e., 75% of data used for accumulation and 25% of data in the active learning phase) led to a high recognition rate (that is similar to the one obtained with the threshold set to 50%) and, at the same time, a significantly reduced active learning rate. Indeed, this setting allows *SelfAct* to identify clusters that better reflect activity patterns and that can be effectively used for the active learning phase.

In general, our results show that increasing the accumulation threshold decreases the active learning rate. This is due to the fact that our unsupervised learning strategy is based on cluster density. Indeed, the higher the amount of accumulated data and the denser the clusters.

5 Conclusion and Future Work

In this paper, we presented *SelfAct*: a framework for personalized sensor-based HAR. Thanks to a novel combination of self-supervised learning and active learning, *SelfAct* requires only a limited amount of labeled data from target users to fine-tune a model pre-trained with unlabeled data only. While the results obtained so far are promising, we plan to investigate several aspects in future work.

First, even though *SelfAct* does not assume knowing in advance the activities performed by the target user, in this work the pre-trained feature extractor and the fine-tuned activity classifier are trained considering the same activity classes. In future work, we want to truly assess the capability of *SelfAct* for activity discovery and domain adaptation. A challenge will be to distinguish outliers from new activities.

A drawback of *SelfAct* is that it relies on a fixed threshold to decide when switching from *accumulation phase* to *active learning phase*. In future work, we will investigate automatic methods based on the clusters' density (e.g., computing the DBCV score [27] of the clusters).

Another limitation of *SelfAct* is that, while it requires limited labeled data samples to obtain high recognition rates, the active learning query rate is relatively high. This aspect has a negative impact on usability. We will investigate more sophisticated active learning strategies aiming at further minimizing the

number of required samples. For instance, an interesting strategy is to establish a budget of queries for each cluster. Nonetheless, these results are also probably due to the fact that the considered datasets are small, and we could not evaluate *SelfAct* with large amounts of data for the accumulation and active learning phases. We expect that, in real-world scenarios where it is possible to collect a large amount of unlabeled data, the active learning rate would be significantly lower.

SelfAct also assumes that it is possible to store and process locally unlabeled and labeled datasets. However, in real-world scenarios, unlabeled data may be collected for a significantly long time. For instance, the accumulation phase may last several days. We will investigate solutions based on trusted edge devices with storage and computational capabilities. Users' devices should be only in charge of running the fine-tuned classifier.

Acknowledgment. Part of this research was supported by projects SERICS (PE00000014) and by project MUSA - Multilayered Urban Sustainability Action, funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strenghtening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D.

References

1. Abdallah, Z.S., Gaber, M.M., Srinivasan, B., Krishnaswamy, S.: Streamar: incremental and active learning with evolving sensory data for activity recognition. In: 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, vol. 1, pp. 1163–1170. IEEE (2012)
2. Abdallah, Z.S., Gaber, M.M., Srinivasan, B., Krishnaswamy, S.: Activity recognition with evolving data streams: a review. ACM Comput. Surv. (CSUR) **51**(4), 1–36 (2018)
3. Abedin, A., Motlagh, F., Shi, Q., Rezatofighi, H., Ranasinghe, D.: Towards deep clustering of human activities from wearables. In: Proceedings of the 2020 ACM International Symposium on Wearable Computers, pp. 1–6 (2020)
4. Adaimi, R., Thomaz, E.: Leveraging active learning and conditional mutual information to minimize data annotation in human activity recognition. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **3**(3), 1–23 (2019)
5. Ahmed, A., Haresamudram, H., Ploetz, T.: Clustering of human activities from wearables by adopting nearest neighbors. In: Proceedings of the 2022 ACM International Symposium on Wearable Computers, pp. 1–5 (2022)
6. Allaoui, M., Kherfi, M.L., Cheriet, A.: Considerably improving clustering algorithms using UMAP dimensionality reduction technique: a comparative study. In: El Moataz, A., Mammass, D., Mansouri, A., Nouboud, F. (eds.) ICISP 2020. LNCS, vol. 12119, pp. 317–325. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51935-3_34
7. Almaslukh, B., AlMuhtadi, J., Artoli, A.: An effective deep autoencoder approach for online smartphone-based human activity recognition. Int. J. Comput. Sci. Netw. Secur **17**(4), 160–165 (2017)

8. Baevski, A., Hsu, W.N., Xu, Q., Babu, A., Gu, J., Auli, M.: Data2vec: a general framework for self-supervised learning in speech, vision and language. In: International Conference on Machine Learning, pp. 1298–1312. PMLR (2022)
9. Bouchabou, D., Nguyen, S.M., Lohr, C., LeDuc, B., Kanellos, I.: Using language model to bootstrap human activity recognition ambient sensors based in smart homes. Electronics **10**(20), 2498 (2021)
10. Chatterjee, S., Chakma, A., Gangopadhyay, A., Roy, N., Mitra, B., Chakraborty, S.: Laso: exploiting locomotive and acoustic signatures over the edge to annotate IMU data for human activity recognition. In: Proceedings of the 2020 International Conference on Multimodal Interaction, pp. 333–342 (2020)
11. Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., Liu, Y.: Deep learning for sensor-based human activity recognition: overview, challenges, and opportunities. ACM Comput. Surv. (CSUR) **54**(4), 1–40 (2021)
12. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, pp. 1597–1607. PMLR (2020)
13. Cruciani, F., Cleland, I., Nugent, C., McCullagh, P., Synnes, K., Hallberg, J.: Automatic annotation for human activity recognition in free living using a smartphone. Sensors **18**(7), 2203 (2018)
14. Cui, Y., Hiremath, S.K., Ploetz, T.: Reinforcement learning based online active learning for human activity recognition. In: Proceedings of the 2022 ACM International Symposium on Wearable Computers, pp. 23–27 (2022)
15. Haresamudram, H., Essa, I., Plötz, T.: Assessing the state of self-supervised human activity recognition using wearables. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **6**(3), 1–47 (2022)
16. Hassan, I., Mursalin, A., Salam, R.B., Sakib, N., Haque, H.Z.: Autoact: an auto labeling approach based on activities of daily living in the wild domain. In: 2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR), pp. 1–8. IEEE (2021)
17. Hiremath, S.K., Nishimura, Y., Chernova, S., Plötz, T.: Bootstrapping human activity recognition systems for smart homes from scratch. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **6**(3), 1–27 (2022)
18. Hossain, H.S., Roy, N.: Active deep learning for activity recognition with context aware annotator selection. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1862–1870 (2019)
19. Jain, Y., Tang, C.I., Min, C., Kawsar, F., Mathur, A.: Collossl: collaborative self-supervised learning for human activity recognition. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **6**(1), 1–28 (2022)
20. Khaertdinov, B., Ghaleb, E., Asteriadis, S.: Contrastive self-supervised learning for sensor-based human activity recognition. In: 2021 IEEE International Joint Conference on Biometrics (IJCB), pp. 1–8. IEEE (2021)
21. Kwon, Y., Kang, K., Bae, C.: Unsupervised learning for human activity recognition using smartphone sensors. Expert Syst. Appl. **41**(14), 6067–6074 (2014)
22. Ma, H., Zhang, Z., Li, W., Lu, S.: Unsupervised human activity representation learning with multi-task deep clustering. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **5**(1), 1–25 (2021)
23. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**(11) (2008)

24. Malekzadeh, M., Clegg, R.G., Cavallaro, A., Haddadi, H.: Protecting sensory data against sensitive inferences. In: Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems, pp. 1–6 (2018)
25. McInnes, L., Healy, J., Astels, S.: hdbscan: Hierarchical density based clustering. J. Open Source Softw. **2**(11), 205 (2017)
26. McInnes, L., Healy, J., Melville, J.: UMAP: uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
27. Moulavi, D., Jaskowiak, P.A., Campello, R.J., Zimek, A., Sander, J.: Density-based clustering validation. In: Proceedings of the 2014 SIAM International Conference on Data Mining, pp. 839–847. SIAM (2014)
28. Presotto, R., Civitarese, G., Bettini, C.: Semi-supervised and personalized federated activity recognition based on active learning and label propagation. Pers. Ubiq. Comput. **26**(5), 1281–1298 (2022)
29. Ronald, M., Poulose, A., Han, D.S.: iSPLinception: an inception-resnet deep learning architecture for human activity recognition. IEEE Access **9**, 68985–69001 (2021)
30. Saeed, A., Ozcelebi, T., Lukkien, J.: Multi-task self-supervised learning for human activity detection. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **3**(2), 1–30 (2019)
31. Sanabria, A.R., Zambonelli, F., Ye, J.: Unsupervised domain adaptation in activity recognition: a GAN-based approach. IEEE Access **9**, 19421–19438 (2021)
32. Soleimani, E., Nazerfard, E.: Cross-subject transfer learning in human activity recognition systems using generative adversarial networks. Neurocomputing **426**, 26–34 (2021)
33. Stisen, A., et al.: Smart devices are different: assessing and mitigatingmobile sensing heterogeneities for activity recognition. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 127–140 (2015)
34. Tang, C.I., Perez-Pozuelo, I., Spathis, D., Mascolo, C.: Exploring contrastive learning in human activity recognition for healthcare. arXiv preprint arXiv:2011.11542 (2020)
35. Trabelsi, D., Mohammed, S., Chamroukhi, F., Oukhellou, L., Amirat, Y.: An unsupervised approach for automatic activity recognition based on hidden Markov model regression. IEEE Trans. Autom. Sci. Eng. **10**(3), 829–835 (2013)
36. Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L.: Deep learning for sensor-based activity recognition: a survey. Pattern Recogn. Lett. **119**, 3–11 (2019)
37. Weiss, G.M., Lockhart, J.: The impact of personalization on smartphone-based activity recognition. In: Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence (2012)



A Novel Method for Wearable Activity Recognition with Feature Evolvable Streams

Yixiao Wang[1,2], Chunyu Hu[1,2(✉)], Hong Liu[3], Lei Lyu[3], and Lin Yuan[1,2]

[1] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China
10431210389@stu.qlu.edu.cn, {hcy,yuanl}@qlu.edu.cn
[2] Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan, China
[3] School of Information Science and Engineering, Shandong Normal University, Jinan, China
lvlei@sdnu.edu.cn

Abstract. Wearable activity recognition plays an important role in human health monitoring. Traditional wearable activity recognition models are trained in an offline mode with static and pre-defined sensor configurations. However, in real scenarios, data arrives in streams and wearable sensors dynamically appear or disappear, resulting in corresponding changes in the feature space, which is referred to as feature evolution. Addressing the issue of feature evolution is a significant challenge in wearable activity recognition. In this paper, we propose a new method, namely Online Learning method for Feature Evolvable Streams (OLFES). OLFES learns the optimal model depth online according to the complexity of the data stream, recovers the old features through the feature space generation strategy, and finally integrates the prediction results according to a weighted combination strategy. Extensive experimental results on data science datasets and activity recognition datasets demonstrate the feasibility and effectiveness of the proposed method.

Keywords: Activity recognition · Feature evolution · Online learning

1 Introduction

Human activity recognition (HAR) has become a hot topic for many areas, including healthcare, interactive games and sports [1,2]. It is able to detect and identify human activities by analyzing the data collected from various types of sensors. Wearable sensors, such as heart rate monitoring sensors, accelerometers, gyroscopes, and magnetometers, exhibit robustness to occlusion, point-of-view variations, illumination changes, and background fluctuations. This characteristic enables individuals to transcend the limitations imposed by their immediate

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 392–407, 2024.
https://doi.org/10.1007/978-3-031-63989-0_20

environment and offers significant advantages in activity recognition. Therefore, wearable activity recognition is gaining increasing popularity among a growing number of individuals.

Existing activity recognition methods typically assume that data is collected with fixed sensors. The feature space will not undergo any changes. However, this assumption does not hold in realistic scenarios. Specifically, in an open and dynamic environment, the feature space evolves over time as sensors appear or disappear. Traditional activity recognition models face two challenges. Firstly, they are unable to utilize the data from newly emerged sensors to recognize user behavior. Secondly, they fail to function properly when existing sensors are removed. A straightforward idea is to learn a new activity recognition model on the new feature space and use the new model to recognize user's activities. However, there are some weaknesses in this method. On one hand, the new feature space has just emerged and the collected samples are insufficient to train a well-performing model. On the other hand, the knowledge learned by the old model on the vanished old feature space cannot be reused, resulting in a waste overhead of previously collected data. To solve these problems, a new learning paradigm for online learning from evolvable feature spaces, named feature evolution learning [3]. There have been many related researches [4–6] and have been applied in many fields [7,8]. It not only rely on the current features, but also try to leverage the knowledge stored in the existing model by recovering the disappeared features.

In the pioneering work [3], an important observation is that the feature space does not change in an arbitrary way, and there is an evolving phase in the process of change in which the old feature space and the current feature space exist simultaneously. In the activity recognition task, we usually put on a new wearable device in advance before the old one runs out of power. Therefore, Hou [3] et al. used Online Gradient Descent (OGD) to update the model in the old feature space and the model in the new feature space. At the same time, the mapping matrix from the current feature to the old feature is learned by taking advantage of the simultaneous existence of the old and new feature spaces in the evolution stage, so as to establish the mapping relationship between the two feature spaces. After the old feature space disappears, the new model learned from the current feature space can be aided by the historical data in the old feature space by mapping from the current feature space to the old feature space. Liu [9] et al. introduce a second-order online learning method into the evolvable feature space to improve the convergence speed of the model. Zheng [10] proposed a feature evolution learning method for image streams by replacing the fully connected layer in Online Deep Learning (ODL) with ResNet and approximating the connection between two feature spaces by minimizing the potential representation mapping. Lian [11] et al. use variational self-encoders to map two feature spaces into a shared latent subspace that can encode arbitrary expressible mapping functions for feature reconstruction. However, for sensor stream data, shallow learners have limited learning capabilities and cannot face high-dimensional features. Meanwhile, linear mapping as well as self-coding reconstruction are not sufficient to recover the vanished feature space well.

To address these challenges, we propose a novel Online Learning method for Feature Evolvable Streams, namely OLFES. OLFES is designed to support the appearance and disappearance of sensors in real scenes. Specifically, OLFES updates the classifier online by Hedge Backpropagation (HBP), which not only learns the predictive model sequentially from the data stream, but also enables the transition from shallow to deep models. In the evolving stage, a mapping from new features to old features is learned through a feature space generation strategy, then the learned mapping relation is used to recover the old features when only the new feature space is available, and finally the prediction results of the old and new models are integrated in a weighted combination. Experiments were conducted on datasets from both data science and activity recognition domains, and OLFES was able to obtain better classification accuracy and minimal regret.

The rest of the paper is organized as follows. In Sect. 2, we present the related work on activity recognition and feature evolvable stream learning. In Sect. 3, we present the problem description of OLFES. In Sect. 4, we describe the proposed OLFES method in detail. In Sect. 5, extensive experiments are conducted to evaluate the performance of OLFES and specific experimental results are given. Conclusions and future work are presented in Sect. 6.

2 Relate Work

2.1 Activity Recognition

Traditional human activity recognition is mainly based on pattern recognition and feature engineering, which consists of two main parts: feature extraction and classification algorithms. Feature extraction requires the researcher to manually design the extracted features. For example, Bao and Intille [12] collected a large amount of accelerometer data using an accelerometer sensor and then manually extracted four features of the data samples: mean, energy, frequency domain entropy, and correlation coefficient, and then fed these extracted features to a classifier for training. In terms of classification algorithms, various machine learning methods have been applied in the field of wearable activity recognition, such as SVM [13], decision trees [14].

Compared to shallow machine learning models, deep learning models using neural networks have significant advantages in terms of classification accuracy [15]. In [16], a deep learning algorithm based on deep self-encoders is used to reduce the feature dimensionality of the data, minimize unwanted activities and extract hierarchical features of the data to learn the most appropriate feature vector for activity recognition. In literature [17], Vollmer C et al. used sparse coding method to learn human activity features using feature dimension reduction strategy to reduce computational complexity and time. In the literature [18], Zeng et al. proposed a convolutional neural network-based human activity recognition method, which can automatically extract features for activity recognition by capturing the local dependence and scale invariance of the signal, and changes in the same activity can be easily captured. Wang [19] et al. proposed a neural network framework consisting of CNN and bidirectional LSTM to extract spatial

and temporal data from the cell phone the raw acceleration data from mobile phones to extract spatial and temporal features. Wen [20] developed their own convolutional and recurrent network architectures, but did not illustrate their performance in real-world heterogeneous environments.

However, the existing work requires that the dataset is fully available before training and that the existing models are not applicable to scenarios with changing feature spaces. Real-world scenarios where data often arrives in streams and the number of sensors changes dynamically due to various factors make static models inapplicable.

2.2 Feature Evolvable Stream Learning

In an open environment [21], the feature dimension of the data is likely to change. The earliest study of feature changes was incremental learning of features in incremental learning. Guan [22] et al. proposed the ITID algorithm to solve the problem of new features appearing in the training of neural networks. When new features appear, they do not train from scratch using new data, but train the old network with the old features, train the new network on the new features, and finally combine them together. The paper [23] combined the PCA method based on ITID algorithm to reduce the interference between input features and improve the accuracy of model classification. Hu [24] et al. proposed FIRF, a random forest-based feature incremental learning method, to solve the problem of incremental features in wearable activity recognition due to the addition of new sensors. However, the above method does not consider the scenario of feature disappearance in an online environment.

In [3], Hou et al. proposed a feature evolution scenario that includes feature addition and disappearance, which assumes that the change of features is not arbitrary but there is an evolving stage, and learns the mapping relationship between the two feature spaces during the evolving so as to recover the data using the newly emerged features after the initial feature space disappears, and trains the classifier on the two feature spaces separately, and finally integrates the two prediction results are integrated. Zhang [25] et al. considered a more realistic scenario where the disappearance of features is temporary and arbitrary, and proposed an adaptive integration learning method FIDE for the problem of feature increment and decrement in gesture recognition. Based on [3], Zhang [26] et al. further considered the scenario where features and distributions evolve simultaneously, and proposed evolutionary differences to measure differences between successive batches of data and implement a difference minimization algorithm using deep neural networks. Similarly, Hou [27] et al. proposed the one-time incremental decremental learning approach (OPID), which assumes the scenario that there will be an overlap of old and new features when the features evolve, i.e., the old features will not disappear completely when they disappear, but only partially, and the model that learns the disappearing features and the surviving features well will make predictions for the surviving features, and then the multiclassification vectors obtained from the predictions and the new features will be put together to learn the new. Zheng [10] proposed a feature evolution

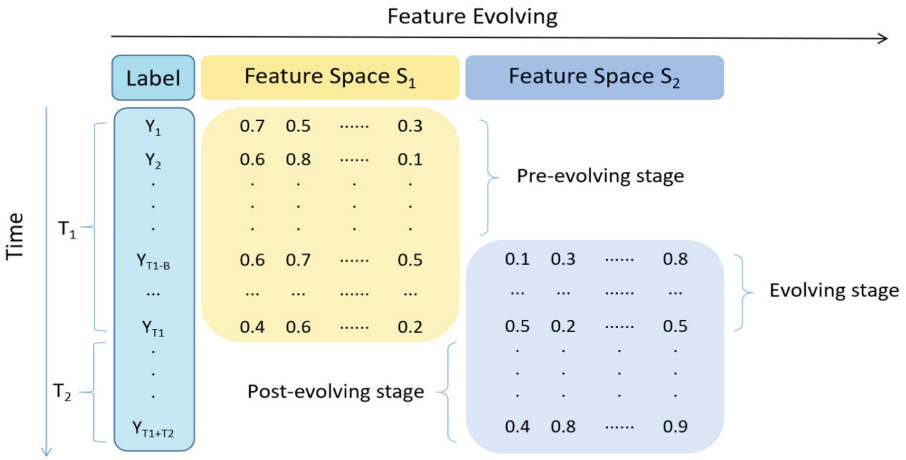


Fig. 1. Illustration of feature evolution

learning method for image streams by replacing the fully connected layer in a neural network with ResNet and approximating the connection between two feature spaces by minimizing the potential representation mapping. The current state-of-the-art method is proposed by Lian [11] et al. who combine online deep learning with a variational self-coding mapping approach to feature evolving streams. However, the above methods are not suitable for complex and variable sensor data streams.

3 Problem Statement

We focus on the data stream classification task in the feature space evolution scenario. Assume $\{(x_t, y_t)|t = 1, 2, ..., T_1 + T_2\}$ is the input data stream, where x_t is the instance observed at moment t. The true label $y_t \in \{1, 2, ..., c\}$, where c is the number of classes. In the wearable activity recognition task, we assume S_1 and S_2 denote the previous feature space and the current feature space, respectively. Consistent with the scenario presented in [3], OLFES can also be divided into three stages: the pre-evolving stage, the evolving stage and the post-evolving stage. As shown in Fig. 1, in the pre-evolving stage, the data stream comes from the old feature space S_1. Then, in the evolving stage, a small number of instances come from both the old feature space S_1 and the new feature space S_2. Subsequently, in the post-evolving stage, instances come only from the new feature space S_2. A detailed description of the three time periods is given below.

- Pre-evolving stage: For each round in $t = 1, ..., T_1 - B$, the classifier receives an instance $x_t^{S_1} \in \mathbb{R}^{d_1}$ collected from the feature space S_1, where d_1 is the dimension of the feature space S_1 and T_1 is the number of rounds in S_1.
- Evolving stage: For each round in $t = T_1 - B + 1, ..., T_1$, the classifier receives an instance consisting of two vectors x_t^{S1} and x_t^{S2}. $x_t^{S_1} \in \mathbb{R}^{d_1}$ and $x_t^{S_2} \in \mathbb{R}^{d_2}$

are sampled from feature spaces S_1 and S_2, where d_2 is the dimensionality of feature space S_2.

- Post-evolving stage: For each round in $t = T_1 + 1, ..., T_1 + T_2$, the classifier receives an instance $x_t^{S_2} \in \mathbb{R}^{d_2}$ sampled from the feature space S_2, where T_2 is the number of rounds of S_2.

Note that in Fig. 1 we only show the feature evolution learning process in one cycle which contains three different stages. This evolution can be extended to the case with multiple cycles easily, where the post-evolving stage can be seen as the pre-evolving stage for the next cycle.

At any moment t in the cycle, the classifier f_t receives the instance x_t. Assume the prediction is denoted as $\hat{y}$. Our goal is to learn a series of classifiers $\{f_1, f_2, ..., f_{T_1+T_2}\}$, that minimizes the empirical risk:

$$min_{f_1,...,f_{T_1+T_2}} \frac{1}{T_1+T_2} \sum_{t=1}^{T_1+T_2} \mathcal{L}(y_t, f_t(x_t)), \tag{1}$$

where $\mathcal{L}(\cdot,\cdot)$ denotes a loss function, such as cross-entropy loss or square loss.

4 Our Method

In this section, we present the proposed OLFES method. This section is organized as follows: First, in Sect. 4.1 we introduce the online deep learning method with HBP, when S_1 and S_2 are newly available we train deep adaptive models in an online manner. Then, Sect. 4.2 presents the feature space generation strategy used to recover old features by reconstructing S_1 from S_2. Finally, Sect. 4.3 describes how to combine the models learned in the old and new feature spaces in a weighted manner to obtain the final prediction. The architecture of OLFES is described in Fig. 2.

4.1 Online Deep Learning with HBP

Traditional neural network algorithms usually require the entire training data to be available before training. However, there are many realistic scenarios where data arrives in the form of streams, limiting the use of neural networks. Therefore, a Hedge Backpropagation method is proposed to enable the use of deep neural networks in an online environment [28].

Assuming a deep neural network with L hidden layers, the final prediction result can be expressed as:

$$F(x) = \sum_{l=0}^{L} \alpha^{(l)} f^{(l)}. \tag{2}$$

where $f^{(l)} = softmax\left(h^{(l)}\theta^{(l)}\right), \forall l = 0, ..., L$, $h^{(l)} = \sigma\left(W^{(l)}h^{(l-1)}\right), \forall l = 1, ..., L$, and $h^{(0)} = x$.

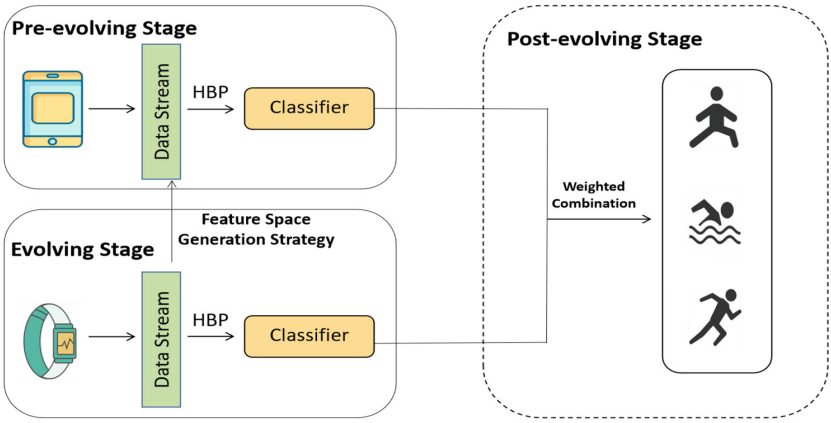


Fig. 2. Architectural description of our OLFES

The final prediction results in the use of $h^{(0)}, ..., h^{(t)}$ feature representations of the weighted combinations of the learned classifiers, thus allowing the learner to acquire network knowledge from "shallow to deep". The weight of each classifier is denoted as $\alpha^{(l)}$. Our goal is to learn and update the parameters $\alpha^{(l)}$, $\theta^{(l)}$ and $W^{(l)}$ in the model and train the model by minimizing the loss function: $\mathcal{L} = \sum_{l=0}^{L} \alpha^{(l)} \mathcal{L}(f^{(l)}(x), y)$.

For the update of α, in the initial iteration, all the weights α are uniformly distributed, i.e., $\alpha^{(l)} = \frac{1}{L+1}$. In each subsequent iteration, the classifier $f^{(l)}$ makes a prediction $\hat{y}_t^{(l)}$. Subsequently, the basic fact is revealed that the weights of the classifier are updated according to the losses in this round:

$$\alpha_{t+1}^{(l)} \leftarrow \alpha_t^{(l)} \beta^{\mathcal{L}(f^{(l)}(x),y)}, \tag{3}$$

where $\beta \in (0, 1)$ is the discount rate parameter and $\mathcal{L}(f^{(l)}(x), y) \in (0, 1)$. In each iteration, the weights of the classifier are discounted according to the performance of the classifier by a factor of $\beta^{\mathcal{L}(f^{(l)}(x),y)}$. At the end of each round, the value of weight α will be renormalized such that $\sum_l \alpha_t^{(l)} = 1$. Since the shallow model tends to converge faster than the deep model, we set the smoothing parameter $S \in (0, 1)$ to balance exploration and exploitation [29] by setting a minimum weight for each classifier. The weights of α are smoothed after updating them in each iteration: $\alpha^{(l)} \leftarrow max(\alpha^{(l)}, \frac{S}{L})$.

For the learning of the parameters $\theta^{(l)}$, consider the online gradient descent approach, where the input to the l^{th} classifier is $h^{(l)}$, which is similar to the update of the output layer weights in the original feedforward network and can be expressed as:

$$\begin{aligned} \theta_{t+1}^{(l)} &\leftarrow \theta_t^{(l)} - \eta \nabla_{\theta_t^{(l)}} \mathcal{L}(F(x_t, y_t)) \\ &= \theta_t^{(l)} - \eta \alpha^{(l)} \nabla_{\theta_t^{(l)}} \mathcal{L}(f^{(l)}, y_t). \end{aligned} \tag{4}$$

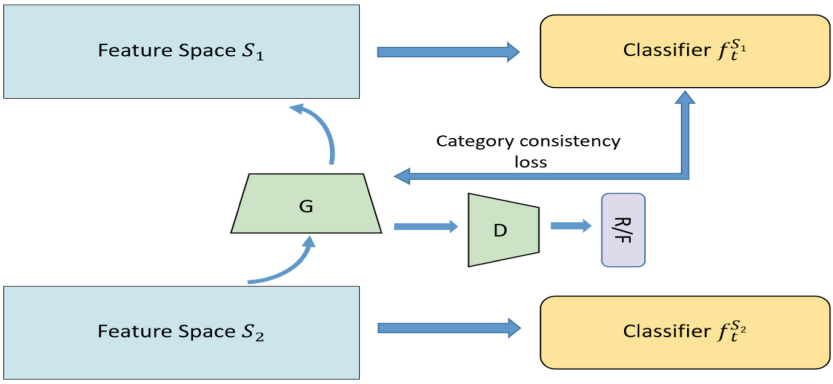


Fig. 3. In the evolving stage, the generator is used to generate S_1 from S_2 while imposing an additional loss of category consistency.

For the learning of the parameters $W^{(l)}$, where the error derivatives are back-propagated from the output layer, which is different from the previous back-propagation. Here, the error derivatives are back-propagated from each classifier $f^{(l)}$. Therefore, using the adaptive loss function $\mathcal{L}$ and applying it to the OGD, the update of $W^{(l)}$ can be expressed as:

$$W_{t+1}^{(l)} \leftarrow W_t^{(l)} - \eta \sum_{j=l}^{L} \alpha^{(j)} \nabla_{W^{(l)}} \mathcal{L}(f^{(j)}, y_t), \tag{5}$$

where η is the learning rate. Here, the summation (gradient term) starts from $j = l$ because only the deeper classifiers rely on $W^{(l)}$ for prediction. Hedge enjoys the regret of $R_T \leq \sqrt{TlnN}$, where N represents the depth of the network in our method.

4.2 Feature Space Generation Strategy

In the post-evolving stage, the old feature space disappears. To be able to use the knowledge learned in the old model, we used a generative model to recover the vanished features. The generation model GAN [30] contains two networks: a generator and a discriminator. Where the generator is trained to generate samples and the discriminator is trained to discriminate between the samples generated by the generator. The competing strategy makes both networks continuously improve. Inspired by the idea of conditional GAN [30,31], we propose a new generative network to establish the connection between two feature spaces in the evolving stage. The framework of our model is shown in Fig. 3.

An important observation is that in the evolving stage, the two feature spaces coexist and the corresponding instance labels are consistent. In wearable activity recognition, the activity observed by the sensor at the same time is always the same. Our feature space generation strategy consists of a generator G and

a discriminator D. The generator is used to generate instances $\widetilde{x}_t^{S_1}$ in the corresponding S_1 feature space from instances arriving in the S_2 feature space: $\widetilde{x}_t^{S_1} = G(x_t^{S_2})$. The discriminator is used to discriminate the instances in the generated S_1 feature space. In addition, to ensure the generated S_1 instances get the same predicted labels as the original instances, we additionally impose a category consistency loss for the generator to ensure the consistency of the classification results.

At this point, our optimization goal is:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_C(G) \tag{6}$$

$$\mathcal{L}_C(G) = -\frac{1}{N} \sum_j \sum_{c=1}^{M} y_{ic} log(P_{ic}), \tag{7}$$

where $\mathcal{L}_C(G)$ is the category consistency loss, M represents the number of categories. y_{ic} is the true category of the symbolic function sample i equal to c takes 1, otherwise takes 0. P_{ic} is the probability that the observed sample belongs to category c.

4.3 Prediction with Weighted Combination

During the T_2 time span, the old feature space S_1 disappears. Since the evolving stage B is very short, it is unlikely that the model $f_t^{S_2}$ learned on the new features will achieve a good convergence. Relying on $f_t^{S_2}$ alone to predict instances in S_2 would lead to great regret.

In order to be able to benefit from the recovered feature space, we utilize the model $f_t^{S_1}$ trained on the old feature space. Based on the generative model learned in Sect. 4.2, we reconstruct the old features from the new features. When $f_t^{S_2}$ training instances are insufficient, the predictions of the two models are weighted and combined to produce accurate predictions. Specifically, we train the model $f_t^{S_2}$ in time period T_2 using instances received in the new feature space; at the same time, the mapping relations learned in the evolving stage are utilized to recover the data in the old feature space, following which we continue to update the learned model $f_t^{S_1}$. Finally, the predictions of the two models are ensembled and represented as follows:

$$\hat{y}_t = p \cdot f_t^{S_1}(\widetilde{x}_t^{S_1}) + (1-p) f_t^{S_2}(x_t^{S_2}), \tag{8}$$

where p is the weight of the classifier $f_t^{S_1}$ learned in the feature space S_1.

5 Experiment

In this section, we compare the proposed OLFES algorithm with four other state-of-the-art methods on six datasets involving two domains. We describe the experimental setup in detail, including the datasets, the comparison methods, and the evaluation metrics, in Sect. 5.1. Section 5.2 shows the results of the comparison between OLFES and the other methods, along with the corresponding conclusions.

Table 1. The statistical information of the dataset. T_1 represents the number of instances in the entire feature space S_1, B represents the number of instances in the evolving stage, and T_2 represents the number of instances in the post-evolving stage where only S_2 exists. d_1 and d_2 represent the dimensions of the two feature spaces, respectively, and c represents the number of categories.

Datasets	T_1	B	T_2	d_1	d_2	c
Magic	19,019	1,919	17,100	10	30	2
Adult	32,559	3,559	29,000	14	30	2
Dry Bean	13,611	1,400	12,211	16	32	7
DSADS	130,000	15,000	115,000	18	27	19
PAMAP2	200,000	20,000	180,000	12	24	12
HHAR	200,000	15,000	185,000	6	6	6

5.1 Experimental Setup

Datasets. To evaluate the performance of OLFES, we conducted comparative experiments on three UCI data science dataset as well as three UCI activity recognition datasets. For the UCI activity recognition dataset, we use the collected raw sensor data for training. Table 1 shows the details of these public datasets.

- Magic [32]: UCI data science dataset. Magic is MC generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. This dataset initially has only one feature space S_1, and we artificially created a new feature space S_2 by means of a random Gaussian matrix. These two feature spaces were concatenated in the manner showed in Fig. 1 to simulate a feature evolution scenario.
- Adult [33]: UCI data science dataset. Adult was extracted from the 1994 Census database. Prediction task is to determine whether a person makes over 50K a year. The data set is processed in the same way as above and cascaded as in Fig. 1.
- Dry Bean [34]: UCI data science dataset. Dry Bean contains 13,611 images of seven different registered dry beans with 16-dimensional features such as area and perimeter extracted by a computer vision system. Process in the same manner and connect in series as shown in Fig. 1.
- DSADS [35]: UCI activity recognition dataset. DSADS consists of 19 daily and physical activities performed by eight subjects. Three types of sensors (accelerometer, gyroscope, and magnetometer) were placed at five different positions of the body. To simulate the feature evolution scenario, we divided the sensors of five body positions into two parts. The first part contains sensor data from two positions, represents the feature space S_1; the second part contains sensor data from the remained three positions, represents the feature space S_2.

- PAMAP2 [36]: UCI activity recognition dataset. PAMAP2 consists of 18 different physical activities performed by 9 subjects. Each subject wore 3 inertial measurement units (IMU) and a heart rate monitor. Some subjects were missing 6 optional activities, we selected only activities from the 12 protocols. As there are many missing values in the data collected by the heart rate monitor, we only use the data collected by the IMU in our experiments. Each IMU consists of two accelerometers, one magnetometer and one gyroscope. We divided the three IMUs into two parts. The first part contains the data of one IMU, which represents the feature space S_1; the second part contains the data of the remained two parts IMU, which represents the feature space S_2.
- HHAR [37]: UCI activity recognition datasets. The HHAR consisted of 6 different daily activities performed by 9 subjects. Each subject wore two different devices (smartphone and smartwatch), each of which contains two embedded sensors, an accelerometer and a gyroscope. We divided the two devices into two parts. The first part contains data from smartphones, represents the feature space S_1; the second part contains data from smartwatches, represents the feature space S_2.

Comparison Methods. We compare OLFES with four state-of-the-art methods. The details of the comparison methods are listed below.

- FESL-C [3]: Pioneer of feature evolvable streams. Use a shallow learner and employ a linear mapping function to learn the mapping relationship. The final prediction is a weighted combination of the two models learned on the old and new feature spaces.
- FESL-S [3]: Instead of combining old and new models weighted for prediction as in FESL-C, the best single model is utilized for prediction.
- OLD^{3}S [11]: The most advanced algorithm available. For the first time, online deep learning is combined with variational self-encoder mapping. The final prediction is a weighted combination of the old and new models.
- OLFES-N: A variant of OLFES. For the vanished feature space, we do not do anything with it and only use the new model trained on the new feature space for classification.

Evaluation Metrics. Online Classification Accuracy (OCA) as well as Average Cumulative Regret (ACR) [11] are frequently used measures in online learning methods. OCA dynamically evaluates the accuracy of the classifier f_t at round t and validates it on the most recent K instances:

$$OCA\left(f_t\right) = \frac{1}{K} \sum_{i=t-K}^{t} \left[y_i = f_t(x_i)\right]. \tag{9}$$

Table 2. Comparison results of the average cumulative regret (ACR) over the six datasets, with the symbol • indicating the optimal result.

Datasets	FESL-C	FESL-S	OLD^{3}S	OLFES-N	OLFES
Magic	.0473±.014	.0448±.013	.0414±.020•	.0504±.024	.0456±.015
Adult	.0500±.013	.0420±.017	.0514±.013	.0493±.021	.0346±.020•
Dry Bean	.1313±.018	.1132±.016	.0576±.019	.0657±.018	.0495±.014•
DSADS	.1593±.039	.1590±.036	.0942±.026	.0826±.035	.0589±.014•
PAMAP2	.1182±.012	.1235±.011	.1075±.009	.0902±.008	.0877±.006•
HHAR	.1137±.012	.1085±.012	.0658±.010	.0618±.009	.0569±.008•

ACR evaluates the degree of regret of the online classifier compared to the optimal f^* by accumulating the OCA difference between f_t and f^* in the T round. In general, a lower ACR indicates better performance of an online algorithm. ACR can be expressed as:

$$ACR = \frac{1}{T}\sum_{t=1}^{T}[\max_{f^*} OCA(f^*) - OCA(f_t)]. \tag{10}$$

5.2 Experimental Results

In this section, we show the results of the proposed OLFES algorithm compared with the other four methods. Table 2 gives the ACR of all the compared methods on the datasets of Table 1. The OCA of all the compared methods on the datasets of Table 1 is given in Fig. 4. Results for all methods are averaged over 10 independent runs.

From Table 2, we can observe that our OLFES method can achieve the smallest ACR on the Adult, Dry Bean, DSADS, PAMAP2, and HHAR datasets. And on the Magic dataset, OLD^{3}S achieved the smallest ACR. The reason why both variants of FESL will achieve the highest ACR in most cases compared to the nonlinear mapping methods is that the linear mapping methods reconstruct the feature space with a large error compared to the original feature space, leading to a decrease in accuracy.

OLFES has a higher OCA in the non-evolving stages than FESL-C, FESL-S and OLD^{3}S methods in Fig. 4. The reason for this is that the two variants of FESL learn linear shallow models and cannot learn nonlinear functions in complex application scenarios. Although OLD^{3}S uses the HBP method to update the parameters of the network online, it employs a variational self-encoder mapping into a latent space before the data stream is fed to the classifier. Since the data stream is complex and variable, the self-encoder produces a certain error in the mapping, thus leading to a gap in the OCA.

At the end of the evolving stage, OLFES does not lead to a sudden drop in OCA performance, indicating that our feature space generation strategy is

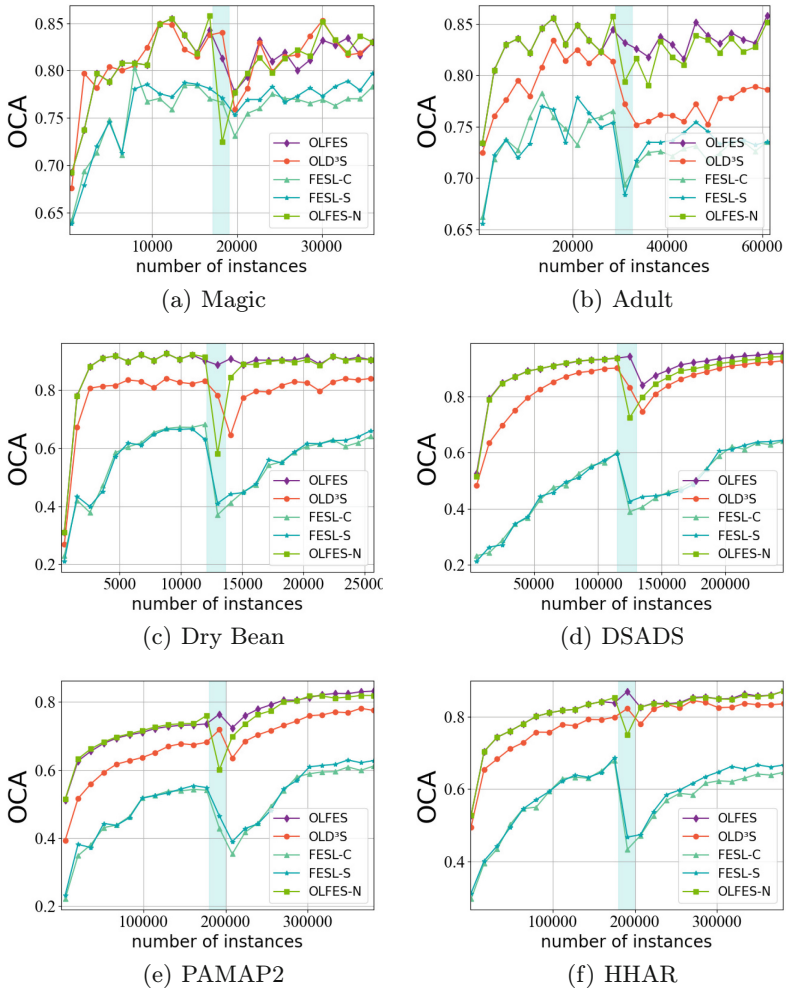


Fig. 4. Compare the online accuracy of five methods on six datasets. For all six subplots, the horizontal axis indicates the number of instances and the vertical axis indicates the online classification accuracy.

superior to the linear mapping and the variational self-encoder mapping methods. In terms of the degree of decline, the two variants of FESL have the largest gap between before and after evolution, and the OLD^{3}S method is significantly better than the method using linearity. The reason for this is that the complex relationship between two feature spaces is often nonlinear and cannot be approximated by a simple linear mapping.

In the comparison results with OLFES-N, our method does not lead to a steep drop in OCA in the case of evolution of the old feature space, which has a clear advantage. The reason is that the old features generated through the

proposed mapping method positively help the learning of the current features, indicating that our feature space generation strategy is reasonable and effective.

6 Conclusion

In this paper, a novel Online Learning method for Feature Evolvable Streams (OLFES) is proposed for the feature evolution problem due to the appearance and disappearance of sensors in wearable activity recognition. The method can not only learns the optimal depth of the model from the data stream, but also reconstructs the old features from the new ones after the old have disappeared, allowing the model to use the information of the old and new features and make accurate online predictions through a weighted combination. Comparative experiments with four other state-of-the-art methods on three data science datasets and three activity recognition datasets demonstrate the effectiveness of OLFES.

However, only the appearance and disappearance of sensors of the same type and frequency are considered in our method. In real life, there are many sensors of different types and frequencies. Compared with a single type of sensor, multiple types of sensors often bring complementary information that can help to further improve the performance of the model. In our future research, we will plan how to incorporate multimodal sensors into the paradigm of feature evolvable learning to further enhance the generalizability of the model.

Acknowledgement. This study is supported by the National Natural Science Foundation of China (Grant No. 62002187, No. 62276156 and No. 61976127), Qilu University of Technology (Shandong Academy of Sciences) Talent Scientific Research Project (No. 2023RCKY127 and No. 2023RCKY128), Technology Small and Medium Enterprises Innovation Capability Improvement Project of Shandong Province (No. 2023TSGC0279).

References

1. Antunes, R.S., et al.: A survey of sensors in healthcare workflow monitoring. ACM Comput. Surv. (CSUR) **51**(2), 1–37 (2018)
2. Wang, Y., Cang, S., Yu, H.: A survey on wearable sensor modality centred human activity recognition in health care. Expert Syst. Appl. **137**, 167–190 (2019)
3. Hou, B.J., Zhang, L., Zhou, Z.H.: Learning with feature evolvable streams. IEEE Trans. Knowl. Data Eng. **33**(6), 2602–2615 (2019)
4. He, Y., Dong, J., Hou, B.J., Wang, Y., Wang, F.: Online learning in variable feature spaces with mixed data. In: 2021 IEEE International Conference on Data Mining (ICDM), pp. 181–190. IEEE (2021)
5. Schreckenberger, C., He, Y., Lüdtke, S., Bartelt, C., Stuckenschmidt, H.: Online random feature forests for learning in varying feature spaces. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 4587–4595 (2023)
6. Gu, S., Qian, Y., Hou, C.: Learning with incremental instances and features. IEEE Trans. Neural Netw. Learn. Syst. (2023)

7. Nie, X., Gao, R., Wang, R., Xiang, D.: Online multiview deep forest for remote sensing image classification via data fusion. IEEE Geosci. Remote Sens. Lett. **18**(8), 1456–1460 (2020)
8. He, Y., Wu, B., Wu, D., Beyazit, E., Chen, S., Wu, X.: Toward mining capricious data streams: a generative approach. IEEE Trans. Neural Netw. Learn. Syst. **32**(3), 1228–1240 (2020)
9. Liu, Y., Li, W., Gao, Y.: Confidence-weighted learning for feature evolution. Int. J. Softw. Inf. **12**(2) (2022)
10. Zheng, T., Wang, X., Chen, Y., Yu, F., Li, J.: Feature evolvable learning with image streams. In: Intelligent Data Analysis (Preprint), pp. 1–17 (2022)
11. Lian, H., Atwood, J.S., Hou, B.J., Wu, J., He, Y.: Online deep learning from doubly-streaming data. In: Proceedings of the 30th ACM International Conference on Multimedia, pp. 3185–3194 (2022)
12. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) Pervasive 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24646-6_1
13. Zhao, H., Liu, Z.: Human action recognition based on non-linear svm decision tree. J. Comput. Inf. Syst. **7**(7), 2461–2468 (2011)
14. Zhao, Z., Chen, Y., Liu, J., Shen, Z., Liu, M.: Cross-people mobile-phone based activity recognition. In: Twenty-Second International Joint Conference on Artificial Intelligence. Citeseer (2011)
15. Ronao, C.A., Cho, S.-B.: Deep convolutional neural networks for human activity recognition with smartphone sensors. In: Arik, S., Huang, T., Lai, W.K., Liu, Q. (eds.) ICONIP 2015. LNCS, vol. 9492, pp. 46–53. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26561-2_6
16. Zhou, X., Guo, J., Wang, S.: Motion recognition by using a stacked autoencoder-based deep learning algorithm with smart phones. In: Xu, K., Zhu, H. (eds.) WASA 2015. LNCS, vol. 9204, pp. 778–787. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21837-3_76
17. Vollmer, C., Gross, H.-M., Eggert, J.P.: Learning features for activity recognition with shift-invariant sparse coding. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E.P., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 367–374. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40728-4_46
18. Zeng, M., Nguyen, L.T., Yu, B., Mengshoel, O.J., Zhu, J., Wu, P., Zhang, J.: Convolutional neural networks for human activity recognition using mobile sensors. In: 6th International Conference on Mobile Computing, Applications and Services, pp. 197–205. IEEE (2014)
19. Wang, J., Long, Q., Liu, K., Xie, Y., et al.: Human action recognition on cellphone using compositional BIDIR-LSTM-CNN networks. In: 2019 International Conference on Computer, Network, Communication and Information Systems (CNCI 2019), pp. 687–692. Atlantis Press (2019)
20. Hammerla, N.Y., Halloran, S., Plötz, T.: Deep, convolutional, and recurrent models for human activity recognition using wearables. arXiv preprint arXiv:1604.08880 (2016)
21. Zhou, Z.H.: Open-environment machine learning. Natl. Sci. Rev. **9**(8), nwac123 (2022)
22. Guan, S.U., Liu, J.: Incremental neural network training with an increasing input dimension. J. Intell. Syst. **13**(1), 71–94 (2004)

23. Wang, T., Liu, F., Zhou, W., Zhu, X., Guan, S.U.: Neural incremental attribute learning based on principal component analysis. In: 2016 IEEE International Conference on Big Data Analysis (ICBDA), pp. 1–5. IEEE (2016)
24. Hu, C., Chen, Y., Peng, X., Yu, H., Gao, C., Hu, L.: A novel feature incremental learning method for sensor-based activity recognition. IEEE Trans. Knowl. Data Eng. **31**(6), 1038–1050 (2018)
25. Zhang, Y., Chen, Y., Yu, H., Yang, X., Sun, R., Zeng, B.: A feature adaptive learning method for high-density semg-based gesture recognition. Proc. ACM Interact. Mobile Wearable Ubiq. Technol. **5**(1), 1–26 (2021)
26. Zhang, Z.Y., Zhao, P., Jiang, Y., Zhou, Z.H.: Learning with feature and distribution evolvable streams. In: International Conference on Machine Learning, pp. 11317–11327. PMLR (2020)
27. Hou, C., Zhou, Z.H.: One-pass learning with incremental and decremental features. IEEE Trans. Pattern Anal. Mach. Intell. **40**(11), 2776–2792 (2017)
28. Sahoo, D., Pham, Q., Lu, J., Hoi, S.C.: Online deep learning: learning deep neural networks on the fly. arXiv preprint arXiv:1711.03705 (2017)
29. Hoi, S.C., Jin, R., Zhao, P., Yang, T.: Online multiple kernel classification. Mach. Learn. **90**, 289–316 (2013)
30. Goodfellow, I., et al.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
31. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 (2017)
32. Bock, R.: MAGIC Gamma Telescope. UCI Machine Learning Repository (2007)
33. Becker, B., Kohavi, R.: Adult. UCI Machine Learning Repository (1996)
34. Dry Bean Dataset: UCI Machine Learning Repository (2020)
35. Barshan, B., Altun, K.: Daily and Sports Activities. UCI Machine Learning Repository (2013)
36. Reiss, A.: PAMAP2 Physical Activity Monitoring. UCI Machine Learning Repository (2012)
37. Blunck, Henrik, B.S.P.T.K.M., Dey, A.: Heterogeneity Activity Recognition. UCI Machine Learning Repository (2015). https://doi.org/10.24432/C5689X



Let's Vibrate with Vibration: Augmenting Structural Engineering with Low-Cost Vibration Sensing

Masfiqur Rahaman[1], Md. Nazmul Hasan Sakib[1(✉)], Nafisa Islam[2], Saiful Islam Salim[1], Uday Kamal[3], Raihan Rasheed[1], and A. B. M. Alim Al Islam[1]

[1] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
0422052005@grad.cse.buet.ac.bd
[2] Department of Civil Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
[3] Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

Abstract. Using low-cost piezoelectric sensors to sense real structural vibration exhibits great potential in augmenting structural engineering, which is yet to be explored in the literature to the best of our knowledge. An example of such unexplored augmentation includes classifying diverse structures (such as buildings, flyovers, foot over-bridge, etc.). To explore these aspects, we develop a low-cost piezoelectric sensor-based vibration sensing system aiming to collect real vibration data from diversified civil structures remotely. We dig into our collected sensed data to classify five different types of structures through rigorous statistical and machine learning-based analyses. Furthermore, we design a lightweight Convolutional Neural Network architecture and perform necessary hyperparameter tuning to achieve better accuracy in classification. Our analyses achieve a classification accuracy of up to 97% with an F1 score of 0.97.

Keywords: Structure classification · Structural vibration · Piezoelectric sensor · Time domain signal · Sensor systems

1 Introduction

In the domain of Structural Engineering, vibration pattern of civil structures exhibits applications in diversified areas such as designing architectures of the structures, Structural Health Monitoring, occupancy estimation, etc., [12,19]. In the case of designing architectures, every structure follows specific vibration criteria that should be fulfilled when designing the structure. Concrete structures such as buildings, concrete foot overbridges, etc., are generally considered

M. Rahaman and N. H. Sakib—These authors contributed equally to this work.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 408–425, 2024.
https://doi.org/10.1007/978-3-031-63989-0_21

to generate less vibration. On the other hand, steel foot overbridges, suspension bridges, etc., generate more vibration. These characteristics are very crucial when designing the architecture of a structure as misinterpretation of any of the characteristics or even ignorance of any of them may result in possible damage or structural health hazard in the future.

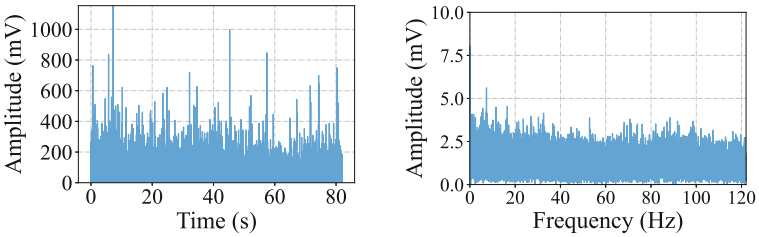
Due to the importance mentioned above, the dynamics of structural vibration have been investigated by several recent research studies [4,16,23,24]. However, these approaches lack some important considerations. Most of the existing studies consider a single structure, i.e., bridge, building, rail line, wind turbine, machine structures, etc., [3,14,21,26]. However, a study covering diverse civil structures is yet to be explored in the literature to the best of our knowledge. There exist research studies on implications of vibration generated by civil structures, e.g., structure and machine fault classification, engine classification, human identification, etc., from the pattern of vibration [1,3,11,16]. However, classifying diverse structures from their vibration patterns is yet to be explored in literature. The relation only pertains to *frequency* domain specially applicable for vibration data collected using high-cost sensors [19]. It is yet to be explored how the relationship would be in the case of vibration data collected using low-cost sensors. Moreover, it is important to know whether the relationship in the case of vibration data collected using low-cost sensors would work in the conventional *frequency* domain or it would get shifted to any other domain (such as the *time* domain).

Keeping all these considerations in mind, in this paper, we present a novel approach to classifying diversified civil structures based on their generated vibration. To do so, first, we devise and develop a low-cost piezoelectric vibration sensing module. Using the vibration sensing module, we build a diverse dataset by sensing vibration from five different types of civil structures after a year-long on-field data collection. We show that there is a significant difference in vibration generated by different types of civil structures and the structures can be classified based on their generated vibration patterns. To the best of our knowledge, this finding is yet to be revealed in the literature.

The overall methodology of our study includes developing a customized sensing system, deploying the sensing system in a real environment, sending vibration data to the cloud and storing the vibration data there, visualizing the data on a real-time dashboard, performing statistical and machine learning-based analyses for structure classification Here, for our machine learning-based classification, we perform feature selection according to correlation and regression. Further, for Deep learning-based analysis, we perform hyperparameter tuning, i.e., tuning batch size, kernel size, number of filters, and activation function.

Based on our study, we make the following set of contributions in this paper.

– We design and develop a low-cost vibration sensing module using the piezoelectric sensor. Using the sensing module, we collect real vibration data from 12 different civil structures having five different categories through a year-long on-field study.



(a) Time domain (b) Frequency domain

Fig. 1. No significant frequency component after FFT

– We classify the five categories of civil structures based on their generated vibration through statistical and machine learning-based analyses. Further, to achieve better accuracy, we develop a customized Deep Neural Network and utilize it for the classification task.

Our contribution in classifying structures from their *time* domain vibration may contribute in the future in the field of SHM through classifying faults in structures. In this study we have not devised any SHM solution, rather proposed a new aspect of structure classification which may contribute to the *time* domain signal processing in SHM.

2 Related Work

In this section, we discuss existing studies in the field of vibration and its applications.

2.1 Vibration Source Detection

There have been several studies on detecting the source of vibration through different sensor-based data analytics [11,16,23,24]. For example, Kucukbay et al., [16] classified human, vehicle, and animal-induced acoustic and vibration data. According to the type of vibration data, their proposed system triggers a camera event as an action for detecting intruders (human or vehicle). Besides, Rivas et al., [24] proposed a wireless sensor network on the road that can precisely detect the presence of vehicles. Their proposed system can calculate vehicle speed and travel direction from Accelerometer data. Sigmund et al., [26] showed that vibration sensed from distant vehicles may be used to help in identifying key vehicle features such as engine type, engine speed, and the number of cylinders.

Garrity et al., [11] classified the category of vehicles in an airport by distinguishing flight landing and vehicle movement on the runway. They developed an automated real-time monitoring and alert system that integrates GUI-based software to handle data collection and analysis. For data collection, they used a 2-axis Accelerometer.

Berlin et al., [4] classified train type and estimated train length from data accumulated by 3D MEMS Accelerometer. They studied Europe's busiest railroad sections and collected vibration patterns of 186 trains. They classified them into six categories using various methods. Chakraborty et al., [8] proposed a WSN-based automated system that can sense vibration induced by a running train from a long distance and can detect if there is any missing rail block on the track so that it can inform the train driver about a possible accident ahead.

2.2 Structural Damage Detection

Identifying structural damage is another important study in Structural Engineering. There have been several studies in recent years regarding vibration-based Structural Health Monitoring (SHM) [8,9,18,31]. For example, Lee et al., [18] presented an effective method for damage estimation of steel girder bridges using ambient vibration data. They used the frequency domain decomposition technique to identify modal perimeters.

Goyal et al., [12] presented the most used signal processing techniques in SHM such as time series models, wavelet transform, and HHT. Magalhaes et al., [21] installed a dynamic monitoring system in a concrete arch bridge at the city of Porto, in Portugal. They proposed a strategy to minimize the effects of environmental and operational factors on the bridge's natural frequencies, enabling the identification of structural anomalies.

Zonzini et al., [33] proposed a sensor network that can be used with either MEMS accelerometers or piezoelectric sensors to extract modal parameters of structures. Testoni et al., [30] proposed a sensor network based on low-power, low-cost, and lightweight MEMS sensor nodes to measure the tilt angles of structures.

2.3 Machine Fault Detection

Another important field of study is classifying faults in machine structures from their vibration characteristics. There have been many studies regarding such fault detection and classification [1,3,5,7,13,14]. Joshuva et al., [14] developed a data model for a multi-class wind turbine blade fault diagnosis. From acquired Accelerometer data, they developed several models using data modeling techniques. Ahmed et al., [1] presented an engine fault detection and classification technique using vibration data. They built a four-stroke gasoline engine for experimentation. Their proposed fault diagnostic system can detect known engine faults with various degrees of severity.

From the study of existing literature, we find studies focusing on vibration source detection, structural damage detection, and machine fault detection. However, classifying structures from their vibration characteristics is still unexplored in the literature. Nonetheless, existing studies are mostly based on high-cost sensors and focused on the *frequency* domain. Thus, it has to be investigated whether we can still work on the *frequency* domain or not while using the low-cost sensors.

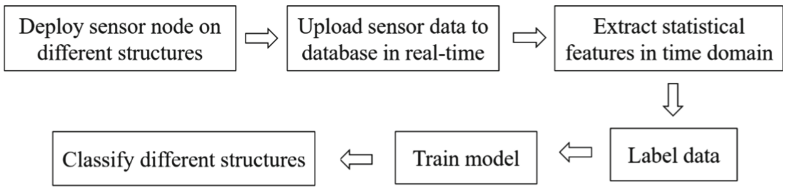


Fig. 2. Illustration of our proposed methodology on classifying structures.

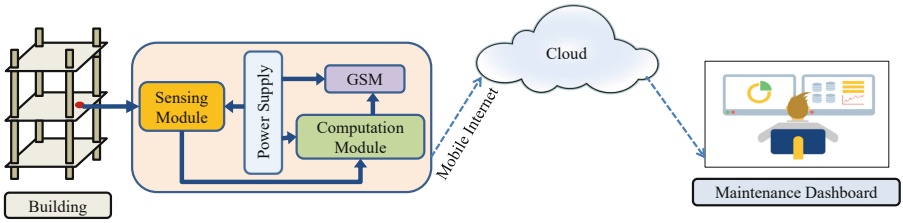


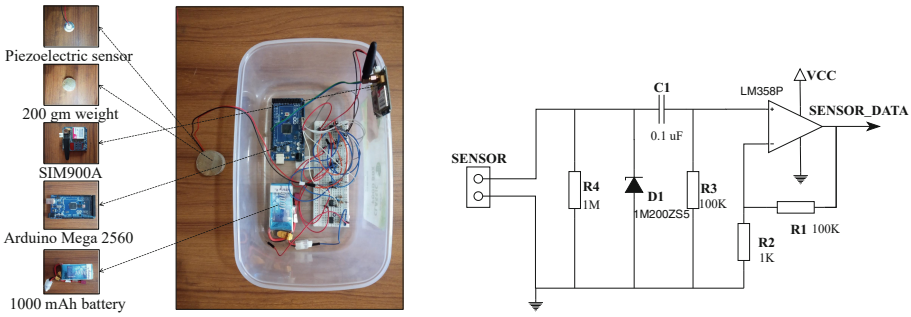
Fig. 3. Our proposed system architecture: sensing, communication, computation, power supply, and maintenance dashboard

3 Methodology

In this study, first, we build the sensing module and deploy it on the surfaces of building floors, flyover/overbridge spans, and rail-block to collect vibration data. We collect and store collected vibration data in a database in real time. Then, based on the collected vibration data, we formulate the problem of identifying the source of a structure as a classification problem and attempt to solve it. A brief overview of our proposed methodology is illustrated in Fig. 2.

The most straightforward approach to classify structures involves directly performing the Fast Fourier Transform on the vibration signal, and then looking for the fundamental *frequency* component for the signal. So, we first remove DC components from the time domain signal as shown in Fig. 1a. However, as shown in Fig. 1b, after FFT, we do not observe any obvious *frequency* component. As a result, the classical approach of exploring the fundamental frequency of the structure fails in the case of low-cost piezoelectric sensors. Also, we can not address the comparison between the response of the accelerometer and piezoelectric sensor as existing studies focused on the *frequency* responses of the accelerometer and we can only analyze the time domain response of piezoelectric sensors [12,19,21].

Hence, we move forward to time-domain analysis. In the time domain, we first extract different statistical features from raw data points. We label the data according to their sources. In our study, we explore five different sources covering building, flyover, steel overbridge, concrete overbridge, and rail line. We collect vibration data from 12 different locations in Dhaka city covering the above-mentioned five classes of structures. For flyover, we collect data for multiple spans. In the case of a building, we collect data from every floor. Subsequently, we train several classification models to classify each type of structure. We present detailed results and findings in Sects. 6 and 7.



(a) Our hardware setup (b) Circuit diagram of Sensing module

Fig. 4. Hardware components and circuit diagram of sensing module

4 Proposed Sensing System

tended tasks, we design and develop a customized sensing system. Here, we use low-cost components to make sure that the whole system remains low-cost in nature. The main components of our system include a sensing module, computational module, communication module, power supply module, and a real-time dashboard. Figure 3 shows the architecture of our proposed system and Fig. 4a shows the hardware setup. We elaborate on each of the components below.

Sensing Module: We use a low-cost piezoelectric disc [28] to sense the ambient vibration of structures. We use a 200 gm weight bar on top of the piezo sensor to fix the piezo disk and get a stable signal as shown in Fig. 4a. The total average weight of our system is around 250-300 gm. The raw analog signal collected by the piezoelectric disc is first amplified through an amplification circuit as shown in Fig. 4b. We choose the LM358P as the operational amplifier which is a low-power dual operational amplifier. The natural *frequency* of a bridge and other concrete-made structures varies in the range of 2–4 Hz, but values 0–14 Hz have also been reported [2]. LM358P's cutoff frequency is 200 Hz which is favorable considering the input signals' frequency response.

The amplification factor of the amplifier circuit is 100. We have also tried amplification factors of 200, 500, and 1000. However, for some structural vibrations, the signal cuts at the analog value of 1023 for an amplification factor greater than 100. Also, the more the amplification factor, the more the power consumption. That is why we chose the amplification factor of 100. Then, we feed the amplified signal to a 10-bit analog-to-digital converter (ADC) on an Arduino Mega [27] whose range is 0 to 5 V having the maximum sampling frequency of 9615 Hz which is greater than the operational amplifier's output signal (200 Hz). Thus, the choice of LM358P supports both the input signals' frequency response and the sampling frequency of ADC.

Computational Module: We use the Arduino Mega 2560 as our computational module. It takes sensed data from the sensing module at an interval of eight

Table 1. Cost analysis of necessary hardware equipment

Component name	Model name	Quantity	Unit price (USD)
Piezoelectric sensor	7BB-20-6L0	1	1
Amplifier	LM358P	1	6
Resistor	1M,1K,100K	4	0.25
Zener Diode	5 V	1	0.5
Capacitor	0.1 uF	1	0.5
Microcontroller	Arduino Mega2560	1	14
GSM Module	SIM900a	1	16
Power supply	Polymer Lithium Ion Battery - 1000mAh	1	10
Total price	50 USD per node		

seconds. Subsequently, it determines 12 statistical features from the captured time-series data. The statistical features are mean, median, mode, standard deviation, max, min, rms, the total number of peaks, average of peak values, skewness, kurtosis, and crest factor.

Communication Module: We use SIM900A [29], which is a GSM-based device, to send statistics of our sensed data to the server in real-time. The use of SIM900A gives robustness to our system by providing network support outside the home or office where WiFi or broadband is not available. This is why we can deploy our sensing module at diversified places covering the structures, such as flyovers, overbridges, and rail lines.

Power Supply Module: We can choose either direct or battery power options as a source of power. In room environment, we use a direct power supply unit with a 220 V to 5 V adapter. For outdoor cases such as flyovers and overbridges, we use a 5V 1000 mAh battery as the source of power.

Real-Time Dashboard: We send the statistical measurements from the collected raw data as an HTTP post request in a URL which is then stored in a database. We develop a dashboard to display the data points in real time.

Table 1 presents a breakdown of the equipment cost of our proposed sensing system. The equipment cost of the system is 50 USD per unit, which is comparable to that of a widely adopted smartphone unit. Thus, our system exhibits the potential to be a ubiquitous solution.

5 System Deployment

We deploy our sensing module in 12 different locations in Dhaka city. This enables sensing from 12 structures having five different categories among them.



(a) Overbridge (b) Flyover (c) Building (d) Railline

Fig. 5. Deployment of sensor nodes on different structures

The five different categories are flyover, building, steel overbridge, concrete overbridge, and rail line. In all cases, we place the sensor on a horizontal surface to sense vertical vibration. Figure 5 shows some snapshots of such deployment. A brief overview of the position of nodes, duration of data collection, and characteristics of structures is presented in Table 2.

In the case of flyover, we consider four different spans for data collection. We choose the middle of each span to deploy our sensing module so that maximum vibration can be captured. Besides, we deploy the module on both the left and right sides of the flyover to achieve symmetry as well as diversity.

We collect data from four academic and residential buildings. In each building, data from every floor contribute to our dataset. In the case of two of the buildings, the vibration of two columns contributes to the dataset.

In the case of foot overbridge, Our dataset contains data collected from four different steel-made and one concrete-made foot overbridges. When we collect data from these structures, varying numbers of crowds: light, medium, and dense are crossing over the bridges. We collect data for at least 2 different positions on each overbridge.

We also cover rail lines. In rail lines, data from both meter gauge and broad gauge lines, contribute to the dataset. Here, the data is collected only when no train passes by. We cover crossings over rail lines where buses, cars, bikes, cycles, and people cross rail lines from one side to another.

As mentioned earlier a web server keeps all data collected by the sensor. We organize the data by location and type of structure and store them accordingly. From all structures under investigation, we collect data of a total interval of 4 h and 20 min. As we have on average 200 data points at each second, our dataset contains a summary of a total of 3 million raw data points. To be specific, our dataset contains 1,159 summary data points.

After collecting the time-series data, we extracted 12 statistical features from those data points. Table 3 shows some sample entries in our dataset.

6 Classifying Structures from Vibration Data

We perform different visualization methods, statistical analysis, and learning-based analysis over the collected data using scikit-learn version 0.23.1 [17]. The analyses help to visualize the data effectively and at the same time signify the possibility of better classification.

Table 2. Details of different structures (node positions, duration of data collection, structural demographics, etc.) where sensor module is deployed

Flyover	#Spans covered	Duration (minutes)	Deployment position
Flyover-1	5	30	On surface of road

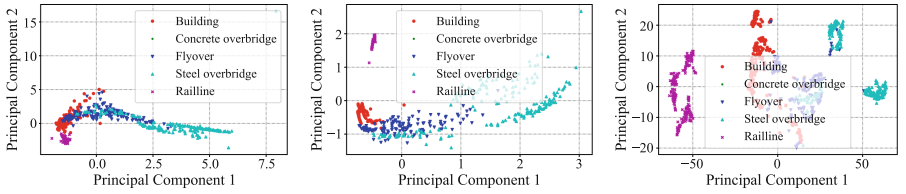
Building	Type of building	#Floors	Duration (minutes)	Deployment position
Building-1	Office	11	60	Floor, column
Building-2	Office	5	30	Floor, column
Building-3	Office	3	10	Floor
Building-4	Residential	4	20	Floor

Foot overbridge	Type of overbridge	Crowd density	Duration (minutes)	Deployment position
Overbridge-1	Steel-made	High	10	Middle of span
Overbridge-2	Steel-made	Medium	10	Middle of span
Overbridge-3	Steel-made	Low	20	Middle of span
Overbridge-4	Steel-made	High	10	Middle of span
Overbridge-5	Concrete-made	High	10	Middle of span

Railline	Line type	#Tracks	Duration (minutes)	Deployment position
Railline-1	Meter and broad gauge	2	30	Attached with steel block
Railline-2	Meter and broad gauge	2	20	Attached with steel block

Table 3. A small portion of our dataset (single row is shown from each type of structures)

Mean	Mode	Median	Standard deviation	Max	Min	RMS	Number of peaks	Average of peak values	Skewness	Kurtosis	Creast factor	Type of structure
20.16	19	21	10.38	96	0	22.68	651	26.59	1.9	10.03	4.23	Building
28.21	0	0	35.72	255	0	45.52	579	66	1.68	3.68	5.6	Flyover
62.86	5	63	1.89	66	37	62.89	498	63.77	−4.81	49.46	1.05	Railline
154.67	0	0	178.66	747	0	236.31	613	297.05	1.19	0.85	3.16	Steel overbridge
24.9	68	0	24.45	259	0	34.9	630	44.88	1.78	7.29	7.42	Concrete overbridge



(a) Principle component analysis (PCA)

(b) Factor analysis

(c) T-distributed stochastic neighbor embedding (t-SNE)

Fig. 6. Graphical representation of clusters formed by different structures

6.1 Visualization of Data

To better visualize the data, we use principle component analysis (PCA). This reduces feature dimension from 12 to two principal components and forms clusters of the same type of structures as shown in Fig. 6a. We also conduct T-distributed Stochastic Neighbor Embedding, and Factor Analysis for better visualization. We present the outcomes of all these analyses in Fig. 6. These figures clearly portray that there is a significant difference in the vibration of the five structures.

6.2 Correlation Between Statistical Features and the Type of Structure

We use Pearson's correlation coefficient (pearsonr() available in scipy stats package [17]) to identify- 1) how different statistical features and the type of structure are correlated with one another, and 2) whether there exists any statistically significant association ($r >= 0.4$ and $p < 0.00005$) [25]. Here, we first generate the correlation matrix, and then we determine the prediction values of the regression matrix. In a correlation matrix, the feature having the highest absolute correlation coefficient value is highly related to the type of structure. On the other hand, in a regression matrix, the feature with the least prediction value is highly significant to the type of structure.

Table 4 shows the correlation and prediction values for all the features with the different types of structures. The table demonstrates that RMS, an average of peaks, mean, standard deviation, and max exhibits strong correlation values. The same features also exhibit the lowest prediction values. Thus, we can deduce that RMS, average of peaks, mean, standard deviation, and max are highly significant features in terms of getting correlated. Accordingly, we conduct further analysis on classifying the type of structure using machine learning algorithms based on the selected five features.

6.3 Machine Learning for Classifying Structures

We apply several machine learning algorithms to our prepared dataset. Here, we formulate the task of predicting the type of structure from associated feature values as a classification problem where each class corresponds to one of the five different structures. The accuracy in our case corresponds to the number of correctly classified instances over the number of total test instances. We calculate different performance metrics such as precision, recall, and F-measure in this regard. We use 10-fold cross-validation for training each model. Then we conduct testing of each classifier model on unseen data points. In all cases, we maintain the ratio between the training and testing dataset as 8:2.

Table 5 presents the performance of all the classifiers under consideration. Among these classifiers, k-NN, RandomForest, and RandomTree perform the best in the metrics of accuracy, precision, recall, and F-measure. Among them, k-NN (for $k = 1$) shows 91% accuracy and outperforms others. For optimizing the value of k, here, we cross-validate the k-NN model for k out of the range

Table 4. Correlation matrix and regression matrix(p-value) of type of structure with all features

Features	Correlation value	Prediction value
Mean	0.716656053	0.000000023
Median	0.090901681	0.190540928
Mode	0.154416336	0.025589244
Standard deviation	0.584570249	0.00000163
Max	0.406880713	0.000000041
Min	0.26979996	0.000078
RMS	0.784558941	0.000000001
Number of peaks	−0.274614018	0.0000572
Average of peaks	0.756443394	0.000000017
Skewness	0.228503295	0.000875476
Kurtosis	0.123029244	0.075948388
Creast factor	0.195535324	0.004549568

Table 5. Performance matrices of different classifiers

Classifier	Accuracy(%)	Precision	Recall	F-Measure
k-NN(k=1)	91	0.92	0.91	0.91
RamdomForest	90	0.92	0.90	0.90
RandomTree	89	0.90	0.89	0.89
Bagging	84	0.88	0.84	0.84
DecisionTable	80	0.82	0.80	0.79
NaiveBayes	71	0.66	0.72	0.67

from 1 to 30 with the training dataset. Our cross-validation results show that when the value for k is 1, validation accuracy exhibits the highest value.

Figure 7 presents a normalized confusion matrix for k-NN (k = 1). Here, among five structures, the flyover gets misclassified as a building or steel overbridge several times. Also, concrete overbridge gets misclassified as building and flyover in some cases. This leads to a high false-positive rate for flyover and concrete overbridge. The cause behind this is a lower number of data points for flyover and concrete overbridge as we cover only one flyover and one concrete overbridge in our data collection phase. Another reason is the fact that both concrete overbridges and flyovers are made of concrete. Thus, there can be a similarity of vibration for these two types of structures. Building, steel overbridges, and rail lines, on the other hand, get no false positive or false negative case. This is because we have a substantial amount of data points for buildings, steel overbridges, and rail lines. Also, vibration propagates more through metal structures, and, more importantly, in a more distinctive manner. Now, as there

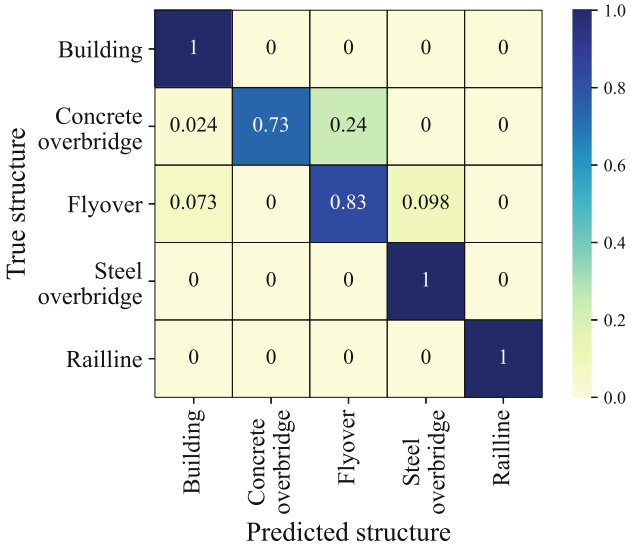


Fig. 7. Normalized confusion matrix for testing phase in machine learning based approach (k nearest neighbour)

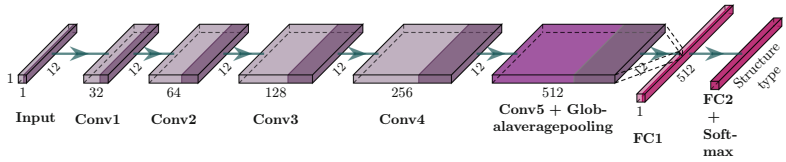
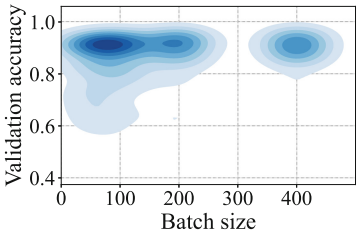


Fig. 8. Architecture of our proposed Deep Learning based model

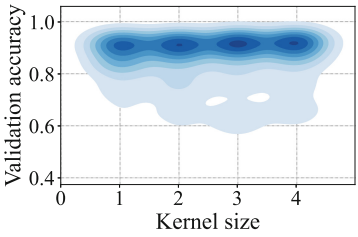
exists substantial room for further improvement, we employ Deep Learning for this purpose next.

7 Deep Learning

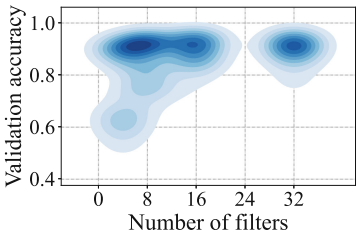
Among the 12 features in our dataset, we chose five features according to the correlation between the features and target classes. However, all models exhibit at most 90% accuracy except k-NN. Even k-NN exhibits substantial error in classifying two classes (flyover and concrete overbridge). This suggests that a more advanced feature extraction method might be required for better performance. Therefore, we employ Deep Convolutional Neural Network (CNN) which has shown excellent performance for different sensor-based classification tasks [22,32]. In this regard, we propose a customized Deep Convolutional Neural Network for our intended task. In the next subsections, we demonstrate the architecture of our designed model and explain the experimental results.



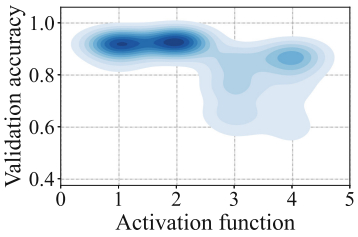
(a) Effect of batch size



(b) Effect of kernel length



(c) Effect of filter numbers



(d) Effect of activation function

Fig. 9. Kernel density estimation plot of the 256 experiments for each of the hyperparameters. In Figure (d), the numbers 1, 2, 3 and 4 indicate 'ReLU', 'ELU', 'Tanh' and 'Sigmoid' activation functions respectively.

7.1 Model Architecture

The convolution block of our model consists of five convolutional layers. The input shape to this convolution block is $n \times 1 \times 12$ where n is the batch size and the number 12 is for all the 12 features from raw vibration data. The kernel size for each convolutional layer is 3×3. To learn a rich set of features, we increase the number of filters exponentially with the depth of the layers. The number of filters at the r^{th} convolutional layer is $2^r \times q$ where $0 <= r < 5$ and the value of q is selected as 32. The convolution operations are usually followed by activation functions that introduce non-linearity in the network. In our proposed model, we use the Exponential Linear Unit (ELU) as our activation function, which is defined as:

$$f(x) = \begin{cases} x & x > 0 \\ \alpha * (e^x - 1) & x \leq 0 \end{cases} \tag{1}$$

Our choice of hyperparameters is explained in the following subsection. We use a Globalaveragepooling layer after the convolutional block to minimize the learnable parameters. Finally, we use a fully connected layer with N number of output neurons along with softmax activation function to map the N class scores to N probability values $p = [p_1, p_2, ..., p_N]$ for each class, which sums up to 1.

We present an overview of the whole architecture in Fig. 8 and Table 6.

Table 6. Network parameters

Layers.	Output Size	Kernels
Input	1×12	–
Conv1D & elu	12×32	$f = 32, K = 3, s = 1$
Conv1D & elu	12×64	$f = 64, K = 3, s = 1$
Conv1D & elu	12×128	$f = 128, K = 3, s = 1$
Conv1D & elu	12×256	$f = 256, K = 3, s = 1$
Conv1D & elu	12×512	$f = 512, K = 3, s = 1$
Globalaveragepooling1D	1×512	–
Fully connected	$1 \times N$	–

* Here f, K, s, and N represent the number of filters, kernel length, filter stride, and the number of classes respectively.

Table 7. Model performance on training phase

Training accuracy	Validation accuracy
97.7%	96.7%

Table 8. Model performance on testing phase

Testing accuracy	Precision	Recall	F-Measure
97.1%	0.97	0.97	0.97

7.2 Experimental Setup

The model hyperparameters for our network contain batch size, kernel size, number of filters, and activation function. Here, we vary the batch size to 50, 100, 200, and 400. Besides, we vary kernel length as 1, 2, 3, and 4. We also vary the number of filters (q) for the first convolution layer as 4, 8, 16, and 32.

At first, we split the dataset into 70% training set, 10% validation set, and 20% test set. We use 10-fold cross-validation to find the value of the hyperparameters. This results in a total of 10×256 experiments. Here 10 is the total number of folds. The number 256 is the number of possible combinations of hyperparameters. Some possible combinations of hyperparameters (batch size, kernel size, number of filters, activation function) are (50, 1, 4, Tanh), (50, 2, 32, ELU), (200, 2, 4, ReLU), etc. The average results of the 256 experiments over the 10 folds are shown as kernel density estimation plots in Fig. 9. It is evident that a combination of batch size 100, kernel length 3, number of filters 32, and the 'ELU' activation function achieves the highest validation accuracy. Based on these results, we set the hyperparameters in our model as (100, 3, 32, ELU). All of the experiments regarding training, testing, and hyperparameter tuning of the networks are performed in Kaggle kernel environments which provide Nvidia

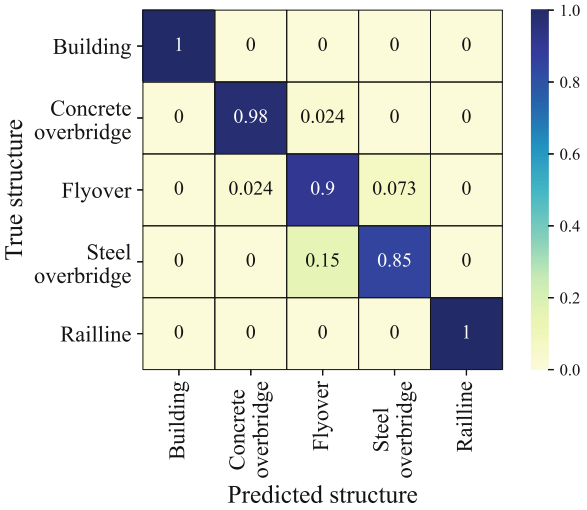


Fig. 10. Normalized confusion matrix of testing phased in deep learning based approach

K80 GPUs [20]. We write necessary codes in Python and implement the neural network models using the Keras API with TensorFlow in the back-end [6,10].

7.3 Results

We evaluate the performance of our Deep Learning-based model over the collected dataset in two stages. At first, we evaluate the performance of our model in the training phase with 10-fold cross-validation. In each fold, we train the model for 1000 epochs. We use Adam [15] as an optimizer with an initial learning rate of 10^{-2}. We also use a learning rate decay factor of 0.8 if the validation accuracy does not improve for 10 consecutive epochs. Table 7 represents the average training accuracy and validation accuracy of this experiment.

In the second stage, we evaluate our model over unseen test data, which can be of untrained buildings, rail lines, steel overbridges, and concrete overbridges. As we collect data from only one flyover, we use it for both training and testing. Among the 10 models from every 10 folds, we choose the best model having the highest validation accuracy. Then, we evaluate the model over several performance metrics, such as accuracy, precision, recall, and F1-score. Our Deep Learning-based model outperforms the best-found machine learning-based k-NN (91%) in terms of all performance metrics. Table 8 presents values of all performance metrics.

Figure 10 presents a normalized confusion matrix for the test set evaluation. Among the five structures, the flyover gets misclassified as concrete and steel overbridge for few times, though the false positive rate here is less compared to machine learning-based approach as in Fig. 7. This happens as we cover only

one flyover in our data collection phase, resulting in a relatively smaller amount of data. Besides, both concrete overbridge and flyover are made of concrete, and thus there can be a similarity of vibration between these two structures. Nonetheless, building and rail lines get no false positive or false negative case.

The reason for Deep Learning based approach performing better is that Deep Learning does not require any feature selection procedure. On the other hand, in our machine learning based approach, we select five statistical features among 12 from our dataset according to our analysis on correlation and significance. However, in our Deep Learning based approach we take all of the 12 features ignoring their correlation and significance. This helps in learning of our model significantly, and thus, in achieving a higher accuracy.

8 Conclusions

Analyzing the dynamics of vibration for diversified civil structures is little explored in literature - especially from the perspective of using low-cost vibration sensing. Therefore, in this study, we analyze the dynamics in depth by devising and utilizing a low-cost vibration sensing module. Our sensing module continuously uploads statistical features extracted from raw vibration data to a remote cloud server and we can visualize the data points through an interactive dashboard in real-time. We then explore different machine learning algorithms to classify different structures by collected vibration data, which gives an accuracy of up to 91%. We build a Deep Neural Network and tune its hyperparameters for improvements resulting in an accuracy of up to 97%.

One fundamental paradigm shift realized in our study is that We explore the *time* domain of vibration while analyzing the vibrations generated by the different civil structures, as only this domain exhibits considerable values in the case of using low-cost vibration (piezoelectric) sensors. This clearly differs from the existing research studies, which explore the *frequency* domain of vibration while analyzing the vibrations generated by civil structures, as *frequency* domain exhibits considerable values in the case of using high-cost vibration sensors.

Moving forward, there are several scopes for future studies. Examples include - (1) tuning the time series window size, which is considered as eight seconds in this paper, (2) evaluating the power consumption of the system to confirm long-term energy efficiency, (3) comparing our proposed system with existing high-cost accelerometer and geophone-based systems, and (4) exploring specific applications of the proposed sensing module in structural health monitoring.

References

1. Ahmed, R., El Sayed, M., Gadsden, S.A., Tjong, J., Habibi, S.: Automotive internal-combustion-engine fault detection and classification using artificial neural network techniques. IEEE Trans. Veh. Technol. **64**(1), 21–33 (2014)
2. Bachmann, H., et al.: Vibration problems in structures: practical guidelines. Birkhäuser (2012)

3. Barbosa, T.S., Ferreira, D.D., Pereira, D.A., Magalhães, R.R., Barbosa, B.H.: Fault detection and classification in cantilever beams through vibration signal analysis and higher-order statistics. J. Control Autom. Electr. Syst. **27**(5), 535–541 (2016). https://doi.org/10.1007/s40313-016-0255-1
4. Berlin, E., Van Laerhoven, K.: Sensor networks for railway monitoring: detecting trains from their distributed vibration footprints. In: 2013 IEEE International Conference on Distributed Computing in Sensor Systems, pp. 80–87. IEEE (2013). https://doi.org/10.1109/DCOSS.2013.38
5. Bhuiyan, M.H.R., Arafat, I.M., Rahaman, M., Toha, T.R., Alam, S.M.M.: Towards devising a vibration based machinery health monitoring system. Mater. Today Proc. **56**, 2490–2496 (2022)
6. Brain, G.: Tensorflow (2015). https://www.tensorflow.org/
7. Casoli, P., Pastori, M., Scolari, F., Rundo, M.: A vibration signal-based method for fault identification and classification in hydraulic axial piston pumps. Energies **12**(5), 953 (2019). https://doi.org/10.3390/en12050953
8. Chakraborty, T., Khan, T.A., Islam, A.A.A.: Poster: Railcop: detecting missing rail on railway using wireless sensor networks. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion, pp. 16–16 (2016). https://doi.org/10.1145/2938559.2948844
9. Chakraborty, T., et al.: A new network paradigm for low-cost and lightweight real-time communication between train and rail track to detect missing and faulty rail blocks. J. Netw. Comput. Appl. **131**, 40–54 (2019). https://doi.org/10.1016/j.jnca.2019.01.018
10. Chollet, F.: Keras (2015). https://keras.io/
11. Garrity, P., Bhattacharyya, S., Shen, C., Dawadi, D., Panja, B.: Vibration monitoring and analysis using a wireless sensor network (WSN) to classify vehicles
12. Goyal, D., Pabla, B.: The vibration monitoring methods and signal processing techniques for structural health monitoring: a review. Arch. Comput. Methods Eng. **23**(4), 585–594 (2016). https://doi.org/10.1007/s11831-015-9145-0
13. Janssens, O., et al.: Convolutional neural network based fault detection for rotating machinery. J. Sound Vib. **377**, 331–345 (2016). https://doi.org/10.1016/j.jsv.2016.05.027
14. Joshuva, A., Sugumaran, V.: Selection of a meta classifier-data model for classifying wind turbine blade fault conditions using histogram features and vibration signals: a data-mining study. Progress Ind. Ecol. Int. J. **13**(3), 232–251 (2019)
15. keras: Adam optimizer (2020). https://keras.io/api/optimizers/adam/
16. Küçükbay, S.E., Sert, M., Yazici, A.: Use of acoustic and vibration sensor data to detect objects in surveillance wireless sensor networks. In: 2017 21st International Conference on Control Systems and Computer Science (CSCS), pp. 207–212. IEEE (2017)
17. scikit learn: Machine learning in python (2020). https://scikit-learn.org/stable/
18. Lee, J.J., Yun, C.B.: Damage diagnosis of steel girder bridges using ambient vibration data. Eng. Struct. **28**(6), 912–925 (2006)
19. Li, Q., Wu, J., Liang, S., Xiao, Y., Wong, C.: Full-scale measurements and numerical evaluation of wind-induced vibration of a 63-story reinforced concrete tall building. Eng. Struct. **26**(12), 1779–1794 (2004)
20. LLC, G.: Kaggle (2010). https://www.kaggle.com/
21. Magalhães, F., Cunha, Á., Caetano, E.: Vibration based structural health monitoring of an arch bridge: from automated OMA to damage detection. Mech. Syst. Signal Process. **28**, 212–228 (2012)

22. Ravi, D., Wong, C., Lo, B., Yang, G.Z.: A deep learning approach to on-node sensor data analytics for mobile or wearable devices. IEEE J. Biomed. Health Inform. **21**(1), 56–64 (2016)
23. Richman, M.S., Deadrick, D.S.: Seismic method for vehicle detection and vehicle weight classification (2013), US Patent 8,405,524
24. Rivas, J., Wunderlich, R., Heinen, S.J.: Road vibrations as a source to detect the presence and speed of vehicles. IEEE Sens. J. **17**(2), 377–385 (2016)
25. Schober, P., Boer, C., Schwarte, L.A.: Correlation coefficients: appropriate use and interpretation. Anesthesia Analgesia **126**(5), 1763–1768 (2018). https://doi.org/10.1213/ANE.0000000000002864
26. Sigmund, K.J., Shelley, S.J., Bauer, M., Heitkamp, F.: Analysis of vehicle vibration sources for automatic differentiation between gas and diesel piston engines. In: Automatic Target Recognition XXII, vol. 8391, p. 839109. International Society for Optics and Photonics (2012)
27. Sparkfun: Arduino mega 2560. https://www.sparkfun.com/products/11061
28. Sparkfun: Piezo element. https://www.sparkfun.com/products/10293
29. Techshop: Sim900a (2017). https://www.techshopbd.com/product-categories/eval-board/2041/sim900a-kit-techshop-bangladesh
30. Testoni, N., Zonzini, F., Marzani, A., Scarponi, V., De Marchi, L.: A tilt sensor node embedding a data-fusion algorithm for vibration-based SHM. Electronics **8**(1), 45 (2019). https://doi.org/10.3390/electronics8010045, https://www.mdpi.com/2079-9292/8/1/45
31. Ubertini, F., Comanducci, G., Cavalagli, N.: Vibration-based structural health monitoring of a historic bell-tower using output-only measurements and multivariate statistical analysis. Struct. Health Monit. **15**(4), 438–457 (2016)
32. Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L.: Deep learning for sensor-based activity recognition: a survey. Pattern Recogn. Lett. **119**, 3–11 (2019)
33. Zonzini, F., Malatesta, M.M., Bogomolov, D., Testoni, N., Marzani, A., De Marchi, L.: Vibration-based SHM with upscalable and low-cost sensor networks. IEEE Trans. Instrum. Meas. **69**(10), 7990–7998 (2020). https://doi.org/10.1109/TIM.2020.2982814

Security Management

Research on Data Drift and Class Imbalance in Android Malware Detection

Zhen Liu[1], Ruoyu Wang[2(✉)], Bitao Peng[1], Changji Wang[1], and Qingqing Gan[1]

[1] School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou 510006, China

[2] Information and Network Engineering Research Center, South China University of Technology, Guangzhou 510041, China

rywang@scut.edu.cn

Abstract. In the Android ecosystem, malware detection is still a nontrivial task. Existing works have recently applied convolution neural networks (CNNs) for detecting Android malwares. However, data drift and class imbalance are still open problems in this field. The distribution of malware data may vary significantly if data are represented by unstable features, leading to data drift problems. The model may not be able to effectively detect malwares on the future data. In addition, the class imbalance may degrade a model on identifying a specific type of malwares with fewer training samples. To handle both of the two problems, this paper presents a new Android malware detection framework. Specifically, we devise a data distribution-aware feature learning framework for learning features with a stable distribution to handle data drift. We further devise a new loss function for CNN to handle the class imbalance problem. Using our loss function, this model can reinforcement learn the minority class samples and hard samples. The experimental results on the real datasets revealed that our method outperforms existing works for Android malware detection on the datasets with data drift and class imbalance problems.

Keywords: Android malware detection · data drift · class imbalance · feature learning · CNN

1 Introduction

According to statistics published by statcounter [1], Android owned a 71.96% mobile OS market in November 2022. The popular usage of Android mobile devices [2] also attracts threats from mobile malwares [3, 4]. These threats include stealing user privacy information, automatically triggering deductions, controlling the system, etc. [4], which causes massive harm to mobile users. The AV-TEST Institute reported more than 90 million new malicious apps in 2022 [6].

To protect mobile apps from being attacked, various techniques have been proposed for detecting mobile malwares [5]. Static analysis methods [5] mainly obtain string

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 429–444, 2024.
https://doi.org/10.1007/978-3-031-63989-0_22

information (such as permission, opcode, API calls, etc.) from the decompiled dex and manifest.xml. Reverse engineering techniques are usually used for decompiling. Dynamic analysis methods [7] collect data (such as network traffic) while running apps in an isolation environment. Static analysis and dynamic analysis can also be used together for malware detection [5].

Recent works have focused on utilizing CNN for training an Android malware detection model [8]. For example, Rahali et al. [9] utilized a feature learning and a CNN for training a malware detection model. The string features are transformed into images as the input of CNN. However, the app data distribution would be changed with the upgrading of apps or changing in attack behaviors when the data are represented by unstable features. The trained model may become ineffective on the samples with data shift. In addition, there are a few samples from some malware families (i.e., minority classes). The model trained with the objective of obtaining high overall accuracy may obtain high accuracy for classifying the majority class samples but low accuracy for classifying the minority class samples. This means that Android malware detection also faces the class imbalance problem. To address both of the two problems, this paper proposes a new Android malware detection framework. Our main contributions are as below.

(1) Our proposed android malware detection method is based on a data distribution-aware feature learning and a margin loss (MDFM). It mainly includes feature learning and model training stages.
(2) In the feature learning stage, we propose a feature learning algorithm named Data Distribution Aware Feature Learning (DDAFL). It learns the features with low data distribution discrepancy among the app samples and with high discriminations among classes on the target samples.
(3) In the training stage, we propose a new loss function named M-LDAM by modulating the LDAM loss function [13]. It aims at reinforcement learning the hard samples and minority class samples.
(4) To evaluate the performance of MDFM, we carry out multiple experiments on real datasets. Results show that MDFM performs better in most cases, especially on multiclass datasets with unstable data distributions.

The remainder of this paper is organized as follows. Section 2 overviews related works in Android malware detection field. Section 3 presents our proposed method for detecting Android malwares. Section 4 analyzes the results of Android malware detection. This paper is concluded in Sect. 5.

2 Related Work

2.1 Related Work on CNN-Based Malware Detection Methods

In the field of CNN-based Android malware detection, some studies convert [10, 26] the opcodes from the binary codes of Apks into images and then utilize CNNs to train a malware detection model. Ren et al. [27] proposed to apply the raw bytecodes of Apks as the input of deep neural networks, including CNN and RNN (recurrent neural network). CNN-based and RNN-based methods achieve approximately 93.4% and 95.8% accuracy respectively. Rong et al. [28] proposed an Android malware detection

method named TransNet for handling the data shift. It is based on a deep transfer learning algorithm and applies transferable normalization in their deep neural network. A classification accuracy of 95.89% and an Fscore of 96.09% are obtained by their model on two public datasets. Vasan et al. [10] proposed a method named IMCFN, in which CNN is pretrained on images from ImageNet, and then the pretrained model is finetuned by malware samples. The binary codes of Apks are converted into color images, which will be used for finetuning the pretrained model. IMCFN obtains 97.35% accuracy on IoT-android samples.

Recently, DIDroid [9] was presented for malware detection. It firstly selects features using an extremely randomized tree classifier and then converts the vectors represented by selected features into 2D gray images, which are used for training the CNN model. It obtains 93% accuracy in detecting the 12 categories of Android malwares. In contrast to other image-based methods, DIDroid [9] is based on the images transferred from feature vectors (such as API calls, Permissions, etc.) rather than the images on the binary codes.

Our work is similar to DIDroid, and the benchmark dataset (i.e., CIC2020 [11]) used in DIDroid is also used in this paper. We also reshape the app data with static features into 2D gray images. In contrast to DIDroid, we propose a new data distribution-aware feature learning algorithm that aims to learn stable and discriminate features and to present an LDAM-based loss function for CNN that aims to reinforcement learn minority class samples and hard samples.

2.2 Related Work on Handling the Class Imbalance Problem

As malicious samples are difficult to obtain, malicious samples are fewer than benign samples. The trained model may be biased towards the majority class (i.e., the benign class). This paper mainly utilizes the CNN as the classification algorithm and handles the class imbalance from the aspect of improving the loss function. To handle the class imbalance in CNNs, a variety of loss functions have been proposed. Lin et al. [12] proposed the focal loss. This method is based on modulating the cross entropy loss to reinforce learning the hard samples. Kim et al. [16] proposed CCE (complement cross entropy) loss, which is designed to penalize the model heavily for incorrect predictions. Ye et al. [17] proposed a class-dependent temperature (CDT) loss function that encourages the model to balance its predictions across all classes. Cao et al. [13] proposed LDAM (label distribution aware margin) loss. This loss encourages higher margins for samples from minority classes with the aid of label distribution information to improve the performance of classifying minority classes. Existing works prove that the two factors of label distribution and classification probability are useful for building the loss function for CNN. However, they mainly only consider one of the two factors. This paper pays attention on both of the two factors based on LDAM.

3 Proposed Android Malware Detection Method

This paper utilizes CNN [14, 15] as the malware classification algorithm. This is because that the convolution operation requires less parameters than fully connected networks, which is suitable for the data in high dimension and the sparse data [23, 25]. The features

of apps can be obtained from APIs, activities, intents, permission strings, etc., which may lead to high dimensionality. The values of these features are set according to the occurrence of the corresponding features. This means that the feature vectors of apps are high dimension and sparse. Therefore, the CNN is suitable for the app data.

3.1 The Framework of Our Method

In Android malware detection, the data distribution drifts with the changing of attack behaviors and the updating of apps. In addition, malware samples are difficult to obtain. As a result, the malware class has fewer samples than the benign class. Android malware detection also faces the class imbalance problem. For handling data drift and class imbalance problems, we propose a new method named Malware Detection with data distribution-aware Feature learning and Margin loss (MDFM).

The architecture of MDFM is shown in Fig. 1. There are two stages, including feature learning and model training. In the first stage, DDAFL (data distribution-aware feature learning) is devised for learning features with a stable data distribution. It is inspired by the MFSAN [18], which is designed to learn the invariant features for cross-domain feature alignment. The stable features are helpful for enhancing malware detection performance. In the second stage, the output of conv block1 is the data represented by the learned features. These data will be further flattened and reshaped and then fed into the CNN model in Fig. 1. To handle the class imbalance problem, the loss function named M-LDAM is devised and used for CNN training.

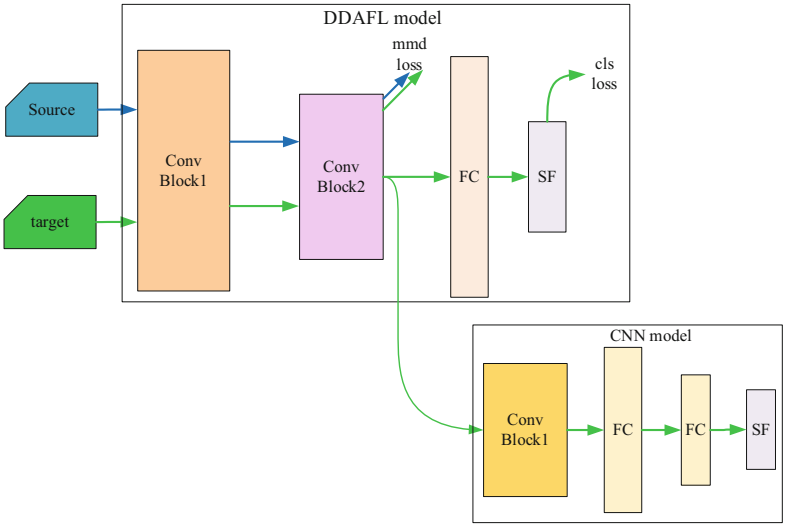


Fig. 1. The neural network architecture of MDFM

3.2 Data Distribution-Aware Feature Learning

The neural network framework of DDAFL is shown as the "DDAFL model" in Fig. 1. Conv Block1 includes three layers of conv nets. Each conv net includes a convolutional layer, batch normalization and pooling. The kernel size is 3*3 in each convolutional layer. The numbers of kernels of the three layers are 32, 64 and 128. The pooling size is 2*2. The activation function of ReLu is used for each convolutional layer. Conv Block2 includes three layers of conv nets, which learn high-level features without changing the feature dimension. Each conv net also includes a convolutional layer and batch normalization. The number of kernels is 256 in each convolutional layer. Three conv nets are followed by a pooling layer with a size of 7*7 to finally decrease the number of features.

Here, we utilize a source dataset of unlabeled samples for feature learning of the samples in the target dataset. This means that other sources of unlabeled samples could be used for feature learning. The samples from other sources are also in the domain of malware detection but from the dataset collected by other research groups. In addition, our method does not rely on the class label information. This implies that datasets from other research groups (such as unlabeled datasets shared in public) could be used to improve feature learning on target samples.

We assume that the source dataset is denoted by S and the target dataset is denoted by T. Given a batch image $\mathbf{x}_{si}$ from the source sample set $\{\mathbf{X}_{si}\}$ and a batch image x_t from the target sample set $\{\mathbf{X}_t, \mathbf{Y}_t\}$, the two batches are fed into the network of Fig. 1. Assume that F denotes conv Block1 and H denotes conv Block2, H(F($\mathbf{x}_{si}$)) denotes the output H with input of the ith source sample set $\mathbf{x}_{si}$. Similarly, the output of the H network of target sample set $\mathbf{x}_t$ is denoted by H(F($\mathbf{x}_t$)).

DDAFL is devised to learn the stable and discriminative features. This means that we hope that the data distribution of learned features is stable. Therefore, one objective of DDAFL is to minimize the data distribution discrepancy between S and T. The maximum mean discrepancy (mmd) [24] metric is generally applied to evaluate the data distribution discrepancy between S and T. Another objective of DDFAL is to minimize the classification loss for learning discriminative features on T with class label information. Therefore, the loss function of DDFAL is defined as

$$\mathcal{L}_{total} = \mathcal{L}_{cls} + \beta \mathcal{L}_{mmd} \tag{1}$$

where $\mathcal{L}_{mmd}$ denotes the mmd loss, and $\mathcal{L}_{cls}$ denotes the classification loss. Given observations $\mathcal{D}_s := \{x_1^s, x_2^s, ..., x_{ns}^s\}$ and $\mathcal{D}_t := \{x_1^t, x_2^t, ..., x_{nt}^t\}$. They are from the data distributions of P and Q, respectively. The mmd [24] is defined as Eq. (2). Based on the observed samples $\mathcal{D}_s$ and $\mathcal{D}_t$, it is a kernel two-sample test on the null hypothesis P = Q.

$$D_{\mathcal{H}}(P, Q) := \|\mathrm{E}_P[f(x^s)] - \mathrm{E}_Q[f(x^t)]\|_{\mathcal{H}}^2 \tag{2}$$

$\mathrm{E}_P[f(x^s)]$ and $\mathrm{E}_Q[f(x^t)]$ denote expectations with respect to P and Q; f denotes a feature map that maps the original samples to reproducing kernel Hillber space ($\mathcal{H}$) [24]. In reality, the unbiased estimation of $D_{\mathcal{H}}$ is defined as Eq. (3) based on the observations.

The mmd loss is defined as Eq. (4).

$$\widehat{D}_{\mathcal{H}}(P, Q) = \| \frac{1}{n_s} \sum\nolimits_{\mathbf{x}_i \in \mathcal{D}_s} f(x_i) - \frac{1}{n_t} \sum_{\mathbf{x}_j \in \mathcal{D}_t} f(x_j) \|^2_{\mathcal{H}} \tag{3}$$

$$\mathcal{L}_{mmd} = \frac{1}{N} \sum\nolimits_{j=1}^{N} \widehat{D}\big(H\left(F\left(X_{sj}\right)\right), H(F(X_t))\big) \tag{4}$$

In Fig. 1, FC denotes the fully collected layer, and SF denotes softmax. Let G denote the network (i.e., the FC layer) following H, which aims at obtaining the specific target classification loss. We apply the cross-entropy as the classification loss defined as

$$\mathcal{L}_{cls} = J(G(\mathrm{H}(\mathrm{F}(X_t))), Y_t) \tag{5}$$

The learned features are obtained by the output of the *conv Block*2 layer in DDFAL. The samples represented by learned features are used to train an Android malware detection model using CNN, as shown in Fig. 1. These features can also be used with other machine learning algorithms.

The pseudocode of DDAFL is shown in Algorithm 1. First, it randomly samples m images from S and T, obtaining $\mathbf{x}_s$ and $(\mathbf{x}_t, \mathbf{y}_t)$, respectively. $\mathbf{x}_s$ and $\mathbf{x}_t$ are sequentially fed into *convBlock*1, obtaining $\mathrm{F}(\mathbf{x}_s)$ and $\mathrm{F}(\mathbf{x}_t)$, respectively. The two outputs are further fed into *convBlock*2, obtaining $\mathrm{H}(\mathrm{F}(\mathbf{x}_s))$ and $\mathrm{H}(\mathrm{F}(\mathbf{x}_t))$. Then, it calculates the mmd loss between $\mathrm{H}(\mathrm{F}(\mathbf{x}_s))$ and $\mathrm{H}(\mathrm{F}(\mathbf{x}_t))$ according to Eq. (4). $\mathrm{H}(\mathrm{F}(\mathbf{x}_t))$ is fed into the FC layer. The classification loss is calculated according to Eq. (5). The parameters of neural networks are updated by minimizing the defined loss. The output of Algorithm 1 is the samples from dataset T represented by the learned features. The output is denoted by NT.

Algorithm 1 Data distribution-aware feature learning algorithm.

Input: source datasets $S=\{X_s\}$ and target dataset $T=\{X_t, Y_t\}$, *batch_size*
Output: dataset NT represented by learned features
for i in range (*#inters*)
  $\mathbf{x}_s$=*RS*(S, *batch_size*)
  $(\mathbf{x}_t, \mathbf{y}_t)$ =*RS*(T, *batch_size*)
  mmd_{Loss}=L_{mmd}(H(F($\mathbf{x}_s$)), H(F($\mathbf{x}_t$)))
  cls_{Loss}=L_{cls}(G(H(F($\mathbf{x}_t$))), $\mathbf{y}_t$)
  L_{total}=mmd_{Loss}+$\beta cls_{\mathrm{Loss}}$
  Update the parameters of the DDAFL network by SGD optimization
end for
return NT = H(F(X_t))

3.3 M-LDAM Loss Function

In this paper, the CNN is used as the algorithm for building the malware classification model. We address the class imbalance problem from the aspect of reinforcement learning the hard samples by modifying the loss function of CNN. Our proposed loss function

is based on AM-SoftMax loss [19] and LDAM [13]. Based on the softmax loss [20], the model predicts the sample as class C_1 when

$$|\omega_1||x|cos(\theta_1) > |\omega_2||x|cos(\theta_2) \tag{6}$$

On the class imbalanced dataset, the model may learn less knowledge from the minority class because the minority class has a small number of samples. We can reinforce the learning on the minority class samples by modifying the loss function. In order to reinforce the learning on the samples of a class C_g, the cross entropy loss is modified to reduce the output of C_g. It is defined as

$$\mathcal{L}_{M-LDAM} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s\cdot(\hat{y}_g - m_g)}}{e^{s\cdot(\hat{y}_g - m_g)} + \sum_{j=1, j\neq g}^{k} e^{\hat{y}_j}} \tag{7}$$

m_g is set the same for the majority and minority classes [20]. For handling the class imbalance problem, m_g is defined as Eq. (8) in LDAM [13] to make the model biased toward learning the minority samples.

$$m_g = \frac{(1/n_j^{1/4})C}{max_{j\in\{1,...,k\}}(1/n_j^{1/4})} \text{for } j \in \{1, ..., k\} \tag{8}$$

where n_j denotes the sample size of class C_j, k is the number of classes, and C is a hyperparameter to be tuned. However, Eq. (8) is only correlated with the sample size of each class. In the malware detection field, instances are usually sampled from existing apps. The label distribution in the training set may not be the real label distribution in existing apps. In multiclass classification, there is more than one minority class. In addition, some minority class samples may be easily identified if they are far from the boundary. Therefore, the class imbalance problem is also influenced by the difficulty of classification. To address this problem, this paper imports the information of the difficulty level of each class. The new m_g is defined as Eq. (9). Therefore, we obtain a new margin loss with the newly defined m_g, which is named as modulated LDAM (M-LDAM).

$$m_g = \frac{(1-p_j)(1/n_j^{1/4})C}{max_{j\in\{1,...,k\}}((1-p_j)(1/n_j^{1/4}))} \text{for } j \in \{1, ..., k\} \tag{9}$$

p_j is the model's classification probability of a model on class C_j. It denotes the classification confidence of correctly classifying a sample. If its value is small, the confidence is small and it is more difficult to correctly classify. Therefore, $(1 - p_j)$ is introduced to denote the classification difficulty. If $(1 - p_j)$ is high, sample x_j is difficultly identified. It is obtained by the average over the $(1 - p_j)$ obtained on the samples of C_j. $(1 - p_j)$ is used to further modulate the margin. This means that the margin of the class with higher p_j will be further reduced to reinforcement learning the samples that are easily to classify, otherwise to reinforcement learning the samples that are difficulty to classify.

4 Experiments

4.1 Experimental Datasets

In this paper, three benchmark datasets downloaded from public databases are used in our experiments. Two datasets are publicly shared by CIC (Canadian Institute for Cybersecurity) [11], and one dataset is the OmniDroid [21].

4.1.1 CIC Datasets

The two CIC datasets are CIC-InvesAndMal2019 (CIC2019 in short) and CICMalDroid 2020 (CIC2020 in short). In the CIC2019 dataset, the features include permissions and intents, which are from AndroidManifest.xml. In total, there are 8110 features in this dataset. These malware samples fall into five categories. The detail of the class distribution is shown in Table 1. In the CIC2020 dataset, the benign apps were collected from Androzoo. There are 12 malware categories, as shown in Table 2. The main extracted features include activities and permissions. The details of the features can be found in [9]. In total, there are 9503 features in CIC2020 dataset. All features can be obtained by the static analysis method, i.e., extracted from apks before running them.

CIC2019 and CIC2020 datasets have multiple classes. The details of the two datasets are shown in Tables 1 and 2. The label distribution shows that the benign class has more samples than any one of the malware classes. The two datasets face the class imbalance problem.

Table 1. Details of the CIC2019 dataset

Classes	#Samples	Classes	# Samples
Adware	95	Ransomware	93
Benign	1187	SMS	61
PremiumSMS	47	Scareware	111

4.1.2 OmniDroid Dataset

The samples in the OmniDroid dataset [21] were collected from Koodous and AndroZoo. To handle the class imbalance problem, the authors in [21] sampled 11,000 samples for malware and benign classes. The number of samples is balanced between the two classes. These samples are also represented by features obtained by static analysis, including API calls, permissions, activities and so on. The performance of different combinations of features was compared in [21]. The results show that the model with the API and Permission features performs the best. Therefore, the two feature sets are applied in following experiments.

Table 2. Details of the CIC2020 dataset

Classes	#Samples	Classes	# Samples
Adware	47197	Ransomware	6201
Backdoor	1537	Riskware	97348
Banker	886	SMS	3124
Benign	122577	Scareware	1555
Dropper	2301	Spy	3539
FileInfector	668	Trojan	13541
PUA	2050		

4.2 Experimental Results

To evaluate the performance of our proposed method, this section will carry out a variety of experiments as below.

(1) Experiment on comparing different loss functions: To handle the class imbalance problem in the CNN algorithm, multiple loss functions have been presented, including focal loss [12], CCE (complement cross entropy) [16], CDT (class-dependent temperature) [17], and LDAM (label distribution aware of margin loss) [13]. All of them are compared with our loss function M-LDAM.
(2) Experiment on comparing different CNN-based malware detection methods: To evaluate the performance of MDFM, it is compared with previous related methods, including model fine-tuning, TransNorm, and DIDroid. These methods are based on the CNN algorithm. Fine-tuning and TransNorm handle the data drift from different aspects.

In MDFM, the hyper parameters of CNN architecture are introduced in Sect. 3.2. The hyper parameters for training CNN model are set as follows: the optimization function is SGD with the learning rate of 0.01, the number of epochs is 200, and early stopping is used to avoid the overfitting problem. The other parameters in MDFM are set as follows: the batch size is 128; β in the loss function of feature learning is 0.01.

In these experiments, the *accuracy*, *Fscore* and *g-mean* are used as performance evaluation metrics. On each of the three datasets, the split ratio of training and test is 0.5. In the following experiments, each result is obtained by averaging over the ten runs of the corresponding experiment.

4.2.1 Comparison on Loss Functions

Previous works proposed multiple loss functions (CCE, FL, CDT and LDAM) for handling class imbalance problem. This section carries out experiments to compare different loss functions on handling class imbalance. The MDFM with different loss functions on the three datasets are shown in Tables 3 and 4, Fig. 2. CE denotes the cross-entropy loss function that is used as the baseline for comparison.

The results show that M-LDAM obtains 83.6%, 85.6% and 93.3% *accuracy* on the OmniDroid, CIC2019 and CIC2020 datasets, respectively. It obtains 83.6%, 52.5% and 76.8% *g-mean* on the three datasets, respectively. The results show that FL, LADM and M-LDAM perform better than other loss functions in handling class imbalance. This indicates that the existing methods of FL and LDAM perform better than others in the field of malware detection. M-LDAM performs much better than FL and LDAM on the CIC2019 and CIC2020 datasets in terms of *g-mean*. M-LDAM improves the *g-mean* by approximately 0.1%, 7.2% and 0.9% on the OmniDroid, CIC2019 and CIC2020 datasets, respectively, when compared with LDAM. This shows that the improvement of M-LDAM on the multiclass datasets is more significant than that on the binary class dataset. On the CIC2019 dataset, the *Fscores* of Adware and SMS are much worse than those of other classes. M-LDAM improves the *Fscore* of the two classes by approximately 1.6% and 13.8%, respectively. Compared with LDAM, on the CIC2020 dataset, M-LDAM decreases the *Fscores* of some minority classes, but it also improves the *Fscores* of some other classes. On average, it improves the *Fscores* of the minority classes (all classes except Adware and Benign classes) by approximately 1.8% on the CIC2020 dataset.

Table 3. Loss function comparison results on the OmniDroid dataset

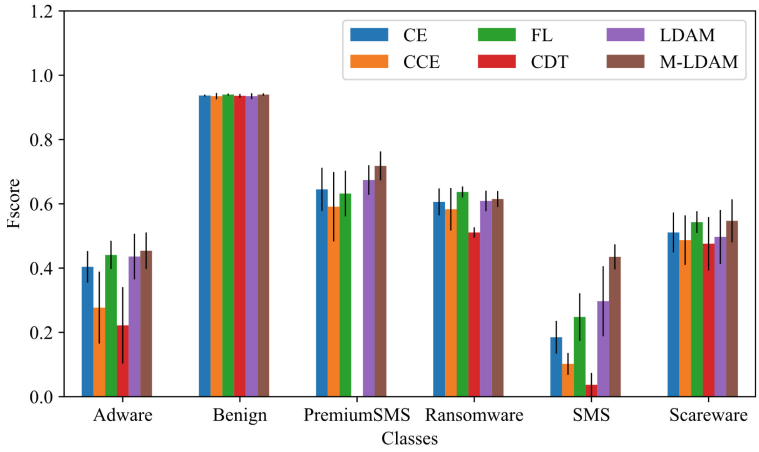
Metrics	Accuracy	Fsorce-M	Fscore-B	G-mean
CE	0.827 ± 0.004	0.823 ± 0.004	0.831 ± 0.005	0.827 ± 0.004
CCE	0.826 ± 0.005	0.823 ± 0.003	0.829 ± 0.007	0.826 ± 0.005
FL	0.831 ± 0.004	0.831 ± 0.003	0.831 ± 0.004	0.831 ± 0.004
CDT	0.827 ± 0.004	0.825 ± 0.003	0.829 ± 0.006	0.827 ± 0.004
LDAM	0.835 ± 0.002	**0.834 ± 0.003**	0.837 ± 0.003	0.835 ± 0.002
M-LDAM	**0.836 ± 0.002**	0.833 ± 0.003	**0.839 ± 0.003**	**0.836 ± 0.002**

Table 4. Loss function comparison results on the CIC2019 and CIC2020 dataset

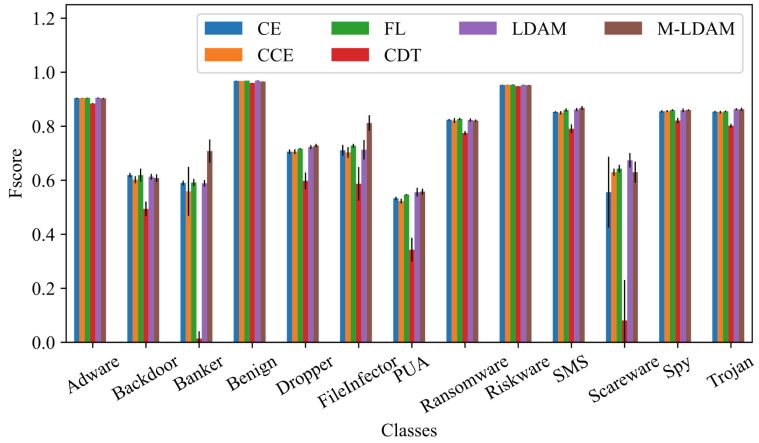
Metrics	CIC2019		CIC2020	
	Accuracy	G-mean	Accuracy	G-mean
CE	0.848 ± 0.007	0.401 ± 0.031	0.933 ± 0.001	0.711 ± 0.025
CCE	0.837 ± 0.014	0.316 ± 0.034	0.933 ± 0.001	0.710 ± 0.018
FL	0.853 ± 0.006	0.449 ± 0.036	**0.935 ± 0.001**	0.735 ± 0.004
CDT	0.819 ± 0.011	0.00 ± 0.00	0.917 ± 0.001	0.193 ± 0.194
LDAM	0.849 ± 0.012	0.453 ± 0.07	0.934 ± 0.001	0.759 ± 0.007
M-LDAM	**0.856 ± 0.008**	**0.525 ± 0.029**	0.933 ± 0.002	**0.768 ± 0.007**

The class label information and the model estimated probability are introduced into the loss function of M-LDAM. There are multiple minority classes on CIC2019 and

CIC2020. The label distribution among those minority classes cannot well reflect the difficulty of classifying those minority classes. The model estimated probability could reflect the difficulty. The difficult class with much fewer samples is double reinforced by M-LDAM to improve the performance of the minority classes.



(a)CIC2019



(b)CIC2020

Fig. 2. The *Fscore* obtained by MDFM with different loss functions on the CIC2019 and CIC2020 datasets

4.2.2 Comparison on Malware Detection Methods

Our method is compared with CNN, DIDroid, fine-tuning CNN and TransNorm. To fairly compare all methods, CNN is used as the basic classification algorithm in all malware detection methods. The neural network architecture of the CNN is shown as the CNN model part in Fig. 1. Using our method, the pretrained model on one source of data is

fine-tuned on target source of data. All data are from the field of malware detection. For example, the model pretrained on the CIC2020 can be fine-tuned on the OmniDroid, so as to adapt to the malware detection on OmniDroid data. The reason is that latent feature learned by the hidden layers of CNN can be used by the target domain of malware detection. The DIDroid is performed as in [9]. In the DIDroid method, the extremely randomized trees method is used for feature selection, and the data represented by low dimension features are used for training the CNN model. In TransNorm, the batch norm in CNN is replaced by transferable normalization [22].

Table 5. Comparison results of different models on OmniDroid

Metric	CNN	DIDroid	Fine-tuning	TransNorm	MDFM
Accuracy	0.824 ± 0.002	0.804 ± 0.006	0.731 ± 0.007	0.812 ± 0.004	$\mathbf{0.836 \pm 0.002}$
Fsorce-M	0.822 ± 0.002	0.810 ± 0.006	0.756 ± 0.005	0.807 ± 0.006	$\mathbf{0.833 \pm 0.003}$
Fscore-B	0.827 ± 0.002	0.797 ± 0.009	0.701 ± 0.012	0.816 ± 0.006	$\mathbf{0.839 \pm 0.003}$
G-mean	0.824 ± 0.002	0.814 ± 0.000	0.730 ± 0.000	0.811 ± 0.004	$\mathbf{0.836 \pm 0.002}$

Table 6. Comparison results of different models on CIC2019

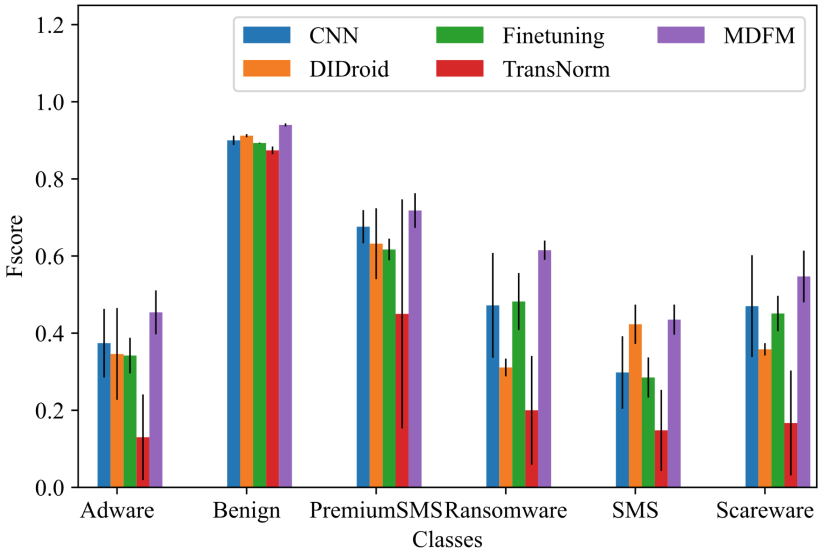
Metric	CNN	DIDroid	Fine-tuning	TransNorm	MDFM
Accuracy	0.821 ± 0.017	0.808 ± 0.009	0.796 ± 0.003	0.780 ± 0.019	$\mathbf{0.856 \pm 0.008}$
G-mean	0.390 ± 0.097	0.376 ± 0.046	0.411 ± 0.051	0.162 ± 0.116	$\mathbf{0.525 \pm 0.029}$

Table 7. Comparison results of different models on CIC2020

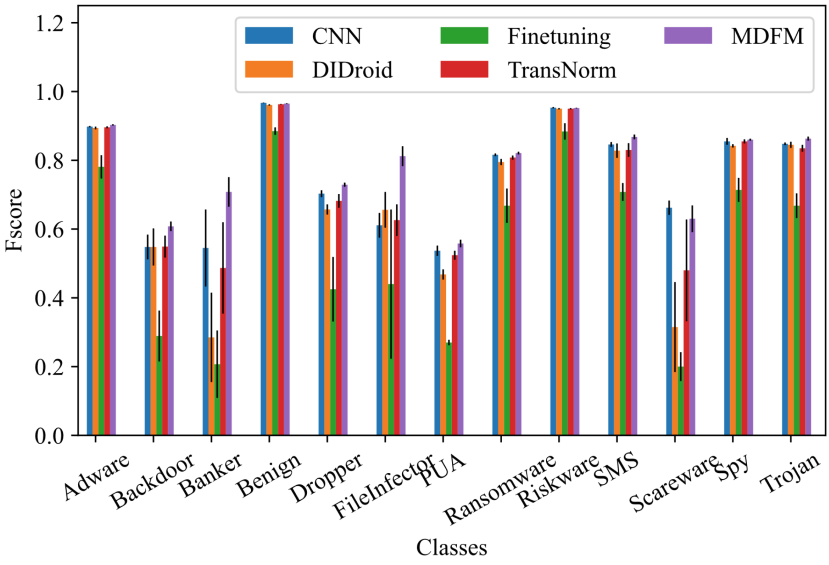
Metric	CNN	DIDroid	Fine-tuning	TransNorm	MDFM
Accuracy	0.930 ± 0.001	0.924 ± 0.004	0.841 ± 0.015	0.926 ± 0.002	$\mathbf{0.933 \pm 0.002}$
G-mean	0.744 ± 0.025	0.615 ± 0.091	0.369 ± 0.047	0.688 ± 0.037	$\mathbf{0.768 \pm 0.007}$

The results obtained on the OmniDroid dataset are shown in Table 5. The results (*accuracy* and *g-mean*) on the CIC2019 and CIC2020 datasets are shown in Tables 6 and 7, respectively, and their *Fscores* are shown in Fig. 3. The results show that MDFM obtains higher *accuracy* and *g-mean* than other methods on the three datasets, especially on the CIC2019 dataset. It improves the *accuracy* and *g-mean* by approximately 1.7% and 5.7%, respectively, when compared with CNN. Figure 3(b) also shows that MDFM obtains higher *recall* for the minority classes of Dropper, FileInfector and PUA. DIDroid also utilizes the feature reduction method before training the CNN model. It applies extremely randomized trees to select features. This paper presents a new data distribution-aware feature learning algorithm for decreasing the feature dimension. The experimental results imply that it outperforms extremely randomized trees.

The results show that the performance of the fine-tuning model is not good. The possible reason is that the data distribution of the target data is much different from that of the source data used for pretraining a model [25]. In such situations, retraining the model (i.e., the CNN model shown in above tables) performs much better. When



(a)CIC2019



(b)CIC2020

Fig. 3. The *Fscore* obtained by different methods on the CIC2019 and CIC2020 datasets

retraining the model, we also utilized the samples from other sources to learn the invariant features that improve the classification performance.

5 Conclusion and Future Work

This paper proposes a new method named MDFM for handling data drift and class imbalance in Android malware detection. To handle the data drift problem, we propose a new data distribution-aware feature learning algorithm named DDAFL. It aims to learn discriminative and stable features. To handle class imbalance, we devised a loss function named M-LDAM for CNN to handle the class imbalance problem. It pays more attention to learn the minority class samples and hard samples. Our method only relies on the assembled data of apps for malware detection. Therefore, it can be used for detecting android malwares before installing apps on the mobile devices. We carry out multiple experiments to compare our method with previous methods. The results are summarized as follows.

(1) The experimental results on comparing loss functions show that M-LDAM can improve the performance of LDAM on the multiclass imbalance problem. This is because it not only considers the label distribution but also considers the classification difficulty. Therefore, it can reinforce the training of the hard samples and minority samples.
(2) The experimental results on comparing MDFM with existing malware detection methods show that MDFM performs the best in detecting malwares. DIDroid is very similar to our work. In contrast to DIDroid, we propose a new feature learning method rather than existing tree-based feature learning. Our feature learning method aims at learning the features with stable distribution and discrimination among classes.

In this paper, we mainly consider the mmd between source data and target data to choose the source data for our feature learning. In the future, we will further analyze how to choose samples from other data sources. Our method mainly used a CNN with three conv blocks, and a CNN with more layers and neurons (such as ResNet50) could be used for DDAFL in the future.

Acknowledgments. We thank the anonymous reviewers for their constructive comments. This work is supported by the Starting Research Fund from the Guangdong University for Foreign Studies [Grant Nos. 2022RC049, 2021RC375, 2022RC100], Guangdong Basic and Applied Basic Research Foundation [Grant No. 2022A1515110980], Guangzhou Science and Technology Project [Grant No. 202201010100], Science Foundation of Ministry of Education of China [Grant No. 23YJAZH106], Guangdong Province Higher Education Teaching Reform Project.

References

1. Statcounter: Mobile operating system's market share worldwide (2022). https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed 26 Dec 2022

2. Statista: Number of apps available in leading app stores (2022). https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/. Accessed 26 Dec 2022
3. Crowdstrike: What is mobile malware? (2022) https://www.crowdstrike.com/cybersecurity-101/malware/mobile-malware/. Accessed 26 Dec 2022
4. Guo, Y., Yan, Q.: Android malware adversarial attacks based on feature importance prediction. Int. J. Mach. Learn. Cybern. **2022**, 1–11 (2022)
5. Sharma, T., Rattan, D.: Malicious application detection in android-A systematic literature review. Comput. Sci. Rev. **40**, 100373 (2021)
6. AV-Test: AV-Test Institute report (2022). https://www.av-test.org/en/statistics/malware/. Accessed 26 Dec 2022
7. Ananya, A., Aswathy, A., Amal, T.R., Swathy, P.G., Vinod, P., Mohammad, S.: SysDroid: a dynamic ML-based android malware analyzer using system call traces. Clust. Comput. **23**, 2789–2808 (2022)
8. Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., Xiang, Y.: A survey of android malware detection with deep neural models. ACM Comput. Surv. **53**(6), 126 (2020)
9. Rahali, B., Lashkari, A.H., Kaur, G., Taheri, L., Gaganon, F.: DIDroid: android malware classification and characterization using deep image learning. In: the 10th International Conference on Communication and Network Security (ICCNS 2020), pp. 70–82 (2020)
10. Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., Zheng, Q.: IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture. Comput. Netw. **171**, 107138 (2020)
11. CIC datasets (2020). https://www.unb.ca/cic/datasets/index.html. Accessed 25 May 2022
12. Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE International Conference on Computer Vision (ICCV), pp. 2999–3007 (2017)
13. Cao, K., Wei, C., Gaidon, A., Arechiga, A., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 1567–1578 (2019)
14. Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G., Dong, Y.: Convolutional neural networks for speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **22**(10), 1533–1545 (2014). https://doi.org/10.1109/TASLP.2014.2339736
15. Lan, S., He, Z., Chen, W., Chen, L.: Hand gesture recognition using convolutional neural networks. In: USNC-URSI Radio Science Meeting (Joint with AP-S Symposium), pp. 147–148 (2018)
16. Kim, Y., Lee, Y., Jeon, M.: Imbalanced image classification with complement cross entropy. Pattern Recogn. Lett. **151**, 33–40 (2021)
17. Ye, H.-J., Chen, H.-Y., Zhan, D.-C., Chao, W.-L.: Identifying and compensating for feature deviation in imbalanced deep learning. arXiv:2001.01385 (2020)
18. Zhu, Y., Zhuang, F., Wang, D.: Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 5989–5996 (2019)
19. Wang, F., Cheng, J., Liu, W., Liu, H.: Additive margin softmax for face verification. IEEE Signal Process. Lett. **25**(7), 926–930 (2018)
20. Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks, In: Proceedings of the 33rd International Conference on Machine Learning, pp. 1–10 (2016)
21. Martín, A., Lara-Cabrera, R., Camacho, D.: Android malware detection through hybrid features fusion and ensemble classifiers: the AndroPyTool framework and the OmniDroid dataset. Inf. Fusion **52**, 128–142 (2019)

22. Wang, X., Jin, Y., Long, M., Wang, J., Jordan, M.I.: Transferable normalization: toward improving transferability of deep neural networks. In: Advances in Neural Information Processing Systems, pp. 1951–1961 (2019)
23. AjitkumarSingh, S., GautamMeitei, T., Majumder, S.: Short PCG classification based on deep learning. Deep Learn. Tech. Biomed. Health Inf. **2020**, 141–164 (2020)
24. Gretton, A., Borgwardt, K.M., Rasch, M.J., Scholkopf, B., Smola, A.: A kernel two- sample test. JMLR **13**(Mar), 723–773 (2012)
25. Liu, Z., Wang, R., Peng, B., Gan, Q.: A convolutional neural network based Android malware detection method with dynamic fine-tuning. In: 32nd ITNAC, pp. 300–305 (2022)
26. Yadav, P., Menon, N., Ravib, V., Vishvanathan, S., Pham, T.D.: EfficientNet convolutional neural networks-based Android malware detection. Comput. Secur. **115**, 102622 (2022)
27. Ren, Z., Wu, H., Ning, Q., Hussain, I., Chen, B.: End-to-end malware detection for Android IoT devices using deep learning. Ad Hoc Netw. **101**(4), 102098 (2020)
28. Rong, C., Gou, G., Cui, M., Gang, X., Li, Z., Guo, L.: TransNet: unseen malware variants detection using deep transfer learning. In: International Conference on Security and Privacy in Communication Systems, pp. 84–101 (2020)



Reputation-Based Dissemination of Trustworthy Information in VANETs

Vincenzo Agate(✉), Alessandra De Paola, Giuseppe Lo Re, and Antonio Virga

Department of Engineering, University of Palermo, Palermo, Italy
{vincenzo.agate,alessandra.depaola,giuseppe.lore,
antonio.virga01}@unipa.it

Abstract. With the emergence of new vehicle communication paradigms such as Vehicle-to-Everything, the possibility of providing advanced services to drivers is becoming a reality. The immediate and targeted warning of dangers offers the opportunity to increase driving safety and make optimal use of the road infrastructure. However, communication reliability between vehicles, or worse, passenger safety, may be compromised by vehicles modified to spread false information or create disorder under coordinated malicious groups. Solutions currently adopted in similar scenarios include the use of Reputation Management Systems (RMS), which allow the reliability of received information to be estimated. However, classic centralized RMSs do not fit the distributed and dynamic nature of vehicular networks.

In this paper, a step is taken towards the design of a fully distributed event detection and dissemination system for VANETs, based on vehicle and data reputation, which does not rely on any fixed communication infrastructure. A new reputation model is proposed to reliably detect events and a new communication protocol is defined to disseminate information among vehicles, based on the population protocol model. The experimental evaluation performed on realistic vehicle routes demonstrates the feasibility of the proposed system and its ability to withstand orchestrated attacks, with a significant performance improvement over other state-of-the-art solutions.

Keywords: VANET · Reputation Management · Event Dissemination

1 Introduction

Intelligent Transport Systems (ITS) have the potential to significantly improve road safety. The emerging Vehicle-to-Everything (V2X) communication paradigm promises to achieve such a goal by enabling easy exchange of information between vehicles, between vehicles and infrastructure, and between vehicles and vulnerable road users (VRUs) such as pedestrians and cyclists [1]. The pervasive use of this technology would pave the way for the implementation of smart services, such as congestion analysis, traffic management, collision avoidance, cooperative driving and comfort/infotainment applications [2], which

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 445–463, 2024.
https://doi.org/10.1007/978-3-031-63989-0_23

could have a significant impact even on road safety, reducing both the severity and the number of accidents.

Two communication paradigms exist to regulate the flow of information between vehicles in a vehicle network: Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V). The first model uses roadside units (RSUs), and there may be difficulties in its realisation for environmental or economic reasons. On the contrary, V2V networks only exploit direct communication between vehicles, thus being independent of any pre-existing physical infrastructure. However, the quality of services provided through this paradigm can be negatively affected by several physical factors, such as vehicle speed, high traffic density and the presence of physical obstacles that lead to communication reliability problems. Furthermore, since centralized quality control is not possible, false information could be spread due to faulty sensors or, even worse, information could be conveniently falsified by malicious parties to cause damage to the system.

To address this problem, trust and reputation-based systems may be the suitable solution to assess the reliability of data exchanged through collaborative vehicle activity [3,4]. Recent solutions proposed in the literature make use of trusted entities, complex authentication schemes and certificates, all of which are incompatible with the dynamic and distributed nature of the scenario under consideration.

This paper proposes a new solution to perform reliable event distribution, suitable for both V2V and V2X contexts. The proposed solution adopts a new communication protocol, based on the population protocol model [5], a theoretical model designed to manage a set of autonomous agents that interact randomly to carry out distributed computations. The proposed solution is also based on an original reputation management system to improve the overall quality of information (QoI) through timely analysis of vehicle reputation and filtering of distributed false messages. To this end, the system uses two different metrics to estimate both the reliability of an event on the basis of the knowledge derived from the information reported by all the interacting vehicles, and to estimate the reputation of the individual vehicles, in order to appropriately weight the information received. The experimental evaluation shows that the proposed solution is resistant to various types of attacks organised by malicious user groups and achieves better performance than recent solutions proposed in the literature.

The main contributions of the proposed work can be summarized as follows:

- A new fully distributed event detection and dissemination system for VANET that does not rely on any fixed communication infrastructure;
- A three-tier architecture that leverages a reputation management module to evaluate the reliability of vehicles and information received, and leverages the population protocol paradigm for event dissemination;
- An extensive experimental validation performed with a dataset containing realistic vehicle tracks over a real area, allowing the performance of the proposed system to be compared with a state-of-the-art system.

The rest of the paper is organized as follows: Sect. 2 presents related works and describes their main limitations; the proposed reputation model and the

event diffusion protocol are described in Sect. 3; the experimental evaluation is presented in Sect. 4. Final considerations and conclusions are given in Sect. 5.

2 Related Work

Some of the main goals of ITS include improving road safety through Vehicular ad hoc networks (VANETs), reducing traffic accidents and residual hazards by enabling cooperation between vehicles for timely information exchange. However, this scenario opens the door to the spread of false information from noisy sensors, or worse, from malicious entities intent on causing inconvenience or damage to the system. In fully distributed environments, an increasingly common solution is to use trust and reputation techniques to detect untrustworthy data or malicious users [6,7]. The basic idea behind this type of solution is to use the history of past interactions to assess the trustworthiness of information shared by participants. There are numerous application scenarios in which reputation-based approaches have been used, such as IoT, e-commerce, participatory sensing [8], etc., but implementation in the context of vehicular networks is even more challenging due to the dynamic nature of the network and the volatility of the information processed.

The authors in [9] proposed a trust-based model to protect V2X communications against internal attacks, by using vehicle-referenced adaptive weights to filter recommendations. However, this solution relies on the presence of RSUs, which are not always available, and suffers from a progressive degradation of the QoI as one moves away from it.

Some works in the literature address the problem of estimating vehicle reputation even in the absence of RSUs, i.e. in a V2V scenario, but, given the highly dynamic nature of VANETs, which are characterized by unpredictable topologies due to the highly variable speeds of their nodes, this goal poses many challenges. Some solutions, such as [10], propose models that use only physical characteristics to evaluate nodes and the information they share, such as packet delivery rate (PDR) and average delivery delay (ADD). However, such systems fail to detect attackers who spread poor quality or false information.

The Attack-resistant Trust Management (ART) scheme [11], one of the first in the VANET domain, has been proposed to assess the reliability of both nodes and messages. Data reliability is estimated based on data collected from multiple vehicles. To measure the reliability of a node, the system combines the estimated probability that the vehicle will perform its tasks and the probability that its recommendations are reliable.

To counter the potential impact of negative feedback from malicious vehicles, the authors of [12] propose REPLACE. This is a reliable recommendation scheme based on a platooning service, i.e., a driving model in which vehicles with common goals move cooperatively. In REPLACE, the intent is to recommend a reliable platoon leader to coordinate the platooning service. However, this model exploits a centralized reputation system to calculate scores using user feedback and cryptographic techniques such as public key cryptography and session key

agreement between vehicles and RSUs, which require fixed static infrastructure. Similarly, the authors of [13] propose a scheme that exploits clustering techniques to elect a cluster head (CH) responsible for sending trusted information into the network. However, the main drawback of this approach is the possibility of electing dishonest CHs when the majority of nodes are dishonest [14]. The study presented in [15] suggests a scheme to calculate trust between entities, identify malicious nodes, and disseminate this information in a network. Malicious vehicles are excluded from the network using different trust metrics, such as event-based trust, direct trust, and indirect trust. The dissemination of overall trust values is done through the periodic exchange of beacon messages. However, the attack model assumed by the authors, i.e., that malicious vehicles always exhibit malicious behavior, is unlikely to be true in a real-world context [16].

The authors of [17] propose a reputation management system called MARINE, with the purpose of detecting malicious nodes that launch man-in-the-middle attacks. The MARINE trust model works in two steps to assess inter-vehicle trust. First, it evaluates the sender node to determine its trustworthiness. This is done through previous interactions and recommendations from neighboring vehicles. Second, once node-centric trust is calculated, the received data is evaluated in three different dimensions: information quality, node's ability to forward messages, and neighbors' opinions. Data from the sender node is accepted only if node-centric and data trust are successfully calculated. Otherwise, the evaluating node will discard the data. MARINE relies on both vehicles and RSUs to compute the overall trust in the sender and the received information.

One of the most recent works on reputation management for vehicular networks has been proposed by the authors of [18]. In this work, the authors propose a distributed RMS that evaluates the behavior of nodes participating in the network as the result of direct interaction and through the recommendation values obtained from neighborhood. In order to correlate the two trust values (direct and indirect), the system uses a coefficient that takes into account qualitative attributes such as familiarity, similarity and timeliness. This allows different weights to be assigned to both trust values. Finally, to detect and identify a malicious node, a trust threshold mechanism is introduced, which, based on specific rules, determines whether or not to exclude the node under investigation from the network.

An in-depth discussion of recent work in which the assessment of the quality of shared information is driven by trust in VANETs can be found in [19].

Some of the approaches recently proposed [14,17] in the literature base information quality control on fixed infrastructural elements or trusted entities that may not always be available, on the contrary, in the solution proposed here the diffusion of events is obtained through a population protocol in a completely distributed manner. Furthermore, information is disseminated on messages to other vehicles only after passing through a check by a reputation management

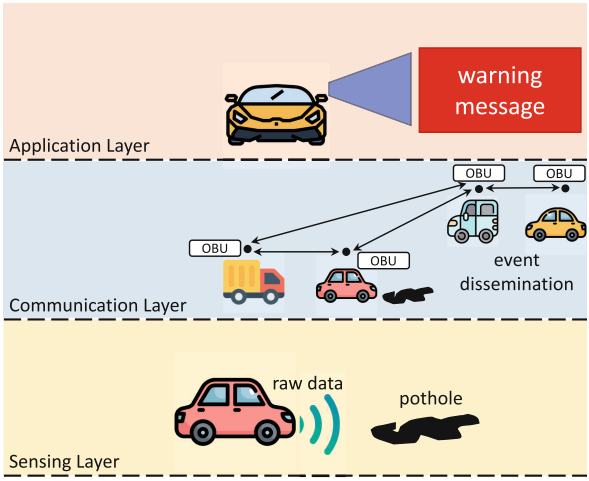


Fig. 1. The multi-layer architecture of the system.

module. By exploiting the proposed layered architecture, the reputation module is able to exploit not only the direct experiences of event perception [20], but also the appropriately weighted reported information.

3 The Multi-layer Vehicular Architecture

The system proposed here exploits a three-layer architecture structured as shown in Fig. 1.

At the lowest layer, the *Sensing Layer* (SL) performs the task of detecting events during vehicle motion. To fulfill this role, the *Sensing Layer* exploits on-board sensors (e.g., cameras, proximity sensors, accelerometers, gyroscopes, air quality sensors, GPS, etc.) that interact with the surrounding environment. At this level, the raw data collected by the sensors as the vehicle moves is passed to a data fusion module, which is responsible for analyzing and classifying events of interest, such as the presence of traffic congestion, potholes [21], accidents and so on. The output of the *Sensing Layer* is a list of events that are sent to the higher layer, the *Communication Layer* (CL), which has two main functions: to spread the knowledge of events directly sensed by the vehicle and to receive events reported by other vehicles with which it has managed to communicate. The functions implemented by the *Communication Layer* allow vehicles to know events promptly thanks to information obtained by other vehicles. To achieve a reliable event dissemination algorithm, the quality of the spread information must not be underestimated. Since no centralized authority can be utilized, the dissemination of false information opens the door to possible attack scenarios.

To maintain control over the quality of disseminated events, a distributed reputation and trust module is employed. This ensures that only reliable events are shared with other vehicles. To this end the *Communication Layer* uses a

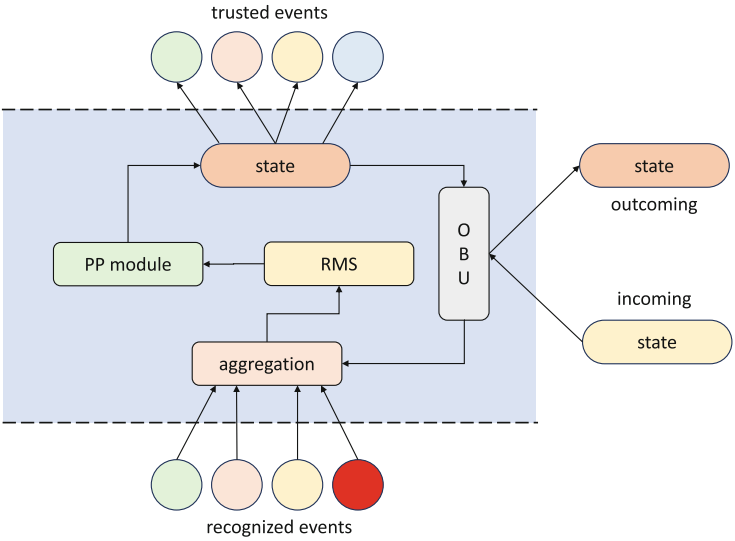


Fig. 2. Internal structure of the Communication Layer.

communication protocol based on Population Protocols, a communication model originally designed to enable sensor networks with limited resources to achieve a common view of their environment. The vehicle cooperation mechanism, carried out by the joint action of the reputation module and the population protocol, is essential for vehicles to obtain information about events not in their proximity and in advance, enabling them to plan appropriate actions to achieve their goals.

Although not all vehicles currently have the ability to benefit from hardware such as OBUs and sensors, which allow them to freely share data to participate in ITS services, it is possible that in the near future manufacturers will equip all vehicles on the road with such tools. Additionally, numerous efforts are underway to make these technologies available for older vehicles at an affordable cost [22].

On top of this architecture, the *Application Layer* (AP) receives the list of reliable events generated by the *Communication Layer* to provide specific advanced services aimed at improving the user's driving experience and safety (such as notification of nearby incidents, the ability to plan actions based on events, etc.).

The core of this system is the Communication Layer, whose main components, shown schematically in Fig. 2, are described in the remainder of this section. This layer has an internal state that summarizes events detected or inferred from messages sent by other vehicles. This information is shared with other vehicles through the on-board units (OBUs), which are also responsible for collecting incoming messages. Following the rules of the proposed population protocol, the received messages are subjected to a filtering process based on a reputation model, and the resulting information contributes to the state update.

3.1 State Model

In the system proposed here, communication between vehicles is performed according to the rules of a Population Protocol (PP). Each vehicle follows the protocol and interacts with other vehicles in its communication range. During an interaction between source and destination vehicles, a message containing a triplet of values is sent:

$$m = \{v_{id}, s, ts\}\,, \tag{1}$$

where v_{id} is a unique identifier for each vehicle, and ts is the timestamp at which the message was sent. The most important part of the message is the state s, which contains two lists:

$$s = \{E_T, E_U\}, \tag{2}$$

where E_T is a list of events considered reliable and E_U is a list of events considered unreliable. The content of each list is determined taking into account both direct interaction with the environment and information reported by other vehicles. The details of how the two lists are populated are discussed later in this section.

Each event ϵ_i contained in one of the two lists of the state s is a tuple of values composed as follows:

$$\epsilon_i = \begin{cases} class_i \\ (x_i,\ y_i) \\ ts_i \\ v_{id} \end{cases} \tag{3}$$

where $class_i$ is an attribute indicating the type of event (e.g., accident, pothole, etc.), the pair $(x_i,\ y_i)$ indicates an approximation of the geographic coordinates relative to the event, ts_i indicates the time at which the event was detected, and finally v_{id} indicates the identifier of the vehicle that detected the event.

3.2 Aggregation of Events

Before assessing the trust level of events and the reputation of vehicles, events are grouped according to their class and geographical coordinates. Since different vehicles may perceive the same event in slightly different geographical locations due to poor accuracy or noise, it is crucial that these events are considered to be the same actual event. Figure 3 helps to understand the problem just discussed.

The proposed solution involves a clustering step implemented through the DBSCAN algorithm [23], which is designed to detect clusters, even of different densities, in large spatial datasets. The choice of this clustering algorithm over others is motivated by the fact that it does not require knowing the number of clusters in advance and can work with clusters of any distribution and shape. Its operation is governed by two parameters, namely the maximum allowed distance *eps* between two points in the same cluster and the minimum allowed size $MinPts$ of a cluster. In the considered scenario they represent the maximum distance in terms of geographic coordinates between two events that should be

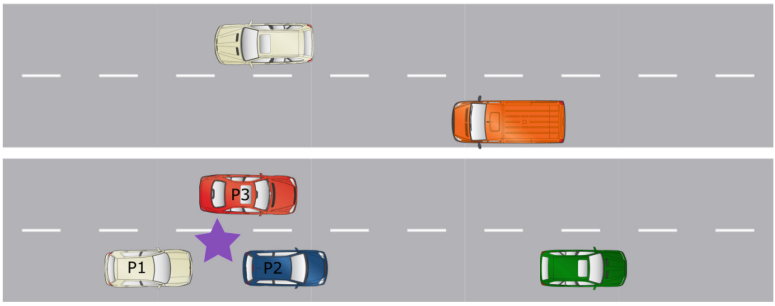


Fig. 3. Example scenario where a set of events reported by different vehicles refer to a single real event. The clustering phase is necessary to aggregate slightly different views within the system.

considered as the same real event and the minimum number of events to be able to form a cluster respectively. Since it is not possible to determine a priori how many times the same event will be perceived, and even a single event may be sufficient to identify a cluster by itself, the value $MinPts$ is assumed to be $MinPts = 1$, thus avoiding single actual events as noise points. The *eps* parameter is set by considering as a reference the maximum distance at which the vehicle sensors are able to obtain raw measurements, plus a tolerance margin of 10% as a fair trade-off to avoid the presence of cluster merging or fragmentation.

3.3 Reputation Model

To quickly propagate event information, each vehicle sends its neighborhood a list of events it considers trustworthy and untrustworthy. Since reported event lists may contain unreliable information, the Reputation Management Subsystem (RMS) is responsible for estimating the level of trust in reported events whenever one is received from a vehicle. To do this, the RMS performs three basic steps: computing the local trust of events (I), estimating the reputation of vehicles (II), and finally computing the trust of received events (III).

Local Trust of Events. Once the RMS receives the list of all aggregated events, it proceeds to calculate the local trust. The different events are now distinguishable by a tag specifying their identifier, called $cluster_{id}$, and through this the RMS can actually count the number of times it has been reported by a vehicle as a trusted and untrusted event. The local trust value $lt(\epsilon_i)$ for each event ϵ_i is calculated through the following equation:

$$lt(\epsilon_i) = \frac{m(\epsilon_i) - k(\epsilon_i)}{n}, \tag{4}$$

with $m(\epsilon_i), k(\epsilon_i) \in [0, n]$ and $0 < m(\epsilon_i) + k(\epsilon_i) \leq n$. The value $m(\epsilon_i)$ represents the number of vehicles that consider the i-th event as trustworthy (i.e., ϵ_i is

contained in the E_T list), while $k(\epsilon_i)$ is the number of vehicles that consider the i-th event as untrustworthy, (i.e., ϵ_i is contained in the E_U list). Finally, n is the number of vehicles that sent their state during the last time interval. Given Eq. 4, the values of $lt(\epsilon_i)$ are in the range $[-1, 1]$.

Vehicle Reputation. Using the trust values of the individual events $lt(\epsilon_i)$ obtained in the previous step, it is now possible to assess the trustworthiness of neighbouring nodes. The idea behind the new score is to evaluate the amount of reliable information received by each of the participants with whom the vehicle interacted.

Assuming that each vehicle has a limited amount of memory available, only the events contained in a time window of fixed size, W, are considered. In order to make the reputation values of the k-th vehicle resistant to sudden fluctuations, the history of received messages is taken into account. For this reason, each received message is associated with a trust value calculated as follows:

$$t_m(m_j^k) = \frac{\sum\limits_{\epsilon_i \in m_j^k} lt(\epsilon_i)}{\left|m_j^k\right|}, \tag{5}$$

where m_j^k represents the j-th message received by the vehicle k.

After obtaining the trust values of each message, the RMS proceeds to calculate the reputation of the vehicle k, called $r(v_k)$, by averaging the trust values of the messages received by it:

$$r_k^t = \alpha * t_m(m_j^k) + (1 - \alpha) * r_k^{t-1}. \tag{6}$$

Note that the above equation uses the exponentially weighted moving average (EWMA), which, as the smoothing factor α varies, allows us to weigh the most recent observation more or less heavily against past history.

Trust of Received Events. After the calculation of the reputation values, the RMS estimates the trust values for the events contained in the messages received in the current step. This phase is essential for the events received by the OBU to reach the PP, which, based on the trust values, will decide to place the events in one of the two lists contained in the state to be shared.

The trust $t(\epsilon_i)$ of the i-th event is calculated through the following equation:

$$t(\epsilon_i) = \frac{\sum\limits_{k \in K} r_k^t - \sum\limits_{z \in Z} r_z^t}{n}, \tag{7}$$

where K is the set of nodes that reported that the event ϵ_i was reliable, while vehicles in Z reported the opposite and $n = |K| + |Z|$.

Algorithm 1: Transition Function $\delta(\cdot)$ upon receiving an internal state

Input : states of the receiving node (s_i), states of the sending node (s_j)
Output: Updated s_i to s_i'

```
for event ∈ s_j.E_T do
    if event ∉ s_i.E_T and event ∉ s_i.E_U then
        s_i.E_T.append(event);
    else if event ⊂ s_i.E_U then
        s_i.E_U.remove(event);
        s_i.E_T.append(event);
for event ∈ s_j.E_U do
    if event ∈ s_i.E_T then
        s_i.E_T.remove(event);
        s_i.E_U.append(event);
```

3.4 Diffusion Model

The goal of the PP module is to ensure that the status of the vehicle is updated with information acquired both during communication with neighboring nodes and with events obtained from the lower Sensing Layer [24,25]. The proposed diffusion model exploits a unidirectional PP [5], which unlike the basic PP, where the update of states occurs only after two agents are mutually synchronized (i.e., after both have exchanged states), the update occurs upon receipt of a message. This feature is specifically designed for VANETs, since node mobility and other phenomena typical of wireless communications, such as fading, could compromise the symmetry of communication.

The Population Protocol proposed here receives the following input paramenters:

- An alphabet of possible initial values Σ, that contains only the null symbol $\varnothing$. In this way, the input function ι initializes the state of each node to a single base configuration common to all nodes. More formally:

$$s : \iota(\varnothing) = \{E_T, E_U\} \; where \; E_T, E_U = \emptyset \tag{8}$$

- An output function ω that, given an input state s, generates a set of trustworthy and untrustworthy ϵ_i events contained in two different lists.
- A transition function δ, used to update the state of the receiving node, defined as follows:

$$\delta(s_i, s_j) = (s_i', s_j'). \tag{9}$$

  In the above equation s_i and s_j represent the states of the receiving node and sending node before the update, while s_i' and s_j' are the states after the update. Considering that the chosen PP is unidirectional, the state of the sender node remains unchanged and only the state of the receiver node undergoes a change, in other words $s_j = s_j'$.

Algorithm 2: Transition Function $\delta(\cdot)$ upon receiving an external state

Input : states of the receiving node (s_i), states of the sending node (s_j), id of the vehicle that is performing the calculation ($myId$), threshold high (θ_h), threshold low (θ_l)
Output: Updated s_i to s_i'

```
1  for event ∈ s_j.E_T ∪ s_j.E_U do
2      if Trust(event) > θ_h then
3          if event ∉ s_i.E_T and event ∉ s_i.E_U then
4              s_i.E_T.append(event);
5          if event ∈ s_i.E_T then
6              Update(event);                    ▷ Update timestamp and vehicle id
7          if event ∈ s_i.E_U and s_i.E_U.event.v_id ≠ myId then
8              s_i.E_U.remove(event);
9              s_i.E_T.append(event);
10     if Trust(event) < θ_l then
11         if event ∉ s_i.E_T and event ∉ s_i.E_U then
12             s_i.E_U.append(event);
13         if event ∈ s_i.E_U then
14             Update(event);                    ▷ Update timestamp and vehicle id
15         if event ∈ s_i.E_T and s_i.E_T.event.v_id ≠ myId then
16             s_i.E_T.remove(event);
17             s_i.E_U.append(event);
```

The transition function $\delta(\cdot)$ has two different behaviors depending on the source of the received state s_j. In particular, the state s_j could come from a node with which the vehicle has interacted or from the *Sensing Layer*. In both cases, the internal state must be updated. Algorithms 1 and 2 summarize the transition function for both scenarios.

According to Algorithm 1, i.e. the case where s_j comes from the underlying layer of the architecture, the events ϵ_i perceived from the vehicle's environment are considered fully trusted. The action performed by the transition function in such a case is to adjust the internal state s_i by placing the events in the lists E_T and E_U, checking for consistency with the knowledge base.

When s_j is the result of exchanging messages with another vehicle, the transition function follows the behavior specified in Algorithm 2. In this scenario, the trust values associated by the node to the received events assumes a key role in correctly updating the s_i state. After setting two thresholds θ_h and θ_l, the first to discriminate trusted events and the second for untrusted events, the algorithm proceeds by analyzing the received events one by one, comparing the trust with the thresholds. As can be seen from the algorithm, the part of pseudocode from line 2 to line 9 concerns events whose confidence is greater than θ_h. In this case, the transition function evaluates whether the event should be



Fig. 4. Map of the test area.

placed in E_T if it was not previously known, or if it is in the list of untrusted events E_U, it should be moved if and only if the information does not conflict with that obtained by its own sensory apparatus. The logic used is that direct experiences always overrides referred information.

Finally from line 10 to line 17, the transition function performs the opposite behavior for events whose trust is below the θ_l threshold.

4 Experimental Evaluation

This section shows the results of the experimental evaluation of the proposed solution, demonstrating that it is well suited for the dissemination of truthfull events in VANETs, and is also resilient to the threat of security attacks by vehicles opportunistically orchestrated to undermine the system.

In addition, the performance of the proposed system in terms of accuracy, precision, recall, and F1-score are compared with the performance of a state-of-the-art technique.

4.1 Simulation Environment

The experimental evaluation has been performed by using the open source framework VEINS [26], which is based on two simulators, SUMO [27], a road traffic simulation suite, and OMNET++ [28], a C++-based simulation library that is suitable for creating network simulators. The proposed model can be employed in a variety of road scenarios, regardless of the network topology. The experiments reported in the following were conducted in urban environments simulated from real maps.

4.2 Experimental Setting

The OpenStreetMap tool was used to create the simulation dataset, and the resulting map covers an area of the university campus of the city of Palermo. Part of the map is visible in Fig. 4, and the size of the working area is 6 Km^2.

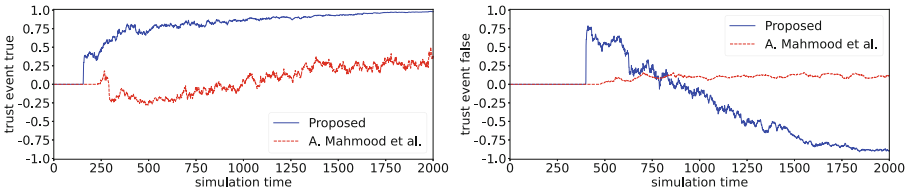
Communication was made more realistic by adding obstacles such as buildings using SUMO's polyconvert script. The communication radius is set to 80 m for each of the vehicles and a message exchange frequency is 5 Hz. The maximum useful distance for recognition of directly perceived events was also set at 3 m for each node, and the speeds of the vehicles in the map depend on the actual speed limits imposed by the roads, which range from 7 m/s to 13 m/s. In order to maintain a proper balance between new reputation values and past history, the α value is set to 0.5 [29].

The red-colored indicators in Fig. 4 represent the events of interest for the scenario considered here (e.g., construction sites, hazards, accidents, etc.). The locations of the events were chosen from all possible intersections of the roads, extracting them according to a Gaussian distribution with mean 0 and variance 1. Each simulation was run for the duration of about half an hour (2000 s).

To demonstrate the effectiveness of the proposed system, its performance was compared with one of the best performing systems in recent literature, which does not use a fixed infrastructure for reputation calculation and information dissemination [18]. The authors propose a system to assess the reputation of vehicles participating in the network by exploiting both direct and indirect trust in order to exclude a possible untrusted node. To do this, they implement a thresholding mechanism that gives a vehicle multiple opportunities to improve its behavior before it is removed from the sharing network. To make a fair comparison between the two systems, different experimental runs were performed under the same conditions, replacing the proposed RMS subsystem with their proposed reputation mechanism and leaving the information dissemination system based on the Population Protocol unchanged.

Model of Malicious Vehicle. In order to test the system under complex though realistic attack scenarios, a set of vehicles with malicious behavior are generated in the map. To ensure that the attack scenario is even more effective, such vehicles are distributed in the map to form evenly distributed outbreaks.

The behaviour of an attacking vehicle consists of adding false information in the messages it disseminates to other vehicles, such as events that do not really exist on the map. In order to make the attack even more insidious, the attacker injects small amounts of false information into messages containing the majority of trustworthy information [30,31]. As a result, the attacker is unlikely to be completely detected while maintaining an average reputation value. In the experiments described below, there will be between 10% and 30% malicious vehicles out of a total of 100 vehicles, and each malicious vehicle spreads false information about one event out of 8 authentic events on the map. The attacks start at simulation time 400.



(a) (b)

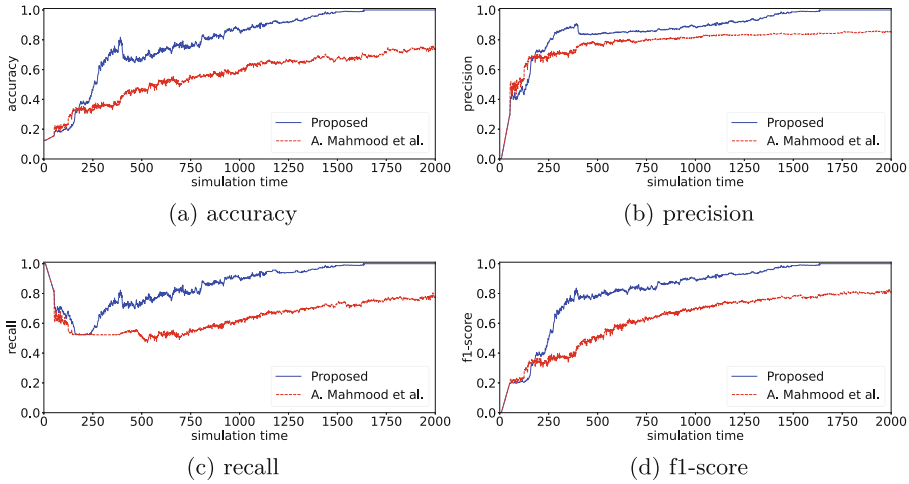
Fig. 5. Comparison of the trend of trust values of a good event (a) and a malicious event (b) between the proposed system and the solution adopted as benchmark.

Since the range of possible trust values is $[-1, 1]$, such interval is divided in the following three regions: trusted, untrusted, and uncertain region. The thresholds values to discriminate an event as trusted or untrusted were set at $\theta_h = 0,33$ and $\theta_l = -0,33$, so that the range of possible confidence values is divided equally among the three labels (i.e., trustworthy, uncertain, and untrustworthy).

4.3 Experimental Results

The first experiment, shown in Fig. 5, is useful to compare the basic operation of the system proposed here against the solution proposed in [18]. Specifically, Fig. 5-a shows the trend of the average trust value of an true event, while Fig. 5-b shows the trend of a false event propagated by the attacking nodes. In this first experiment there are 100 vehicles in the map of which 5% spread false information. As the left curve shows, both systems correctly estimate the real event, with a trend toward the maximum trust value as simulation time increases, although the proposed approach reaches the maximum trust value faster. Figure 5-b shows the trend of the trust value of a malicious event spread simultaneously by the attacking vehicles at time 400. By the curve in blue, it is possible to observe that the average trust estimated by vehicles exploiting the proposed system is initially affected by the false information disseminated by attackers initially considered trustworthy. After the initial phase, the system gradually recognizes false information, as confirmed by the decreasing trend of trust. Surprisingly, the system proposed in [18] initially appears to be unaffected by the spread of false information. However, the slightly increasing trust trend suggests that the system fails to isolate the false information, and suffers the nodes attack. Effectively, the attack is particularly insidious for both systems given that according to the previously described attack model the misinformation, i.e., the false event, is anyway spread along with a number of correct information that allows the attacking nodes to survive in both systems. This demonstrates the remarkable false-event discrimination capability of the approach proposed here.

The second set of experiments aims to quantitatively compare the performance of both systems in correctly detecting events. Since malicious event detection can be considered as a binary classification problem, where an event can be either true or false, according to [32], the performance of both systems can



(a) accuracy (b) precision

(c) recall (d) f1-score

Fig. 6. Comparisons of the trends of evaluation metrics of the proposed model against the RMS proposed in [18].

be evaluated using the following well known metrics: *accuracy*, that is the ratio of correct predictions to all predictions made, *precision*, which is the ratio of correct positive predictions to all positive predictions made, *recall*, which is the ratio of correct positive predictions to all predictions that must be positive, and *F1_score*, which is the harmonic mean between *precision* and *recall*. To solve the problem of the large imbalance between the class of actual events and the class of malicious events, a weighted definition of these metrics was adopted.

Figure 6 shows the trends of the four metrics examined. From Fig. 6-a showing the accuracy trend, it is possible to see the strong initial growth due to the bootstrap phase. During this phase all nodes entered into the network begin to build their knowledge base from scratch. Note how, from simulation time 250, the proposed system outperforms the classification capabilities of the competitor. Starting at time step 400, the clique attack of 10% of the total nodes begins. The attack is appropriately orchestrated so that all attackers simultaneously spread the same false event.

The performance of the system actually degrades slightly, although it remains above its competitor. Although the attack phase never ends until the end of the experiment, the proposed system succeeds in reaching the maximum accuracy value confirming its ability to detect and isolate false information spread by attackers. The Precision, Recall and F1-score curves also confirm the superiority of the proposed system.

The last experiment, shown in Fig. 7, aims to evaluate the cumulative performance of the 4 metrics on the two systems as the percentage of attacking nodes varies from 10% to 30% of the total vehicles. All metrics show imperceptible performance degradation as the number of attackers increases, demonstrating the effectiveness of both solutions. In all attack configurations however, the sys-

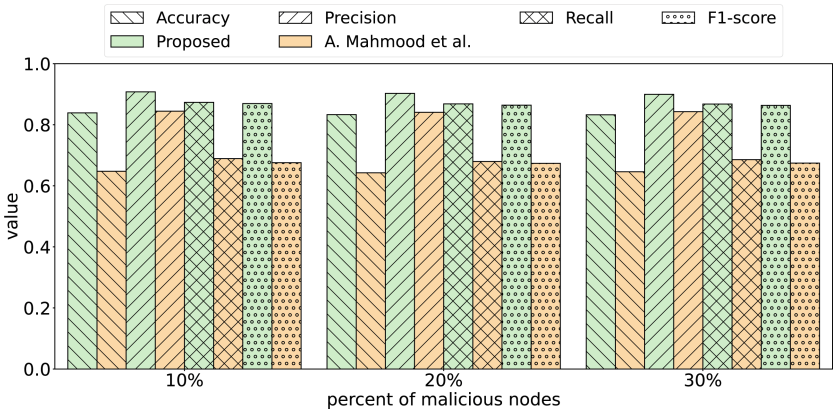


Fig. 7. Comparison of systems performance as the percentage of attackers on the network changes.

tem proposed here consistently outperforms the second system by achieving the largest gap in F1-score value (about 30% better than its competitor) and the smallest gap between precision values (about 10% better than its competitor).

5 Conclusion

This paper presents a novel solution for the dissemination of reliable information within VANETs, which can recognise malicious vehicles and exclude false events spread to cause damage to the network.

In the proposed system, trust in the event is estimated by exploiting two separate metrics, namely the local trust of the event and the reputation of the vehicles sharing information on the same event. The former provides a preliminary estimate of the trustworthiness of an event based on knowledge derived from information reported by other vehicles, while the latter depends on the reliability of vehicles in the network, i.e. their reputation.

Unlike other approaches proposed in recent literature, which rely on an infrastructure or trusted entities that may not always be available, the proposed approach employs a fully distributed communication method based on the population protocol model. In the proposed solution, event information is shared only after being filtered based on the reputation of communicating vehicles, thus ensuring the reliability of the event detection algorithm.

The solution proposed here was compared with a state-of-the-art system and the results of the experiments, performed on real traces, demonstrated its superiority with respect to all metrics considered, resulting in an improvement of up to 30% in the F1-score.

Acknowledgments. This research is partially funded by the S6 Project (PO FESR Sicilia 2014/2020) and partially funded by the European Union - PON Ricerca e Innovazione 2014-2020 - DM 1062/2021.

References

1. Yoshizawa, T., et al.: A survey of security and privacy issues in v2x communication systems. ACM Comput. Surv. **55**(9), 1–36 (2023). https://doi.org/10.1145/3558052
2. Singh, S., Agrawal, S.: VANET routing protocols: issues and challenges. In: 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1–5 (2014)
3. Agate, V., De Paola, A., Lo Re, G., Morana, M.: A platform for the evaluation of distributed reputation algorithms. In: 2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 1–8. IEEE (2018)
4. Huang, X., Yu, R., Kang, J., Zhang, Y.: Distributed reputation management for secure and efficient vehicular edge computing and networks. IEEE Access **5**, 25408–25420 (2017)
5. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 97–120. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-89707-1_5
6. Agate, V., De Paola, A., Lo Re, G., Morana, M.: Vulnerability evaluation of distributed reputation management systems. In: Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools, ValueTools 2016, Taormina, Italy, pp. 235–242 (2017). https://doi.org/10.4108/eai.25-10-2016.2266868. ISBN 9781631901416
7. Agate, V., De Paola, A., Lo Re, G., Morana, M.: DRESS: a distributed RMS evaluation simulation software. Int. J. Intell. Inf. Technol. (IJIIT) **16**(3), 1–18 (2020)
8. Timilsina, A., Khamesi, A.R., Agate, V., Silvestri, S.: A reinforcement learning approach for user preference-aware energy sharing systems. IEEE Trans. Green Commun. Networking **5**, 1138–1153 (2021)
9. Alnasser, A., Sun, H., Jiang, J.: Recommendation-based trust model for vehicle-to-everything (v2x). IEEE Internet Things J. **7**(1), 440–450 (2019)
10. Tan, S., Li, X., Dong, Q.: A trust management system for securing data plane of ad-hoc networks. IEEE Trans. Veh. Technol. **65**(9), 7579–7592 (2015)
11. Li, W., Song, H.: Art: an attack-resistant trust management scheme for securing vehicular ad hoc networks. IEEE Trans. Intell. Transp. Syst. **17**(4), 960–969 (2015)
12. Hu, H., Lu, R., Zhang, Z., Shao, J.: Replace: a reliable trust-based platoon service recommendation scheme in VANET. IEEE Trans. Veh. Technol. **66**(2), 1786–1797 (2016)
13. Mahmood, A., Butler, B., Zhang, W.E., Sheng, Q.Z., Siddiqui, S.A.: A hybrid trust management heuristic for VANETs. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 748–752 (2019)
14. Ahmad, F., Kurugollu, F., Kerrache, C.A., Sezer, S., Liu, L.: NOTRINO: a novel hybrid trust management scheme for internet-of-vehicles. IEEE Trans. Veh. Technol. **70**(9), 9244–9257 (2021)

15. Kerrache, C.A., Lagraa, N., Calafate, C.T., Cano, J.C., Manzoni, P.: T-VNets: a novel trust architecture for vehicular networks using the standardized messaging services of ETSI ITS. Comput. Commun. **93**, 68–83 (2016)
16. Agate, V., De Paola, A., Gaglio, S., Lo Re, G., Morana, M.: A framework for parallel assessment of reputation management systems. In: Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, pp. 121–128 (2016)
17. Ahmad, F., Kurugollu, F., Adnane, A., Hussain, R., Hussain, F.: Marine: man-in-the-middle attack resistant trust model in connected vehicles. IEEE Internet Things J. **7**(4), 3310–3322 (2020)
18. Mahmood, A., Sheng, Q.Z., Zhang, W.E., Wang, Y., Sagar, S.: Toward a distributed trust management system for misbehavior detection in the internet of vehicles. ACM Trans. Cyber-Phys. Syst. **7**(3), 1–25 (2023). https://doi.org/10.1145/3594637
19. Hussain, R., Lee, J., Zeadally, S.: Trust in VANET: a survey of current solutions and future research opportunities. IEEE Trans. Intell. Transp. Syst. **22**(5), 2553–2571 (2020)
20. Agate, V., Ferraro, P., Gaglio, S.: A cognitive architecture for ambient intelligence systems. In: 6th International Workshop on Artificial Intelligence and Cognition, AIC 2018. CEUR Workshop Proceedings, vol. 2418, pp. 52–58 (2019)
21. Agate, V., Concone, F., Ferraro, P.: A resilient smart architecture for road surface condition monitoring. In: Ben Ahmed, M., Boudhir, A.A., Karaş, İR., Jain, V., Mellouli, S. (eds.) SCA 2021. LNNS, vol. 393, pp. 199–209. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94191-8_16
22. Mistareehi, H.: Message dissemination scheme for rural areas using VANET (hardware implementation). In: 2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 120–125 (2021)
23. Ester, M., Kriegel, H.P., Sander, J., Xu, X., Sagar, S.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd, vol. 96, pp. 226–231 (1996)
24. Bordonaro, A., Concone, F., De Paola, A., Lo Re, G., Das, S.K.: Modeling efficient and effective communications in VANET through population protocols. In: 2021 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 305–310 (2021)
25. Bordonaro, A., De Paola, A., Lo Re, G.: VPP: a communication schema for population protocols in VANET. In: 2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS), pp. 11–18 (2021)
26. Sommer, C., German, R., Dressler, F.: Bidirectionally coupled network and road traffic simulation for improved IVC analysis. IEEE Trans. Mobile Comput. (TMC) **10**(1), 3–15 (2011)
27. Lopez, P.A., et al.: Microscopic traffic simulation using sumo. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2575–2582. IEEE (2018)
28. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems (2010)
29. Agate, V., De Paola, A., Lo Re, G., Morana, M.: A simulation software for the evaluation of vulnerabilities in reputation management systems. ACM Trans. Comput. Syst. (TOCS) **37**(1–4), 1–30 (2021)

30. Crapanzano, C., Milazzo, F., De Paola, A., Lo Re, G.: Reputation management for distributed service-oriented architectures. In: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop, pp. 160–165. IEEE (2010)
31. Agate, V., De Paola, A., Lo Re, G., Morana, M.: A simulation framework for evaluating distributed reputation management systems. In: Distributed Computing and Artificial Intelligence, 13th International Conference. AISC, vol. 474, pp. 247–254. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40162-1_27
32. Khatri, N., Lee, S., Mateen, A., Nam, S.Y.: Event message clustering algorithm for selection of majority message in VANETs. IEEE Access **11**, 14621–14635 (2023)

Reputation Systems for Supply Chains: The Challenge of Achieving Privacy Preservation

Lennart Bader[1], Jan Pennekamp[2(✉)], Emildeon Thevaraj[2], Maria Spiß[3], Salil S. Kanhere[4], and Klaus Wehrle[2]

[1] Cyber Analysis and Defense, Fraunhofer FKIE, Wachtberg, Germany
[2] Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany
jan.pennekamp@comsys.rwth-aachen.de
[3] Institute for Industrial Management at RWTH Aachen University, Aachen, Germany
[4] School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia

Abstract. Consumers frequently interact with reputation systems to rate products, services, and deliveries. While past research extensively studied different conceptual approaches to realize such systems securely and privacy-preservingly, these concepts are not yet in use in business-to-business environments. In this paper, (1) we thus outline which specific challenges privacy-cautious stakeholders in volatile supply chain networks introduce, (2) give an overview of the diverse landscape of privacy-preserving reputation systems and their properties, and (3) based on well-established concepts from supply chain information systems and cryptography, we further propose an initial concept that accounts for the aforementioned challenges by utilizing fully homomorphic encryption. For future work, we identify the need of evaluating whether novel systems address the supply chain-specific privacy and confidentiality needs.

Keywords: SCM · confidentiality · anonymity · voter · votee · FHE

1 Introduction and Motivation

In contrast to historically long-living business relationships, today's supply chain networks are much more volatile [47]. This shift manifests in both (i) the demand for more flexible, spontaneous and short-lived business agreements and (ii) a growing demand for digitized relationship establishments and management. While these aspects are well-known influencing factors for traditional supply

L. Bader and J. Pennekamp—These authors contributed equally to this work.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 464–475, 2024.
https://doi.org/10.1007/978-3-031-63989-0_24

chain management (SCM) [48], where various tracing systems tailored to the specific privacy and transparency needs of businesses within and along supply chain structures have been proposed [6,22], these solutions are not applicable for easing the business partner selection processes in such volatile supply chains.

For spontaneously establishing new relationships, potential business partners are in need of reliably assessing the credibility, i.e., *reputation*, of each other. Similarly, they have an incentive to advertise their own business and services to potential partners, where a well-built reputation might attract further business and sales. Since these use cases require businesses to transparently provide insights into their operations and potentially their former business relationships, confidentiality concerns on business and production secrets must be considered at all times when discussing and proposing respective technical solutions [48].

The concept of measuring a business' or product's reputation and granting access to it to potential customers or partners is well-established [28], e.g., for online sellers and their offered articles. However, since such reputations are mostly based on non-verifiable reviews, they can neither guarantee reliability nor accuracy, limiting their value for customers and the businesses themselves.

Hence, business-to-business (B2B)-focused reputation systems, as prevalent in the context of supply chains, exhibit a distinct set of requirements regarding (1) the reliability of ratings and the resulting reputation, (2) the transparency properties, (3) the privacy of involved stakeholders, and (4) the confidentiality of business information. Even though related work studies reputation systems [24, 27], they largely fail to consider the specific needs of B2B settings. In this paper, we thus raise the awareness for this research gap by identifying and discussing respective requirements along with potential conflicts. Moreover, we propose a first concept for a privacy-preserving multi-agent B2B reputation system that combines well-established concepts from supply chain information systems and cryptography for a flexible trade-off between these requirements.

2 Reputation Systems

Information *transparency* is an important aspect of today's supply chain networks [48]. Businesses have identified the various benefits of inter-business transparency for enhancing their collaboration for increased success of both their individual business and their supply chain as a whole [6]. However, *privacy* preservation is an equally important aspect as businesses want to keep their business relationships a secret. Consequently, generally maintaining the *confidentiality* of information (except for deliberately shared slices) as well as sufficient *security* mechanisms (e.g., to protect against unauthorized access as well as manipulation) are fundamental requirements for supply chain-oriented information systems.

2.1 Related Work: The Diversity in Today's Reputation Systems

The universal benefits of reputation systems have resulted in diverse approaches for various domains. Due to our focus on privacy preservation and confidentiality

in supply chains, we study relevant approaches from two well-known surveys [24, 27] in light of these requirements, i.e., we omit inapplicable approaches from our analysis. We augment this foundation with a selection of recent papers [3,36, 50,64] to provide an up-to-date overview. In Table 1 (Appendix), we detail the specific features of these approaches, which serve as the basis for our own design. Just like previous conclusions [24], we confirm that today's systems are unable to satisfy all desirable combinations of properties. Especially privacy properties often depend on the inclusion of at least one trusted third party, which is a challenging and potentially unrealistic assumption for supply chain settings.

2.2 Requirements for Privacy-Preserving Reputation Systems

The desire for inter-business transparency, aiming at an accountable selection of new business partners and increasing their own business' visibility, leads to the research area of privacy-preserving reputation systems [24,27]. These reputation systems adduce customer and business feedback in the form of *votes* to derive a business' *reputation* as a measure of its trustworthiness, quality of offered services, and ability to cooperate. Other customers and businesses can then request this reputation as a foundation for their follow-up decision-making.

Hence, businesses can take the role of a *voter* when submitting a vote or rating, a *votee* when receiving such ratings, or a *requester* when requesting access to the derived reputation score, as formalized by Gurtler and Goldberg [24]. Each business taking one or multiple roles introduces specific requirements to the reputation system. Straightforward requirements, i.e., the reliability and accountability of the reputation, the security and flexibility of submitted ratings, and the possibility for access control for reputation requests, are complemented by the demand for the complex trade-off between privacy and transparency [6], especially since "privacy" can have different notions depending on a business' role [24]. Common privacy requirements for voters are related to their anonymity and the anonymity of individual (*voter-vote privacy*) and even multiple (*two-vote privacy*) votes [24]. Further, votees might require privacy (confidentiality) regarding their reputation, either by proving their current reputation without the need to link themselves to a long-term history (*reputation-usage unlinkability*) or by providing meta-information on their current reputation, e.g., threshold-based, instead of their plain reputation score (*exact reputation blinding*) [24], raising reputation reliability challenges. Similarly, requesters might want to hide their identity when requesting another business' reputation, which, among other aspects, might contradict the votees' demand for sophisticated access control.

The variety of contradicting requirements poses a significant challenge for researchers when designing appropriate and practical reputation systems. Each assessment of the importance or trade-off weights of these requirements leads to different potential designs, such that a wide range of systems has been proposed.

3 Reputation and Privacy Preservation in Supply Chains

The lack of widely-used reputation systems in supply chain settings hints at issues with today's technical approaches (cf. Table 1 for general academic concepts). To better assess this inadequate situation, we now explore the circumstances of deploying and developing supply-chain-focused reputation systems.

3.1 Supply Chain-Induced Requirements

In contrast to commonly studied consumer-oriented reputation systems, e.g., seller or product ratings on online platforms, reputation systems for B2B use and supply chains face specific confidentiality and privacy needs, which render the application of existing reputation systems difficult or simply infeasible.

First, supply chains induce new *information flow dimensions* [48]. Combining subjective ratings with contract-based information and objective production- or service-related information for computing a reputation score requires suitable reputation functions, authenticity checks, and verification mechanisms.

Second, the volatility of modern supply chain networks is a big challenge because, e.g., systems with *votee-owned* [24] scores are not applicable when participants go out of business or just decide to stop participating. Likewise, selecting a trusted third party for managing the businesses' reputations that all participants agree on represents a significant challenge for globalized supply chains.

3.2 Research Gap: Domain-Specific Realization

Existing (privacy-preserving) reputation systems (cf. Sect. 2.1) fail to consider the previously outlined requirements regarding privacy, availability, and the supported data dimensions. For a formal definition of these dimensions in supply chains, we refer to Pennekamp et al. [48]. Hence, developing a sophisticated privacy-preserving design for all involved parties that does not require well-established trust between participants while also accounting for volatile structures remains an open issue in research. In the following, we outline promising design decisions for such a reputation system and detail its conceptual design.

4 Toward a Comprehensive Design

Based on the general requirements for privacy-preserving reputation systems and the additional challenges arising from the specifics of modern supply chain networks, we now derive a comprehensive design for a reputation system tailored to these challenging needs. Given our business focus, we know that all entities (operators, participants, and users) are bound by legislation and to specific jurisdictions. Hence, for this work, we consider malicious-but-cautious attackers [52] who can misbehave in all possible ways while trying to not leave any verifiable evidence of their misbehavior. With this attacker model, we have to include more

attacks than with honest-but-curious (semi-honest) attackers since it explicitly allows for local deviation from protocols unless they are provable by third parties. Consequently, collusion attacks are possible but attackers would have to make sure to not leave any (public) traces to comply with the attacker model.

4.1 Design Decisions

Achieving a reasonable trade-off between reputation privacy and the desired degree of transparency while maintaining a secure operation and accounting for volatile supply chain structures calls for (1) a sophisticated encryption scheme, (2) verifiable reputation computations, and (3) a distribution of competences to multiple independent entities (as we assume malicious-but-cautious adversaries).

First, we identify a *ticket-based* [24] voting process as well suited for combining privacy preservation with rating reliability: A business should be authorized to submit a vote for another business only if a (recent) business relationship existed. Hence, voting tickets can be issued as soon as a business relationship has been established, i.e., a new contract has been signed. The tickets can either be issued by the businesses themselves or by a third entity that reviews relationships within an already-deployed supply chain information system, such as the privacy-preserving realizations ProductChain [37] or PrivAccIChain [6].

Second, to account for the challenge of dealing with volatile supply chain structures (especially, defunct, i.e., departing, businesses), we assess that a *centralized* approach fits best to ensure a reliable operation of the system (availability of reputation scores). With a (conceptual) central and static entity, departing or non-collaborative nodes do not negatively affect the reputation system's operation. Using multiple entities for different roles and potentially splitting a single role onto multiple entities further reduces the risk of operational deficiencies.

Third, as we already encourage the use of *multiple independent entities* for different roles, we reduce the trust in a single entity and limit its ability to compromise information privacy and security, i.e., preventing the threat and impact of collusion attacks. While such an approach already achieves full collusion resistance, increasing the number of central entities further distributes critical responsibilities, which allows for a tunable degree of collusion resistance.

Fourth, to achieve privacy-preserving voting and request processes, we propose the use of *pseudonyms*. Instead of authorizing themselves to other businesses or the reputation engine, voters and requesters receive a (temporary) pseudonym from an independent or government-run entity to authorize their requests. Thus, individual requests cannot be linked to a specific business, and multiple requests from the same business appear as if they were coming from different entities, such that the businesses' privacy as well as privacy of relationship can be achieved, accounting for the needs of (volatile) supply chain networks.

Finally, the use of a (conceptually) central entity mandates an elaborate data security concept. Specifically, we propose to base the voting process as well as the reputation function on *fully homomorphic encryption* (FHE) [38]. In particular, voters encrypt their votes under FHE before submitting them to an available reputation engine, which is unable to decrypt or tamper with individual votes.

The reputation engine then computes the reputation score under FHE while offering a verification mechanism to both voters and votees. Since the result remains encrypted, the reputation engine does not learn the actual, potentially sensitive reputation score, ensuring the desired votee privacy.

To quickly explore the feasibility of FHE for our use in reputation systems, we measured the runtime of the basic operations needed in our design using Pyfhel [30], which is a Python library for Microsoft SEAL [40], utilizing the CKKS [16] scheme. We extrapolated these numbers to realistic scenarios, and after discussions with our applied partner, we concluded that FHE is suitable.

4.2 The Rating Process on a High-Level

To illustrate our design and the interaction of its entities, we now present the rating process consisting of multiple steps (cf. Fig. 1). For the rating, we combine and weigh subjective ratings with a rating based on objective information on the business relationship.

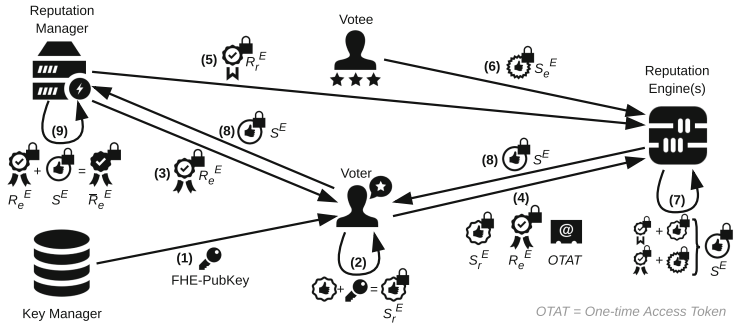


Fig. 1. The reputation calculation is shielded using FHE to ensure confidentiality.

During the rating process, both the *voter* and the *votee* use (temporary) pseudonyms for their interactions. The *voter* (1) retrieves the *votee*-specific FHE key from the *key manager* and (2) encrypts its rating with this key under FHE as S_r^E. The voter then (3) receives the (FH-encrypted) current reputation R_e^E of the *votee* from the *reputation manager* to (4) forward S_r^E, R_e^E, and a one-time *access token* to one of the available *reputation engines*. The *reputation engine* then (5) uses the access token to receive the *voter's* current reputation R_r^E from the *reputation manager* and (6) optionally receives a data-backed (which ideally is verifiable, e.g., since it has been processed using trusted computing [46]) self-rating S_e^E from the *votee* to (7) combine S_r^E and S_e^E into the final rating S^E. Here, the current reputations R_r^E and R_e^E can be used as weights for the ratings. The *reputation engine* then (8) signs and submits the new rating to the *voter*, who forwards the rating S back to the *reputation manager*. Here, the new reputation (9) is calculated as $\bar{R}_e^E$ using R_e^E and S^E.

The use of FHE ensures that neither the reputation engine nor the reputation manager can access individual ratings or a business' plain reputation. By distributing the reputation computation to multiple entities, i.e., *reputation engines*, we maintain privacy of relationship between voter and votee toward the *reputation manager*. The use of the one-time *access token* further prevents the *reputation manager* from linking *voter* and *votee* by request analyses, as the current reputations are requested by different entities.

5 Conclusion and the Road Ahead

In this paper, we have highlighted the lack of reputation systems in B2B settings and volatile supply chain networks despite the availability of technical approaches, indicating a mismatch between offered features and set requirements.

Evaluation Challenges. Generally, this research is hindered by the lack of evaluation data and realistic supply chain models [48]. While our applied partner (Institute for Industrial Management) can provide us with real-world data, its scale and the complexity of business relationships represent only the tip of the iceberg. Specifically, at this point, we rely on data from ERP systems that cover the order and delivery of goods. Moreover, we have access to monitoring data of production machines that allow us to incorporate such data into the calculation of reputation. Regardless, we are interested in acquiring additional datasets to extensively evaluate reputation systems for volatile supply chain networks.

Future Work. In addition to evaluating our proposed approach, we further identify three crucial research directions: (1) Assessing whether multi-key FHE can offer confidentiality benefits, (2) studying how to better utilize existing supply chain information systems, and (3) investigating the implications of dealing with E2E-secured supply chains [46], i.e., incorporating trusted sensors into the reputation computation and capitalizing on their reliability benefits.

Conclusion. The lack of deployed reputation systems in B2B environments encouraged us to look into the respective reasons since reliable reputation scores are beneficial for businesses when managing both short- and long-term operations. Even though privacy-preserving concepts are available, their use is hindered by volatile supply chain networks and differing data dimensions. Addressing this research gap, we are proposing an FHE-based design that accounts for the transparency, confidentiality, and privacy requirements of participating businesses while being compatible with existing supply chain information systems.

Acknowledgments. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612 and the Alexander von Humboldt (AvH) Foundation.

A Appendix: Comparing Reputation Systems in Detail

As a foundation of our work (cf. Sect. 2.1), we have analyzed various reputation systems, especially those covered in previous surveys [24,27], regarding their system architecture and their properties for both feedback provision and the reputation itself. In Table 1, we summarize the corresponding results.

For the architecture, we distinguish between *centralized* (C), *decentralized* (D), and *hybrid* (H) approaches. Additionally, depending on the system, voters can provide feedback values from different sets or value ranges. Here, we indicate the set of available values either as a set (e.g., $\mathbb{R}$ or $\{1, ..., 5\}$) or as an interval. Some systems also allow text and vectors as feedback. Moreover,

Table 1. We analyzed privacy-preserving reputation systems that are promising for use in supply chains regarding nine distinct properties. We represent no level of fulfillment of the respective property by ○ while ● shows fulfillment of the property. For entries labeled with "?", we cannot reliably identify the corresponding aspect.

Publication	Architecture	Set/Range	Granularity	Set/Range	Liveliness	Visibility	Durability	Non-Monotonicity	Aggregation Model
System		Feedback		Reputation					
SMPC-based Systems									
Pavlov et al. (2004) [44]	D	$\mathbb{R}$	M	$\mathbb{R}[0,1]$	●	L	○	●	Sum, Beta reputation
Yao et al. (2007) [62]	D	$\mathbb{Z}$	M	$\mathbb{Z}$	●	L	○	○	Weighted average
Gudes et al. (2009) [23]	D	$\mathbb{R}$	M	$\mathbb{R}$	●	L	○	●	Weighted sum, Mean
Melchor et al. (2009) [39]	D	$\mathbb{Z}$	M	$\mathbb{Z}$	●	L	○	○	Weighted average
Nithyanand and Raman (2009) [42]	D	$\mathbb{R}\{0,1\}$	M	$\mathbb{R}$	●	L	○	●	Ordered weighted average
Gal-Oz et al. (2010) [21]	D	$\mathbb{R}$	M	$\mathbb{R}$	●	L	○	●	Weighted sum, Mean
Dolev et al. (2014) [20]	D	$\{1, ..., 10\}$	M	$\mathbb{R}$	●	L	○	●	Weighted mean
Clark et al. (2016) [18]	D	$[0, v_{max}]$	M	$[0, v_{max}]$	●	L	○	●	Mean
Azad et al. (2018) [4]	D	$\{-1, 0, 1\}$	M	$\{-1, 0, 1\}$	●	G	●	●	Weighted sum
Bakas et al. (2020) [8]	D	$\{1, ..., 5\}$	M	$\mathbb{Z}$	●	G	○	○	Weighted sum
Token-based Systems									
Singh and Liu (2003) [56]	H	?	M	?	?	G	●	?	?
Voss (2004) [59]	C	$\mathbb{Z}$	S	$\mathbb{Z}$	●	G	●	●	Sum
Androulaki et al. (2008) [2]	C	$\{0, 1\}$	S	$\mathbb{Z}$	○	G	●	○	Sum
Kerschbaum (2009) [31]	C	$\{0, 1\}$	S	$[0, 1]$	●	G	●	●	Beta reputation
Schiffner et al. (2009) [54]	C	$\{-1, 1\}$	S	$\mathbb{Z}$	●	G	●	●	Sum
Hussain and Skillicorn (2011) [29]	C	?	M	?	?	G	●	?	Open
Schiffner et al. (2011) [55]	C	$\{-, +\}$	S	$\mathbb{R}$	●	G	●	●	Open
Zhang et al. (2014) [63]	H	?	S	$\mathbb{R}$	●	G	●	●	Open
Bazin et al. (2016) [9]	H	?	S	?	?	G	●	?	Open, Beta reputation
Busom et al. (2017) [15]	C	Text	S	?	●	G	●	●	Union
Blömer et al. (2018) [13]	C	?	S	?	?	G	?	?	Open
Liu and Manulis (2019) [35]	C	$\{1, ..., 5\}$	M	$\mathbb{Z}$	●	G	●	○	Sum
Proxy-based Systems									
Ries et al. (2011) [51]	C	$\{0, 1\}$	M	$[0, 1]$	●	L	○	●	Beta reputation
Petrlic et al. (2014) [49]	C	Vector$\{0, 1\}$	S	$\mathbb{Z}$	●	G	●	?	Sum
Bethencourt et al. (2010) [12]	H	$\{0, 1\}$	S	$\mathbb{Z}$	●	G	●	○	Sum
Lajoie-Mazenc et al. (2015) [33]	H	$\{-, +\}\mathbb{Z}$	S	$\mathbb{R}$	●	G	●	●	Open
Transitory Pseudonym-based Systems									
Miranda and Rodrigues (2006) [41]	C	?	S	?	●	G	●	●	Open
Steinbrecher (2006) [58]	C	?	S	?	●	G	●	●	Open
Hao et al. (2007) [26]	D	$\{-1, 1\}$	S	$\mathbb{Z}$	●	G	●	●	Sum
Hao et al. (2008) [25]	C	$\{-1, 1\}$	M	$\mathbb{R}$	●	G	●	●	Sum, Average
Wei and He (2009) [60]	C	$\{-1, 1\}$	M	$\mathbb{R}$	●	G	●	●	Sum, Average
Peng et al. (2010) [45]	C	?	S	?	?	G	●	?	Open
Anceaume et al. (2013) [1]	D	$[0, 1]$	S	$[0, 1]$	●	G	?	●	Beta reputation
Christin et al. (2013) [17]	C	?	S	?	●	G	●	●	Open
Clauß et al. (2013) [19]	?	$[1, ...)$	M	$[1, ...)$	?	G	?	?	Open
Blockchain-based Systems									
Schaub et al. (2016) [53]	D	$\mathbb{Z}$	S	$\mathbb{R}$	●	G	○	●	Open
Soska et al. (2016) [57]	D	Text$\mathbb{Z}$	M	?	?	G	●	?	Open
Bazin et al. (2017) [10]	D	$\mathbb{Z}$	S	$\mathbb{R}$	●	G	●	●	Open
Azad et al. (2018) [5]	D	$\{-, +\}$	S	$\mathbb{Z}$	●	G	○	●	Beta reputation
Bag et al. (2018) [7]	D	$\{0, 1\}$	M	$[1, 10]$	○	L	○	●	Mean
Bemmann et al. (2018)[11]	C	$\{0, 1\}$	M	?	?	G	●	?	?
Owiyo et al. (2018) [43]	D	?	S	?	?	G	●	?	Open
Liu et al. (2019) [34]	C	$[1, 10]$	S	$\mathbb{N}$	●	G	●	○	Sum
Malik et al. (2019) [36]	D	?	M	?	?	G	●	?	Mean, Median, Beta reputation
Zhou et al. (2021) [64]	D	$[-5, -1]$,$[1, 5]$	M	$[0, 1]$	○	G	●	●	Weighted sum
Arshad et al. (2022) [3]	D	$\{0, 1\}$	M	$\mathbb{R}$	○	G	●	●	Beta reputation, Open
Putra et al. (2022) [50]	D	$\{-1, 1\}$	M	$\mathbb{R}$	●	G	●	●	Weighted average
Other Systems									
Kinateder and Pearson (2003) [32]	D	$[0, 1]$	S	$\mathbb{R}$	●	L	○	●	Open
Bo et al. (2007) [14]	H	?	S	?	●	G	●	●	Open
Yang et al. (2023) [61]	C	$\{-1, 1\}$	M	$\mathbb{Z}$	●	G	●	●	Sum

the feedback granularity indicates whether feedback is provided for a *single* (S) interaction between voter and votee or over *multiple* (M) ones.

For reputation, we further cover six distinct properties per approach. While the set or range is equivalent to the respective feedback property, liveliness [54] indicates whether negative ratings are allowed by the system. This aspect is also related to monotonicity, which indicates that a votee's reputation can only increase over time. Hence, non-monotonicity allows for reputations to decrease over time or when negative ratings are submitted. Some reputation systems provide *global* (G) visibility, while others offer *local* (L) visibility. With global visibility, all requesting parties receive the same response to a reputation query. With local visibility, different requesters can receive different results. The durability indicates whether ratings are stored locally or whether the reputation has to be recalculated on every request, resulting in a trade-off between storage and computation requirements. Finally, a reputation system's aggregation model indicates how feedback is aggregated into a final reputation score.

References

1. Anceaume, E., Guette, G., Lajoie-Mazenc, P., Sirvent, T., Viet Triem Tong, V.: Extending signatures of reputation. In: Hansen, M., Hoepman, J.-H., Leenes, R., Whitehouse, D. (eds.) Privacy and Identity 2013. IAICT, vol. 421, pp. 165–176. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55137-6_13
2. Androulaki, E., Choi, S.G., Bellovin, S.M., Malkin, T.: Reputation systems for anonymous networks. In: PETS (2008)
3. Arshad, J., Azad, M.A., Prince, A., Ali, J., Papaioannou, T.G.: REPUTABLE–A decentralized reputation system for blockchain-based ecosystems. IEEE Access **10**, 79948–79961 (2022)
4. Azad, M.A., Bag, S., Hao, F.: M2M-REP: reputation of machines in the internet of things. In: ARES (2017)
5. Azad, M.A., Bag, S., Hao, F.: PrivBox: verifiable decentralized reputation system for online marketplaces. Future Gener. Comput. Syst. **89**, 44–57 (2018)
6. Bader, L., et al.: Blockchain-based privacy preservation for supply chains supporting lightweight multi-hop information accountability. Inf. Process. Manag. **58**(3), 102529(2021)
7. Bag, S., Azad, M.A., Hao, F.: A privacy-aware decentralized and personalized reputation system. Comput. Secur. **77**, 514–530 (2018)
8. Bakas, A., Michalas, A., Ullah, A.: (F)unctional sifting: a privacy-preserving reputation system through multi-input functional encryption. In: Asplund, M., Nadjm-Tehrani, S. (eds.) NordSec 2020. LNCS, vol. 12556, pp. 111–126. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70852-8_7
9. Bazin, R., Schaub, A., Hasan, O., Brunie, L.: A decentralized anonymity-preserving reputation system with constant-time score retrieval. Cryptology ePrint Archive 2016/416 (2016)
10. Bazin, R., Schaub, A., Hasan, O., Brunie, L.: Self-reported Verifiable reputation with rater privacy. In: IFIPTM (2017)
11. Bemmann, K., et al.: Fully-featured anonymous credentials with reputation system. In: ARES (2018)

12. Bethencourt, J., Shi, E., Song, D.: Signatures of reputation. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 400–407. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14577-3_35
13. Blömer, J., Eidens, F., Juhnke, J.: Practical, anonymous, and publicly linkable universally-composable reputation systems. In: CT-RSA (2018)
14. Bo, Y., Min, Z., Guohuan, L.: A reputation system with privacy and incentive. In: ACIS SNPD (2007)
15. Busom, N., Petrlic, R., Sebé, F., Sorge, C., Valls, M.: A privacy-preserving reputation system with user rewards. J. Netw. Comput. Appl. **80**, 58–66 (2017)
16. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT (2017)
17. Christin, D., Roßkopf, C., Hollick, M., Martucci, L.A., Kanhere, S.S.: IncogniSense: an anonymity-preserving reputation framework for participatory sensing applications. Pervasive Mob. Comput. **9**(3), 353–371 (2013)
18. Clark, M.R., Stewart, K., Hopkinson, K.M.: Dynamic, privacy-preserving decentralized reputation systems. IEEE Trans. Mob. Comput. **16**(9), 2506–2517 (2016)
19. Clauß, S., Schiffner, S., Kerschbaum, F.: k-Anonymous reputation. In: ACM ASIACCS (2013)
20. Dolev, S., Gilboa, N., Kopeetsky, M.: Efficient private multi-party computations of trust in the presence of curious and malicious users. J. Trust Manag. **1**, 1–21 (2014)
21. Gal-Oz, N., Grinshpoun, T., Gudes, E.: Sharing reputation across virtual communities. J. Theor. Appl. Electron. Commer. Res. **5**(2), 1–25 (2010)
22. Gonczol, P., Katsikouli, P., Herskind, L., Dragoni, N.: Blockchain implementations and use cases for supply chains-a survey. IEEE Access **8**, 11856–11871 (2020)
23. Gudes, E., Gal-Oz, N., Grubshtein, A.: Methods for computing trust and reputation while preserving privacy. In: Gudes, E., Vaidya, J. (eds.) DBSec 2009. LNCS, vol. 5645, pp. 291–298. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03007-9_20
24. Gurtler, S., Goldberg, I.: SoK: privacy-preserving reputation systems. Proc. Priv. Enhancing Technol. **2021**(1), 107–127 (2021)
25. Hao, L., Lu, S., Tang, J., Zhang, A.: A low cost and reliable anonymity scheme in P2P reputation systems with trusted third parties. In: IEEE GLOBECOM (2008)
26. Hao, L., Yang, S., Lu, S., Chen, G.: A dynamic anonymous P2P reputation system based on trusted computing technology. In: IEEE GLOBECOM (2007)
27. Hasan, O., Brunie, L., Bertino, E.: Privacy-preserving reputation systems based on blockchain and other cryptographic building blocks: a survey. ACM Comput. Surv. **55**(2), 1–37 (2022)
28. Hendrikx, F., Bubendorfer, K., Chard, R.: Reputation systems: a survey and taxonomy. J. Parallel Distrib. Comput. **75**, 184–197 (2015)
29. Hussain, M., Skillicorn, D.B.: Mitigating the linkability problem in anonymous reputation management. J. Internet Serv. Appl. **2**, 47–65 (2011)
30. Ibarrondo, A.: Pyfhel (2017). https://github.com/ibarrond/Pyfhel
31. Kerschbaum, F.: A Verifiable, Centralized. ACM WPES, Coercion-Free Reputation System. In (2009)
32. Kinateder, M., Pearson, S.: A privacy-enhanced peer-to-peer reputation system. In: EC-Web (2003)
33. Lajoie-Mazenc, P., Anceaume, E., Guette, G., Sirvent, T., Tong, V.V.T.: Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratings. hal-01104837 (2015)

34. Liu, D., Alahmadi, A., Ni, J., Lin, X., Shen, X.: Anonymous reputation system for IIoT-enabled retail marketing Atop PoS blockchain. IEEE Trans. Industr. Inform. **15**(6), 3527–3537 (2019)
35. Liu, J., Manulis, M.: pRate: anonymous star rating with rating secrecy. In: ACNS (2019)
36. Malik, S., Dedeoglu, V., Kanhere, S.S., Jurdak, R.: TrustChain: trust management in blockchain and IoT supported supply chains. In: IEEE Blockchain (2019)
37. Malik, S., Kanhere, S.S., Jurdak, R.: ProductChain: scalable blockchain framework to support provenance in supply chains. In: IEEE NCA (2018)
38. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., et al.: Survey on fully homomorphic encryption, theory, and applications. Proc. IEEE **110**(10), 1572–1609 (2022)
39. Melchor, C.A., Ait-Salem, B., Gaborit, P.: A collusion-resistant distributed scalar product protocol with application to privacy-preserving computation of trust. In: IEEE NCA (2009)
40. Microsoft, Inc.: Microsoft SEAL. https://github.com/Microsoft/SEAL (2018)
41. Miranda, H., Rodrigues, L.: A framework to provide anonymity in reputation systems. In: MobiQuitous (2006)
42. Nithyanand, R., Raman, K.: Fuzzy privacy preserving peer-to-peer reputation management. Cryptology ePrint Archive 2009/442 (2009)
43. Owiyo, E., Wang, Y., Asamoah, E., Kamenyi, D., Obiri, I.: Decentralized privacy preserving reputation system. In: IEEE DSC (2018)
44. Pavlov, E., Rosenschein, J.S., Topol, Z.: Supporting privacy in decentralized additive reputation systems. In: iTrust (2004)
45. Peng, H., Lu, S., Zhao, D., Zhang, A.: Low cost and reliable anonymity protocols in P2P reputation systems. J. Shanghai Jiaotong Univ. (Sci.) **15**, 207–212 (2010)
46. Pennekamp, J., Alder, F., Matzutt, R., Mühlberg, J.T., et al.: Secure end-to-end sensing in supply chains. In: IEEE CPS-Sec (2020), proceedings of the 5th International Workshop on Cyber-Physical Systems Security (CPS-Sec 2020) (2020)
47. Pennekamp, J., Henze, M., Schmidt, S., Niemietz, P., et al.: Dataflow challenges in an *Internet* of production: a security & privacy perspective. In: ACM CPS-SPC (2019)
48. Pennekamp, J., Matzutt, R., Klinkmüller, C., Bader, L., et al.: An interdisciplinary survey on information flows in supply chains. ACM Comput. Surv. **56**(2), 1–38 (2024)
49. Petrlic, R., Lutters, S., Sorge, C.: Privacy-preserving reputation management. In: ACM SAC (2014)
50. Putra, G.D., Kang, C., Kanhere, S.S., Hong, J.W.K.: DeTRM: decentralised trust and reputation management for blockchain-based supply chains. In: IEEE ICBC (2022)
51. Ries, S., Fischlin, M., Martucci, L.A., Muuhlhauser, M.: Learning whom to trust in a privacy-friendly way. In: IEEE TrustCom (2011)
52. Ryan, M.D.: Enhanced certificate transparency and end-to-end encrypted mail. In: NDSS (2014)
53. Schaub, A., Bazin, R., Hasan, O., Brunie, L.: A trustless privacy-preserving reputation system. In: SEC (2016)
54. Schiffner, S., Clauß, S., Steinbrecher, S.: Privacy and liveliness for reputation systems. In: EuroPKI (2009)
55. Schiffner, S., Clauß, S., Steinbrecher, S.: Privacy, Liveliness and Fairness for Reputation. In: SOFSEM (2011)

56. Singh, A., Liu, L.: TrustMe: anonymous management of trust relationships in decentralized P2PSystems. In: P2P (2003)
57. Soska, K., Kwon, A., Christin, N., Devadas, S.: Beaver: a decentralized anonymous marketplace with secure reputation. Cryptology ePrint Archive 2016/464 (2016)
58. Steinbrecher, S.: Design options for privacy-respecting reputation systems within centralised internet communities. In: SEC (2006)
59. Voss, M.: Privacy preserving online reputation systems. In: SEC (2004)
60. Wei, Y., He, Y.: A pseudonym changing-based anonymity protocol for P2P reputation systems. In: ETCS (2009)
61. Yang, X., Yang, X., Luo, J., Yi, X., et al.: Towards sustainable trust: a practical SGX aided anonymous reputation system. IEEE Trans. Sustain. Comput. **9**, 89–99 (2023)
62. Yao, D., Tamassia, R., Proctor, S.: Private distributed scalar product protocol with application to privacy-preserving computation of trust. In: IFIPTM (2007)
63. Zhang, K., Li, Z., Yang, Y.: A reputation system preserving the privacy of feedback providers and resisting Sybil attacks. Int. J. Multimedia Ubiquitous Eng. **9**(2), 141–152 (2014)
64. Zhou, Z., Wang, M., Yang, C.N., Fu, Z., et al.: Blockchain-based decentralized reputation system in E-commerce environment. Future Gener. Comput. Syst. **124**, 155–167 (2021)

Data Management in Appendable-Block Blockchains: A Case Study for IT Life-Cycle Management

Rodrigo W. Silveira[1,2(✉)], Roben C. Lunardi[3], and Avelino F. Zorzo[1]

[1] Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
r.wetzel@edu.pucrs.br, avelino.zorzo@pucrs.br
[2] Hewlett-Packard Brazil (HP Brazil), Porto Alegre, Brazil
[3] Federal Institute of Education, Science and Techlongy of Rio Grande do Sul (IFRS), Porto Alegre, Brazil
roben@lunardi.inf.br

Abstract. Blockchain has been adopted in multiple areas that require secure and decentralized storage of information. One of them is the management of device lifecycle, which typically employs a centralized storage model. However, the application of traditional blockchains in this context has been growing and presenting challenges, particularly in terms of latency, performance, and scalability. Consequently, new blockchain topologies have been developed to enhance these properties by altering their data structure. An example of this is the appendable-block blockchain model, which removes transaction insertion of block insertion, allowing new data to be added to an auxiliary chain from the main block. The main goal of this work is to expand appendable-block blockchain, addressing contributions and enhancements in storage, persistence, and efficient data access. Besides, this work aims to investigate the process of information retrieval for related lifecycle events from the same device or component, i.e., within the same context. In order to achieve this, we present a real-use case study of a patent that addresses the storage and retrieval of data related to the lifecycle of devices within a company.

Keywords: Blockchain · Appendable-block blockchain · IT management · Lifecycle events

1 Introduction

Information security is a topic that is becoming increasingly important in the realm of technology, mainly because applications and systems are getting

This paper was supported by: the Ministry of Science, Technology, and Innovations, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, Residência em TIC 02 - 01245.012095/2020-56; IFRS and HP Inc; and call INCT - MCTI/CNPq/CAPES/FAPs nº16/2014; CAPES, Brazil (Finance code 001), CNPq, Brazil (465450/2014-8 and 306250/2021-7) and FAPERGS, Brazil (17/2551-0000520-1).

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 476–488, 2024.
https://doi.org/10.1007/978-3-031-63989-0_25

progressively more interconnected. One of the areas in which security plays a fundamental role is in information technology (IT) management [3]. In 1994, Charles B. Wang warned that companies should apply investments and reformulations in order to promote advantages over competitors in this type of application [19]. IT management allows companies to manage their resources, providing the necessary tools for accessing and storing essential information for their management or providing a [4] service.

Management in information technology services [2] is a huge area that encompasses several sections of computing. One of them is the management of device lifecycle events. Lifecycle events are all the telemetry collected from machines, from their assembly to their disposal. This data usually contains information about the device's health and performance. The gathering of this information is essential for companies to check the condition of their products, especially when they are being used by consumers. This is vital to check eventual failures and make improvements. The management has many safety and compliance challenges that must be dealt with in increasing importance, especially in the security area. Consequently, companies usually adopt the Information Technology Infrastructure Library (ITIL) [7] standard, used as a set of best practices for managing information technology-enabled services.

Therefore, there has emerged a need for a model that guarantees availability, security, transparency, and immutable and distributed storage of data. Blockchain was developed with these characteristics, providing such features in a cryptographically secure system, enabling unknown entities to trust the same network without the need for a controlling third party. Trust is ensured through a consensus mechanism [18]. Many proposals use blockchain in different types of systems, making it increasingly common in recent years. One of these areas is IT management [2].

In 2019, Rafael Zotto proposed a patent [20] that suggests storing data and information collected from HP (Hewlett-Packard) machines (computers, printers, and notebooks, for example) in a distributed blockchain. Hence, it is possible to create a system that allows the management lifecycle information of devices on demand by the end user, at any time. The purpose is to analyze the status of their device and prevent failures at any time. However, the proposed model does not have an application in a specific blockchain and also does not indicate a single system for its implementation. Additionally, appendable-block blockchain model [10,12,14] (also known as SpeedyChain) seems to be a promising solution for storing IT lifecycle information. In special, due to its capability to store in different contexts [11,13] and allow retrieval of this information from another device connected to the same blockchain.

The proposed work uses the mentioned patent [20] as a case study for the implementation of improvements in the SpeedyChain storage and recovery mechanisms. This work proposes an improvement in SpeedyChain, allowing to addition of new transaction chains for each component of the device, instead of a single chain in the current SpeedyChain model. Every new lifecycle event collected will be stored in its own chain of transactions, according to the device and component it represents. This new structure aims to improve the perfor-

mance of data recovery of specific components and improve the registration of component data after an exchange between devices. Additionally, the results of the evaluation show that the proposed structure can be advantageous for data recovery in contexts of different chains.

2 Related Work

In the last few years, some companies have provided a renting device service called Device-as-a-Service (DaaS) [16] or sometimes called Machine-as-a-Service (MaaS). The main concept of this service is to offer a way for smaller companies to rent a specific device instead of buying it, so the company that provides it needs to have a way to manage this device, especially its use and current health through lifecycle events collected. Therefore it is necessary to safely store and manage these events, since the data collected is sensitive and private, some solutions are adopting the use of blockchain for this matter, relying on its properties for better management.

Chen et al. [5] work provides a guide for the research and implementation of blockchain in Product Lifecycle Management (PLM). The research presents a structure organized into five metrics in order to determine whether the target PLM application is suitable to be solved using a blockchain model. Those metrics are related to data security, the requirement of consensus algorithms, data traceability, and the requirement of automated transactions. Also, it discusses possible applications for the lifecycle events management using blockchain. As users are constantly changing, it becomes difficult to determine the corresponding fees based on product usage. This technology can be used to record each user's information, including use duration, power consumption, and device health. Based on the collected information, fees can be automatically calculated and settled by smart contracts. Secondly, users can understand the complete history of the product before deciding whether to share or use the device.

Some researchers [9,17] proposed the usage of blockchain to manage all steps for device lifecycle management in an industry-based PLM, those stages are: beginning of life (BOL), middle of life (MOL), and end of life (EOL). In the MOL stage, it shows an example of using blockchain for the logistics of product and product proactive maintenance. According to lifecycle data collected in real-time, smart contracts can be used to detect occasional failures or faulty devices, alerting maintainers to replace malfunctioning devices.

Steamchain [8] has created a blockchain-based system to manage payments and decentralize information, facilitating interaction between original equipment manufacturers (OEMs) and users. This technology is provided as a service to other companies seeking to store the information of their products, facilitating communication between both parties. Therefore, companies that contract the service can utilize the software to measure a variety of device productivity parameters in real time and use them according to their needs. This solution can be used for Machine-as-a-Service, working as a device fleet management.

Although blockchain is not a new approach to solving PLM, those solutions still lack a deeper discussion on the storage and recovery steps. In the next sec-

tions, we present a new approach to managing lifecycle events using the concept of appendable-block blockchain. The main idea is to use SpeedyChain properties to create auxiliary chains for each device component, so the data for a specific context can be recovered easily, with a better performance compared to a traditional chain. Another improvement of this solution would be when trading a component between devices, since the component chain of events is separated, it can follow the component to its new chain in the new device.

3 Background

3.1 Appendable-Block Blockchain

Appendable-block blockchain is a concept used by SpeedyChain framework [12, 14], which was designed for the context of IoT systems. Usually, blockchains for this kind of application have low computing power and limited storage capacity, making it difficult to use a traditional blockchain architecture. To improve those limitations, appendable-block blockchain proposes a model that removes the concept of static blocks, allowing parallel transaction insertion and enabling faster insertion of data into the block [11,13].

The chain structure is composed of three main components: devices, gateways, and service providers. Devices are responsible for producing data and sending them to the gateways, which append these values to the blockchain. Devices do not store blockchain data, they behave like a light node in the chain. Gateways control the access and insertions in the blockchain. Service Providers work as middleware, connecting to the gateways to access the blockchain information.

The SpeedyChain node structure allows the insertion of transactions after a block is inserted in the blockchain. When a block is created and appended to the chain, a first genesis transaction is inserted. The main block header has a special field that stores the hash for this first transaction. After the block is created, new transactions can be inserted into the block, and the transaction generated is appended to the last block transaction, forming a kind of auxiliary chain from the block. Each new transaction has a hash for the previous transactions, just like a traditional blockchain.

To handle block insertions, appendable-block blockchain uses different consensus algorithms, like Proof-of-Work (PoW), Practical Byzantine Fault Tolerance (PBFT), and Delegated Byzantine Fault Tolerance (dBFT) [13]. To solve security issues related to transaction insertion, Lunardi et al. [11] proposed a context-based consensus for appendable-block blockchain. Each device has its own context connected with multiple gateways, consensus will be executed inside each context. After a consensus is performed, a gateway can propagate the new consented set of transactions to the gateways from other contexts.

3.2 Patent - Lifecycle Change Cryptographic ledger

The patent [20] describes a real use case scenario that addresses the use of a model similar to the one proposed by SpeedyChain, but in the context of device

lifecycle management. With this new model, it would be possible to improve a company's current IT management by decentralizing the storage of information useful to consumers, such as telemetry data from the lifecycle events of devices such as computers and printers. This data usually includes components' performance and speed telemetry, for all devices. This way, users themselves can retrieve this information and check the "health" of their devices at any time.

In Fig. 1, it is possible to see the structure addressed in the patent, in which each block of the main chain for the devices has pointers to auxiliary chains for each of its components. This way it is possible to retrieve specific components lifecycle at once and efficiently, without needing to go through the entire chain of data. Furthermore, another usability of separating components' telemetry in different chains is the capability of keeping the lifecycle of a component even when trading it to a different device. That's a difficult use case to keep track of in the usual centralized IT management services.

Since the model described in the patent is similar to the SpeedyChain implementation, the idea is to use it to demonstrate the functionality of the new advances explored and implemented in SpeedyChain, especially regarding the storage and retrieval of information, evaluating the viability of the model proposed by the patent in this new context, as well as the behavior of the entire data storage and consistency structure. Therefore, telemetry data for the device will group the information of its components in the same block, as auxiliary chains, to determine improvements in performance when retrieving telemetry data from the same context.

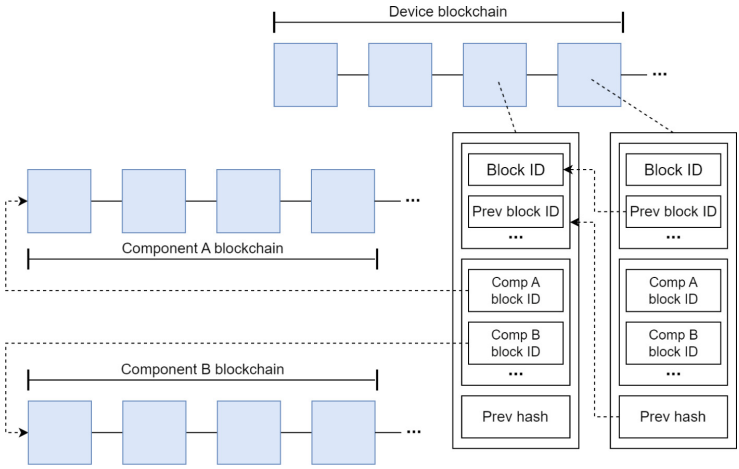


Fig. 1. Patent blockchain structure (adapted from [20]).

4 Appendable-Block Blockchain for IT Management

This work aims to explore the use of blockchain for managing device lifecycle data. To achieve this, advancements are required in the areas of data stor-

age, management, and retrieval within SpeedyChain [14]. In addition to these advancements, the feasibility and functionality of Rafael Zotto's patent [20] are intended to be demonstrated, particularly the implementations involving blockchain and its entire context. SpeedyChain implements an appendable-block blockchain model with the potential for smart contract usage [15]. Various related works developed in the same field will be employed to help the target model [1,10–14]. The development of a decentralized event recovery mechanism will also be evaluated. The goal is to enumerate the challenges of this type of structure, assessing the improvement in managing lifecycle events, from storage to recovery.

The main concept of this proposal is to implement the appendable-block blockchain example depicted in Fig. 2 within the patent's context as a use case. In this model, the blockchain features a main chain, where each block corresponds to a device in the network, and each transaction stored in that block serves as a "pointer" to another blockchain containing lifecycle events for specific components of that device. The proposed example seeks to make modifications to the SpeedyChain model, in order to accommodate the newly acquired data. SpeedyChain includes a mechanism for storing new transactions in blocks already inserted in the blockchain, which facilitates the implementation of the new model proposed in the patent. Studies on the utilization of smart contracts in this chain have been conducted, demonstrating the feasibility of such applications in SpeedyChain.

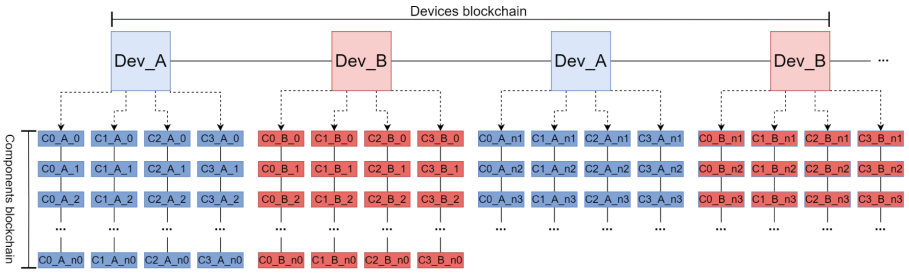


Fig. 2. Implemented Structure

Furthermore, optimizations need to be made in the current version of the appendable-block blockchain to ensure the organization of component data storage in chains of the same context, starting from the main block. As a single device has multiple components, modifications must be made to SpeedyChain to ensure that events for the same device are linked to it, utilizing digital signature information with asymmetric encryption available in SpeedyChain. Currently, this model only displays an auxiliary chain from the main one. Due to multiple components, changes must be implemented to ensure the storage of multiple chains. In the example shown in Fig. 2, each device has four components, meaning four auxiliary chains for each main block.

The work aims to address advancements that have not been fully explored in appendable-block blockchains. One of these advancements is the implemen-

tation of data preservation and consistency processes. Currently, this model is oriented towards virtual machines, with data stored in memory only during processing. Therefore, alterations will be made to ensure data persistence for future executions, storing information in structures such as IPFS (InterPlanetary File System) or locally.

To obtain lifecycle events of components, the goal is to develop an application that performs system calls on a computer, thereby obtaining the state of memories, processors, and other device components. Currently, these values are simulated through a data generation created for this purpose. As for the recovery application of lifecycle events, the primary objective is to utilize a different machine for decentralized information retrieval, as outlined in the patent. Since SpeedyChain can be run on virtual machines, this connection can be emulated using such a topology.

Due to this new structuring and changes in data storage and consistency, we present a solution for information retrieval from a predefined entry point. For the context of the addressed use case, the implementation of a mechanism using a unique identifier for a component or device will be used to retrieve all of its telemetry. Consequently, users can access their machine's history at their convenience, collecting the health of their device. Moreover, the concept of smart contracts in SpeedyChain should be reevaluated, allowing for the creation of new contracts to enhance the model's usability. An example within the patent's context involves using collected lifecycle events to identify the need for preventive maintenance in a faulty component, alerting the user to conduct a replacement before the device becomes irreparable.

4.1 Data Generation

SpeedyChain was improved with a new model and code. The original chain was constructed considering the simulation of IoT device information, and storing collected IoT data, such as temperature, GPS, and others. The first implemented modification was in the type of data generated by the simulation, adding the capability to create data simulating four basic components of a computer: processor, graphics card, RAM memory, and SSD. This information was generated with random values based on parameters of real component access speed and performance, e.g., 2000 MHz, 300 Mb/s, and 2.50 GHz, depending on the component.

Based on the parameters generated for the simulation, a structure is created to be stored in the chain with values corresponding to the lifecycle event (Fig. 3). The structure contains the following main data fields: event index, type of stored event (Type), unique identifier of the component (ID), and the value of the simulated data (Data). The generated lifecycle event is stored in its specific transaction and finally inserted into the transaction chain corresponding to the target block component of the device.

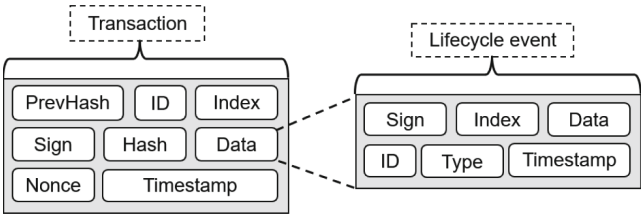


Fig. 3. Structure stored in transactions

4.2 Data Storage

The developed and evaluated blockchain provides data storage through two different methods. The first method utilizes the code previously implemented by its authors to store data from different components in a single transaction chain originating from a block in the main chain. Simulated data is generated for each component and sequentially stored, resulting in the insertion of four transaction block sets into the chain, one after the other, with each new component simulation. Modifications were implemented to store lifecycle event data, instead of the initial use case involving IoT device information, as explained in Sect. 4.1.

The second storage mechanism implemented is the version addressed in Sect. 4 and illustrated in Fig. 1 and Fig. 2. Each block in the main chain features an auxiliary chain for each of the device's components, summing up four transaction chains for the given simulation. Fundamentally, this is the sole distinction between a block in the original chain and the newly implemented version. This alteration enables intelligent context recovery, as it's unnecessary to go through all transactions within a block, it only needs to return the corresponding auxiliary chain for that component.

4.3 Data Retrieval

The implementation of data retrieval was achieved through unique identifiers for both blocks and transactions. In the conducted simulations, each peer in the network behaves as a device containing the four components discussed in Sect. 4.1. Each part of the device has a unique identifier, functioning like a unique serial number. To aid in linking the involved parts, each component's identifier also includes the identifier of its parent device, facilitating the specific recovery of context internally, as the device in question can be recognized solely by the component's identifier.

There are two main methods for information retrieval. The first method utilizes the device identifier to retrieve all blocks from the main chain corresponding to the sought-after device. The second method uses the unique identifier of a component to retrieve, across the entire chain, only the transactions corresponding to that context. It should be noted that, as explained in Sect. 4.2, the transaction retrieval function operates differently for the two types of chains tested, as their internal structures differ due to the number of auxiliary chains

for transactions. In both cases, the analysis method for evaluating data retrieval involves measuring the time required to perform search operations within the chains.

4.4 Data Consistence

Regarding the implementation of techniques for the preservation and consistency of stored data in the chain, functions were created to store and retrieve the state of the chain using text files. These functions gather the key pair from each of the peers involved in the blockchain, along with all relevant information related to the blocks and their respective transactions. The stored values are sufficient for preserving the entire state of the chain, making it possible to recreate it from these values and ensure correct order and validation. Therefore, its data is persisted for future executions and tests with greater ease and reliability in the results obtained from the simulations.

Subsequently, the idea is to enhance this work by using a more robust structure for data persistence, potentially utilizing a system like IPFS for consistent storage of Speedychain's context, thus facilitating the storage of a more complex structure with numerous blocks and transactions. However, initially, the current files generated by the simulation are meeting the requirements and facilitating the acquisition of consistent results for each execution and test.

5 Experiment Evaluation

Tests were conducted by simulating four different devices, each of them with the four components discussed earlier in this study. Figure 4 shows the structure used for data recovery tests. For each device, two blocks were created. In each block, 100 transactions were added for each component, totaling 400 transactions per device. Therefore, in the execution of the standard SpeedyChain, each block of the device has a single chain of 400 transactions, with each data from the components interleaved as shown in Fig. 5. A total of eight main blocks and 3200 transactions were created. For the simulations, a computer with an Intel i5-10400 processor, 2.90 GHz, and 6 cores was used, along with 8GB of 2666 MHz RAM and a KINGSTON A400 480 GB SSD with a reading speed of 500MB/s and a writing speed of 450 MB/s.

In Table 1, the results of the data recovery test of two components for different devices, using their unique identifiers, can be seen. The results demonstrate that the information recovery time for the model illustrated in Fig. 4 is significantly faster compared to the original SpeedyChain's implementation. It happens due to the fact that data retrieval in the new model gathers the entire transaction chain directly instead of searching the entire chain for specific component data.

Another preliminary test conducted was measuring the time for the creation of the entire chain compared to its reconstruction from the generated consistency file, discussed in Sect. 4.4. These values can be observed in Table 2, which illustrates a significant decrease in the time required to recreate all chains of

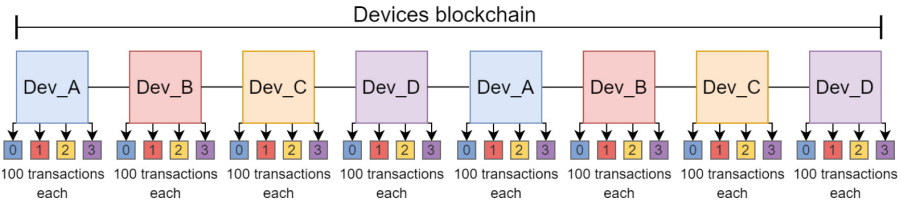


Fig. 4. Modified SpeedyChain structure for the experiment

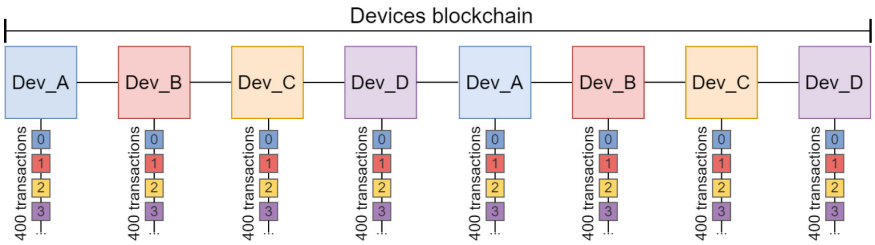


Fig. 5. Original SpeedyChain structure for the experiment

the system. Since the reconstruction of the structure uses pre-validated data from the network, its insertion into the chain is easier compared to the standard creation of authenticated blocks and transactions.

Table 1. Comparison of transaction retrieval times (ms)

	Dev_A - Component 0		Dev_C - Component 2	
ID	New impl	Original impl	New impl	Original impl.
0	0.041007995605	0.519990921021	0.026941299438	0.424861907959
1	0.044822692871	0.463008880615	0.039815902710	0.439882278442
2	0.033140182495	0.416994094849	0.030040740967	0.403165817261
3	0.038862228394	0.405073165894	0.070095062256	0.486135482788
4	0.034093856812	0.531911849976	0.031948089600	0.436067581177
Average	0.0383853912354	0.467595782271	0.0397682189942	0.4378226135254

These are preliminary tests because they consider a small chain with few devices, blocks, and transactions. It is necessary to conduct more elaborate tests, not only to evaluate the recovery performance in this case but also to assess the scalability of the implemented multiple transaction chain approach. Another test that will be addressed in the final study is evaluating the time for block and, mainly, transaction creation. Despite these measurements having already been conducted in the original work of SpeedyChain [14], the changes in the blockchain might have altered these results, especially concerning transactions within multiple chains.

Table 2. Comparison of chain creation times (ms)

ID	Creation	Reconstruction
0	70378.96	23209.98
1	69227.06	22916.13
2	68791.97	22590.59
3	70213.40	23086.82
4	69332.54	22739.11
Average	69588.79	22908.53

6 Final Considerations

The concept of appendable-blockchain presented in SpeedyChain is a recent innovation in the universe of blockchain, and there is no related work to this model in other research in the area of device lifecycle management. Thus, there are still several approaches to be researched and advances in this topology still need to be explored. This work discussed improvements in the storage and retrieval issues of stored data in this structure. The contributions that these advances in the model improved performance, decreasing latency and energy consumption, topics that SpeedyChain itself seeks to improve in relation to traditional blockchains.

As the stored lifecycle data are small, SpeedyChain is expected to be able to meet the data storage performance demand, even if multiple devices are used in the network. Many traditional blockchains use auxiliary structures to perform off-chain data storage. However, SpeedyChain removes the insertion of transactions from the insertion of blocks, thus, transactions can be added in parallel in the chain in the different blocks that correspond to each device. Furthermore, the proposed implementation improved SpeedyChain by using different chains for each component, untying the data from a single chain, allowing not only a smarter retrieval, but also the exchange of context between different machines in a simple way.

A common problem in chained structures such as blockchain is the process of searching for a specific piece of information. Therefore, some existing solutions seek to improve this issue by indexing the stored data. One of the tools widely used in blockchain for this purpose is The Graph [6], a decentralized indexing protocol. This structure guarantees an improvement in information retrieval time because, instead of consulting the entire blockchain, it is able to search for the index of the information to be retrieved in the database. As a future work, this type of protocol can be used to enable efficient data retrieval performance for the implementation by relying on this decentralized indexing structure using a trusted query language.

References

1. de Arruda, E.H.P., Lunardi, R.C., Nunes, H.C., Zorzo, A.F., Michelin, R.A.: Appendable-block blockchain evaluation over geographically-distributed IoT networks. In: 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 1–6 (2020)
2. Berdik, D., Otoum, S., Schmidt, N., Porter, D., Jararweh, Y.: A survey on blockchain for information systems management and security. Inf. Process. Manage. **58**(1), 102397 (2021)
3. Brown, C., DeHayes, D., Perkins, W., Martin, W.: Managing Information Technology. Pearson Prentice Hall, Hoboken (2012)
4. Bryan, V.: Chapter 14: Information technology management compendium. M. Moiseichik(ed.), Management of Park and Recreation Agencies, 3rd Edition **14**, 305–347 (2010)
5. Chen, S., Cai, X., Wang, X., Liu, A., Lu, Q., Xu, X., Tao, F.: Blockchain applications in PLM towards smart manufacturing. Int. J. Adv. Manuf. Technol. **118**, 1–15 (2021). https://doi.org/10.1007/.s00170-021-07802-z
6. Foundation, G.: About the graph - what is the graph?, July 2023. https://thegraph.com/docs/en/about/
7. Gërvalla, M., Preniqi, N., Kopacek, P.: It infrastructure library (ITIL) framework approach to it governance. Int. Feder. Autom. Control - PapersOnLine **51**, 181–185 (2018)
8. Hoske, M.T.: New model optimizes machine metrics. Control. Eng. **65**(11), 10–11 (2018)
9. Liu, X., Wang, W., Guo, H., Barenji, A.V., Li, Z., Huang, G.Q.: Industrial blockchain based framework for product lifecycle management in industry 4.0. Robot. Comput.-Integr. Manuf. **63**, 101897 (2020)
10. Lunardi, R.C., Michelin, R.A., Neu, C.V., Zorzo, A.F.: Distributed access control on IoT ledger-based architecture. In: IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 1–7 (2018)
11. Lunardi, R.C., Alharby, M., Nunes, H.C., Zorzo, A.F., Dong, C., Moorsel, A.V.: Context-based consensus for appendable-block blockchains. In: 2020 IEEE International Conference on Blockchain (Blockchain), pp. 401–408. Rhodes, Greece (2020)
12. Lunardi, R.C., Michelin, R.A., Neu, C.V., Nunes, H.C., Zorzo, A.F., Kanhere, S.S.: Impact of consensus on appendable-block blockchain for IoT. In: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2019, pp. 228–237. Association for Computing Machinery, New York, NY, USA (2019)
13. Lunardi, R.C., Michelin, R.A., Nunes, H.C., Neu, C.V., Zorzo, A.F., Kanhere, S.S.: Consensus algorithms on appendable-block blockchains: impact and security analysis. Mob. Netw. Appl. **27**(4), 1408–1420 (2022)
14. Michelin, R.A., et al.: Speedychain: a framework for decoupling data from blockchain for smart cities. In: 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2018, pp. 145–154. ACM, New York, NY, USA (2018)
15. Nunes, H.C., Lunardi, R.C., Zorzo, A.F., Michelin, R.A., Kanhere, S.S.: Context-based smart contracts for appendable-block blockchains. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–9. IEEE, Toronto, ON, Canada (2020)

16. Posey, B.: Device as a service (DAAS) (Abr 2022). https://www.techtarget.com/searchvirtualdesktop/definition/device-as-a-service
17. Tran, V.H., et al.: Machine-as-a-service: blockchain-based management and maintenance of industrial appliances. Eng. Rep. **5**(7), e12567 (2022)
18. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutorials **18**(3), 2084–2123 (2016)
19. Wang, C.: Techno Vision: The Executive's Survival Guide to Understanding and Managing Information Technology. Management: MacGraw-Hill, McGraw-Hill, New York, NY (1994)
20. Zotto, R.D.: Lifecycle change cryptographic ledger (Abr 2021). https://patents.google.com/patent/WO2021061146A1/en, publisher: Hewlett-Packard Development Company, L.P. (WO2021061146A1)

Exploiting the Potential Anomaly Detection in Automobile Safety Data with Multi-type Neural Network

Quanlong Guan[1,3,6], Tian Zhang[1,6], Xiujie Huang[1], Yuansheng Zhong[2], Cuifeng Du[4], Changjiang Liu[2], Zhefu Li[1(✉)], Guanghui Zhang[1,3,6], Xiaofeng Wu[5(✉)], and Zhifei Duan[7]

[1] Jinan University, Jinan 510632, Guangdong, China
{Gql,t_xiujie,lzf}@jnu.edu.cn
[2] Key Laboratory of Safety of Intelligent Robots for State Market Regulation, Guangdong Testing Institute of Product Quality Supervision, Jinan, Guangzhou, China
zhongys@gqi.org.cn
[3] Guangdong Key Laboratory of Data Security and Privacy Preserving, Jinan, Guangdong, China
[4] Cetc Potevio Science and Technology Co., Ltd., Jinan, Guangdong, China
[5] Guangzhou Polytechnic of Sports, Jinan, Guangzhou, China
[6] Guangdong-Macao Advanced Intelligent Computing Joint Laboratory, Jinan, China
[7] Guangzhou XPeng Motors Technology Co., Ltd., Jinan, Guangdong, China
duanzf@xiaopeng.com

Abstract. As an internal network widely used in automobiles, the automotive CAN bus network lacks effective security protection mechanisms and is vulnerable to network hackers, posing a serious threat to the safety of vehicles and drivers. The automotive intrusion detection system provides effective protection for the security of the automotive CAN network. To address the shortcomings of current intrusion detection algorithms, such as long application time and incomplete detection types, GIDPS and TIDPS models are proposed to perform supervised multi-classification experiments on vehicle intrusion data. Then, the above model is migrated to the ROAD dataset for verification, and the advantages of the new model in terms of time and accuracy compared with the old model are analysed based on the results. The proposed GIDPS and TIDPS models achieve better results than previous models in terms of synthesis. The new models provides a certain reference value for improving the level of automotive network security. They could be applied to domestic or cross-border automotive markets.

Keywords: CAN bus network · Intrusion detection · Network hackers · Multi-classification

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 489–501, 2024.
https://doi.org/10.1007/978-3-031-63989-0_26

1 Introduction

In-vehicle network systems play an important role in connected smart cars [1]. Not only do they need to connect and control each major electronic control unit (ECU) in the car, but they also need to communicate with various external vehicles, traffic systems and cloud platforms [2,3]. The CAN bus system has the advantages of high real-time response, long transmission distance, good interaction effect and good economy. However, the CAN bus protocol does not consider information security issues, and there is no encryption mechanism and identity discrimination mechanism, so hackers can easily obtain information in the car, forge identities to send information, and tamper with bus data, which threatens the security of the car [4,5].

Therefore, it is necessary to improve the detection level of CAN network data in the car and to find illegal intrusion information in time. Intrusion Detection System (IDS) is one of the important security measures to ensure the safe communication of CAN network. The intrusion detection technology it adopts is directly related to the effect of intrusion detection [6]. However, in previous works, the vehicle security intrusion detection model has problems such as slow computation speed and poor detection effect of certain types of attacks.

Based on the problems existing in previous models, we first pre-processed the CAN message data in the car to speed up the model's data processing time. Then, on this basis, we also investigated a new TIDPS model [7,8] based on the self-attention mechanism and a GIDPS model [9] that can be used for time series analysis. Different from the binary classification method of previous models, the model innovatively adopts a multi-classification method to further improve the computational speed of the model and optimise the processing effect of attack types. Finally, the newly proposed model is migrated to a new dataset for comparison, which verifies the effectiveness of our proposed GIDPS and TIDPS models to improve the accuracy and efficiency levels of anomaly detection.

The remaining chapters of this article are organised as follows: Sect. 2 gives an overview of the related work done in the past, analyses the existing problems, and draws out the key points to pay attention to in this article. Section 3 mainly presents two related datasets, including their data structure, type of attack, data volume and preprocessing procedure. Section 4 introduces the structure and computational principles of the three correlation network models. Section 6 is a summary of this thesis, which summarises the work done and the results achieved in this thesis, and introduces the future research direction.

2 Related Work

At present, the more commonly used IDS intrusion detection methods include intrusion detection methods based on statistics and intrusion detection methods based on machine learning [9,10]. During the detection process, if the statistic is found to exceed the threshold, it is judged to be abnormal.

Müter et al. [10] first proposed an entropy-based intrusion detection method that works well for detecting flooding and replay attacks, but has a high false

positive rate for tampering and spoofing attacks, indicating that it can be detected by information theory. Since then, there have been many studies to improve and integrate new variables based on information entropy. For example, in 2021, Zhang et al. [11] proposed a CAN network intrusion detection mechanism based on relative entropy, which replaced the time window in the anomaly detection mechanism based on information entropy with the data window, so it can be detected by analysing non-exception for periodic data frames. However, statistics-based intrusion detection methods may have problems such as modelling difficulties and threshold determination difficulties [4].

The intrusion detection method based on machine learning uses supervised or unsupervised machine learning algorithms to train normal and abnormal data to learn their respective characteristics and identify outliers for intrusion detection. Seo [12] proposed the GIDS (GAN-based vehicle network IDS) model, GAN (Generative Adversarial Network, Generative Adversarial Network), which uses the generation network and the identification network to fight against each other to achieve the goal of co-evolution. Song et al. [13] proposed a DCNN-based IDS to build a simplified ResNet to reduce the complexity of the network. Hu et al. [14] converted the data into two-dimensional images using their own coding methods, and used different convolutional neural networks to extract and train the data features. The deep neural network has the advantages of strong self-learning and low delay. According to tests, it has a high detection accuracy, but it relies on a large amount of data for learning, requires more computational resources and lacks an explanation mechanism.

It can be seen that the previous machine learning detection models have problems such as slow computation speed and poor detection effect for certain types of attacks. Therefore, the work in this paper mainly uses the neural network model based on the self-attention mechanism to pre-process the vehicle intrusion dataset, and then combines appropriate data encoding and multi-classification methods to propose several new models to solve the above problems.

3 Data Analysis and Preprocessing

The dataset used in this experiment is the real vehicle network HCRL data set [13,15] collected and published by Korea Hacker and Countermeasures Research Laboratory (HCRL) and the real vehicle ROAD data set [16] recorded by Oak Ridge National Laboratory (ORNL). The following will briefly introduce the two data sets and data preprocessing.

3.1 HCRL Dataset

The HCRL dataset [13,15] was collected in a real in-vehicle network environment, including normal network communication behaviour and many different types of attacks. Details are given in Table 1.

Table 1. HCRL dataset attack description

Attack name	Attack method	Attack interval
DoS attack	Inject a message with CAN ID "0000" once	0.3 ms
Fuzzing attack	Inject a completely random CAN ID and DATA message	0.5 ms
Spoofing attack	Inject messages for specific CAN IDs related to RPM or gear information	1 ms

Then by intercepting the CAN network message sequence dataset with a window of step size 10, marking the frame containing the 'T' mark as the corresponding attack frame, extracting the ID of each frame and drawing the corresponding ID curve, see Fig. 1 below. The ordinate in the figure is the number of IDs, and the abscissa is the step size.

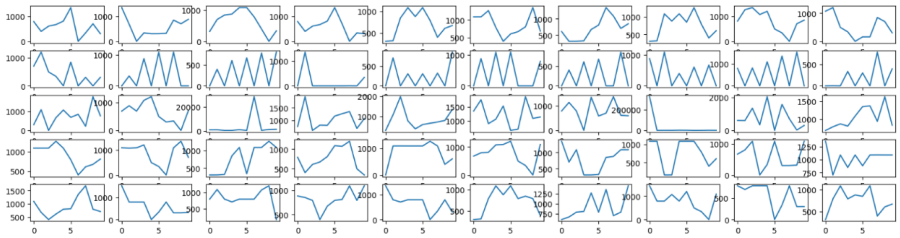


Fig. 1. The partial ID curves of the HCRL, from top to bottom: normal data, Dos attack, Fuzzy attack, gear attack, RPM attack

Observing Fig. 1, it can be seen that the ID curve characteristics of DoS attacks are relatively obvious, showing a clear jagged shape, and can be easily classified according to timing and ID. The remaining three features are not obvious and need to be combined with other data extracted from the network for better classification.

Preprocessing the HCRL Dataset. First, fill in the NAN value of the data. Because there is no data in some bits of some messages, there are NAN values, so the NAN bits are filled as 0. Then, we performed data type conversion and a number system conversion. In this paper, the data is converted directly to decimal and each byte of the ID and DATA fields is used as a feature, making a total of 10 features. Next, we calculated the frame interval. Messages have time stamps and there is a time difference between messages which is not uniform. Therefore, the difference between two adjacent messages is calculated as a feature. We then normalized the data to eliminate the effects of anomalies. In particular, the frame interval, because the gap is almost milliseconds, the order of magnitude is small and needs to be increased. Finally, we label the data. In this dataset, the time slice containing the 'T' tag is marked as the sequence number of the corresponding attack.

Table 2. HCRL data step division result

Step size	Normal	Dos	Fuzzy	Gear	RPM
1	3404896(200000)	233004	196680	180909	178914
5	466153(100000)	84966	89427	96038	102299
10	213629(50000)	42632	51247	55838	56096

From Table 2 we can see that the number of attacks decreases as the step size increases, indicating that the attacks are not sparse.

3.2 ROAD Dataset

The ROAD dataset [16] provides the following realistic attacks: one obfuscation attack, many maximally stealthy target fabrication attacks, and two that do not contain faked advanced attacks on messages.

From Table 3, the message with ID 6E0 on the left has two lines, but the underline is the attack message modified by the attacker, and the original message is removed from the dataset to disguise it. With such a spoofed dataset, frequency-based methods will almost certainly fail to provide accurate detection (Table 4).

Table 3. ROAD dataset simulates masquerading attack

Message conflict	Remove conflict
9.196895) FFF#0000000000000000	9.196895) FFF#0000000000000000
9.196896) 6E0#03AF03A603A403A6	9.197928) 6E0#595945450000FFFF
9.197928) 6E0#595945450000FFFF	9.199034) 354#2016800000012080

Table 4. HCRL data step division result

Step size	Normal	Correlated	Max_engine	Speedometer	Light_off	Light_on
1	10000	5488	43	11689	5476	8032
5	10000	5488	43	11689	5476	8032
10	10000	5400	43	11680	5476	8032

Preprocessing the ROAD Dataset. After our data pre-processing operation, we can know that the number of attacks in this data set is small and sparse.

4 Building the Intrusion Detection Model

The experimental data set and data pre-processing operations have been described in the third section, and in this section we mainly establish the vehicle intrusion detection model. Therefore, according to the time characteristics of the

above data, we established the time series models GIDPS and TIDPS based on GRU [17] and Transforme [18], and compared with the previous LSTM-based model (LIDPS) [19–21]. Therefore, different types of attack data and normal data can be effectively distinguished, and the intrusion attack of the automotive CAN network can be detected.

4.1 LIDPS Model

Previous [19–21] mostly used LSTM neural networks for binary classification tasks, but this paper proposes an LSTM-based LIDPS classification model for multi-classification tasks of vehicle CAN messages, which improves the classification accuracy.

$$f_t = \delta * (W_f * [h_{t-1}, x_t] + b_f) \tag{1}$$

Equation (1) combines the output of the previous layer and the input of this layer to multiply the weight, plus the bias, and passes the result through the sigmoid activation function. In this, δ is the sigmoid activation function, h_{t-1} is the hidden information of the previous layer, x_t is the input of this layer, W_f is the weight factor and b_f is the bias function.

$$\begin{cases} i_t = \delta * (W_i * [h_{t-1}, x_t] + b_i) \\ C_t^{\sim} = tanh * (W_f * [h_{t-1}, x_t] + b_f) \\ C_t = f_t * C_{t-1} + i_t * C_t^{\sim} \end{cases} \tag{2}$$

As shown in Eq. (2), after passing through the tanh activation function, the state value is compressed between –1 and 1 and then multiplied by the output through the sigmoid to obtain the output. Among them, the tanh function is used to help adjust the value flowing through the network and play a role in sorting out. $C_t^{\sim}$ is the memory information of the previous layer.

$$\begin{cases} o_t = \delta * (W_o * [h_{t-1}, x_t] + b_o) \\ h_t = o_t * tanh(C_t) \end{cases} \tag{3}$$

In the output gate, the hidden information of the upper layer in Eq. (3) and the input of this layer first pass through the sigmoid function to determine the output content, and then multiply the memory information passed through the tanh function to obtain the hidden part of the next layer.

4.2 GIDPS Model

Gated recurrent unit (GRU) [22] is a recurrent neural network structure. Compared with LSTM [19], GRU [22] only has two gating units, the reset gate and the update gate. Therefore, based on the previous intrusion detection model based on GRU [22], in order to better compare and highlight the TIDPS model, this paper proposes a GIDPS model that can perform multiple classification tasks.

$$\begin{cases} r_t = \delta * (W_r * [h_{t-1}, x_t] + b_r) \\ z_t = \delta * (W_z * [h_{t-1}, x_t] + b_z) \end{cases} \tag{4}$$

In Eq. (4), the input at time t and the state of the hidden layer at time t-1 contain information about previous nodes. After passing through the sigmoid function, the two inputs become the control signal for remembering or forgetting.

$$h_t^{\sim} = tanh * (W_h * [r_t * h_{t-1}, x_t] + b_h) \tag{5}$$

The calculation method of the reset gate is shown in Eq. (5). When r_t tends to zero, the previous hidden information is discarded, leaving only the input, and when r_t tends to 1, the previous information is added to the current information.

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t^{\sim} \tag{6}$$

The calculation method of the update gate is shown in the above Eq. (6). By selectively forgetting the hidden state of the upper layer and then selectively remembering the hidden state of the current node.

4.3 TIDPS Model

The TIDPS model proposed in this paper is based on Transformer [18], using a self-attention mechanism (self-attention) to dynamically generate weights for different connections between the input and output of the same layer network.

$$\begin{cases} Q(W_Q * X), K(W_K * X), V(W_V * X) \\ Attention(Q, K, V) = softmax(\frac{Q * K^T}{\sqrt{d_k}}) * V \end{cases} \tag{7}$$

Among them, Q, K and V are query vector sequence, key vector sequence and value vector sequence, respectively. The input X is projected into three different spaces by the linear transformation of Eq. (7), and Q, K and V can be obtained. W_Q, W_K and W_V are three learnable parameter matrices.

$$MultiHead(Q, K, V) = Concat(Attention(Q_i, K, V)...Attention(Q_k, K, V)) \tag{8}$$

The multi-head attention mechanism is to repeat the same operation multiple times, and the results are finally concatenated. Different Q pay attention to different features, evaluate the input information from different angles and then integrate them as shown in Eq. (8).

As shown in Fig. 2, the complete Transformer consists of two parts, the encoder and the decoder, but in the classification task, only the encoder part is needed. The position encoding is to add a position information to the input sequence to enhance the timing characteristics of the sequence.

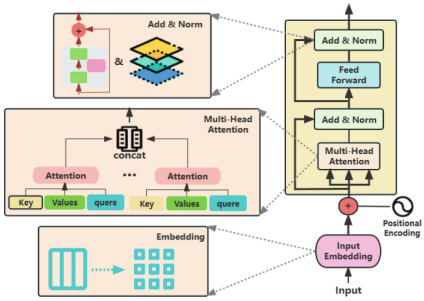


Fig. 2. TIDPS model encoder

5 Experimental Results and Analysis

The CPU used in this experiment is AMD Ryzen 5 5600H with Radeon Graphis, no GPU is used, the operating system is Windows 10, the IDE is jupyter, the Python version is 3.9, and the pytorch deep learning framework is used.

5.1 Experiments on the HRCL Dataset

As the dataset is large, the model converges quickly. The specific parameters are given in Table 5 and the ratio of training set to test set is 3:1.

To find a more suitable model and improve the training accuracy of the model, we added some modules for ablation experiments: the 1 at the end indicates the original model, and the data has gone through three rounds of the modules introduced above. After the model is finished, we expand the data into one dimension, perform a full connection operation, and finally map to 5 categories. 2 is to expand the data, then add a fully connected layer with a length of 256 and then map to 5 categories, and 3 is to add a one-dimensional convolutional layer and then go through the fully connected layer.

Table 5. Training parameters based on HRCL dataset

Parameter	Batch Size	Learning Rate	Optimizer	Classes	Loss
Value	64	0.0005	Adam	5	Cross Entropy

DES of the Table 6 shows the training results using the early stopping strategy and using decimal data, where sql_len represents the sequence step size. It can be seen from the data in the table that the newly added modules can work when the hidden size is small, but as the hidden size increases, although some will speed up the convergence of the model, the accuracy is lower than when no new modules are added.

Table 6. The results of training the model on the HCRL dataset

Cond[1]	Sqe[2]	Hs[3]	$LIDPS_1$	$LIDPS_2$	$LIDPS_3$	$GIDPS_1$	$GIDPS_2$	$GIDPS_3$	$TIDPS_1$	$TIDPS_2$	$TIDPS_3$
DES[4]	**5**	**32**	0.9815	0.9933	0.9898	0.9859	0.9853	0.9655	0.9924	0.9922	0.9879
			00:55	00:51	01:00	00:45	00:50	00:36	01:53	01:33	01:24
		64	0.9949	0.9962	0.9891	0.9935	0.9903	0.9900	0.9939	0.9918	0.9723
			01:56	01:56	00:54	01:27	00:41	00:43	01:25	01:32	00:39
		128	0.9963	0.996	0.994	0.9966	0.9948	0.9957	0.9972	0.9946	0.9933
			02:25	02:15	01:46	02:49	01:17	02:06	02:09	01:44	01:27
	10	**32**	0.9913	0.9933	0.9897	0.9802	0.9853	0.9577	0.9924	0.9922	0.9879
			01:34	00:51	00:46	00:34	00:50	00:32	01:53	01:33	01:24
		64	0.9963	0.9962	0.994	0.9936	0.9903	0.9933	0.9939	0.9918	0.9723
			02:15	01:56	01:19	01:32	00:41	01:26	01:25	01:32	00:39
		128	0.9969	0.996	0.9961	0.9958	0.9948	0.9958	0.9972	0.9946	0.9933
			03:26	02:15	02:35	02:17	01:17	02:40	02:09	01:44	01:27
PDM[5]	**10**	**128**	pooling epoch								
			0.9742	\	\	0.9793	\	\	0.9193	\	\
			06:02	\	\	04:21	\	\	02:13	\	\
			non-pooling epoch								
			0.9973	\	\	0.9967	\	\	0.9962	\	\
			05:37	\	\	03:57	\	\	02:24	\	\
TDM[6]	**5**	**32**	0.9931	0.9952	0.9907	0.9913	0.9887	0.992	0.994	0.9946	0.9943
			02:31	03:11	02:46	01:36	02:04	01:55	01:58	02:07	02:04
		64	0.9970	0.9970	0.9969	0.9944	0.9942	0.9953	0.9952	0.9956	0.9968
			04:06	05:04	04:34	02:45	03:26	03:09	02:27	02:14	02:10
		128	0.9971	0.9970	0.99768	0.997	0.9964	0.9967	0.9972	0.9965	0.9961
			08:32	10:29	09:02	05:45	06:34	06:11	02:43	02:29	02:26

[1] Condition
[2] Sql_len
[3] Hidden size
[4] Training results on the HCRL dataset during early stopping
[5] Increase the training results of pooling different models on the HCRL dataset
[6] Training results of different models on the HCRL dataset

It can be seen that the accuracy and time of our proposed TIDPS are quite stable. Among many results, the convergence speed of GIDPS is the fastest, but the accuracy is just the opposite. This is the price of its model being simpler than the previous LIDPS model.

Maximum pooling can reduce the dimensionality of the data and reduce the amount of computation. There are such practices in previous work, but from the pooling experimental data PDM in the Table 6, it can be seen that not only the training time increases, but also the accuracy drops a lot, so the pooling layer does not play a positive effect.

The TDM of Table 6 shows that with almost equal accuracy, the accuracy of LIDPS and GIDPS increases as the hidden size increases, but the time also increases as the hidden size doubles, while the time for TIDPS was consistently around 2 min. The ratio of time required for LIDPS, GIDPS, and TIDPS to achieve similar accuracy is approximately 18:12:5. The time for training data

may not be long, but the algorithm is used for real-time detection, i.e. the data generated all the time is fed into the network to perform calculations. Although the computation time is very short, the long accumulation time will cause a large time delay, so that even if an anomaly is detected, it cannot be processed in time. Therefore, the use of TIDPS or GIDPS has a great effect in reducing the algorithm time. It can be seen that our proposed GIDPS and TIDPS models have significantly improved in terms of accuracy, precision, and computation time compared to the previous LSTM-based LIDPS model.

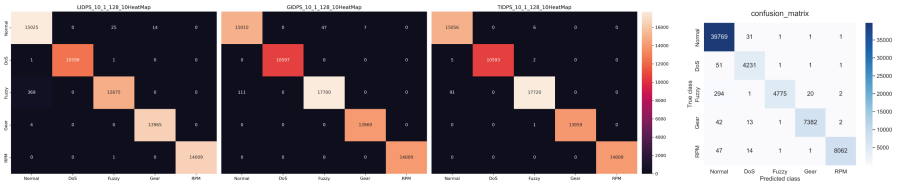


Fig. 3. From left to right are the heat comparison charts of LIDPS, GIDPS, and TIDPS in the HCRL dataset, and the far right is the CNN result map

Figure 3 is the heat map generated by several classification results with better results in this experiment and the resulting map of CNN. When the number of experimental test sets corresponding to the three models we proposed is much larger than that of CNN, each model adopted is more accurate than CNN. Although there is still the problem of detection errors in the detection of fuzzy attacks, the detection success rate of gear and RPM spoofing attacks is much better than the CNN method.

5.2 Experiments on the ROAD Dataset

This data set is different from the HCRL data set [13,15]. Since there is less attack data and training is faster, 20 epochs are used. See Table 7 for details. Since the amount of data is small, the number of test sets is slightly increased to avoid randomness, and the ratio of training set to test set is 7:3.

Table 7. Training parameters based on ROAD dataset

Parameter	BatchSize	LearningRate	Optimizer	Classes
Value	64	0.0005	Adam	6

Table 8 below shows the model with the best GIDPS and LIDPS results in the experiments and the precision, recall, and F1-score of the 3 layers in the TIDPS model.

Table 8. HCRL data step division result

Model	Attack name	Precision	Recall	F1-score
GIDPS	Normal	0.9953	0.9970	0.9961
	Correlated signal	1.0000	1.0000	1.0000
	Max engine coolant temp	0.6000	0.1875	0.2857
	Max speedometer	1.0000	1.0000	1.0000
	Reverse light off	0.9976	1.0000	0.9988
	Reverse light on	0.9988	0.9996	0.9992
LIDPS	Normal	0.9839	0.9919	0.9879
	Correlated signal	0.9917	0.9805	0.9861
	Max engine coolant temp	0.0000	0.0000	0.0000
	Max speedometer	1.0000	1.0000	1.0000
	Reverse light off	0.9976	0.9988	0.9982
	Reverse light on	0.9960	0.9992	0.9976
TIDPS	Normal	0.9927	0.9960	0.9943
	Correlated signal	0.9993	1.0000	0.9996
	Max engine coolant temp	0.0000	0.0000	0.0000
	Max speedometer	1.0000	0.9997	0.9998
	Reverse light off	0.9971	0.9971	0.9971
	Reverse light on	0.9975	1.0000	0.9987

As can be seen from Fig. 4 above, due to the small number of samples for the maximum engine coolant temperature attack, the LIDPS and TIDPS models cannot learn their features, so the recognition results of the two models are 0 for this sample. Therefore, its discriminative power is poor and the whole 6-category task degenerates into a 5-category task. Unlike the other two models, GIDPS is able to detect maximum engine coolant temperature attacks and therefore has a higher accuracy than the other models. Overall, GIDPS performed best, with the highest accuracy and the shortest time. LIDPS performed slightly worse in the related category of signaling attacks. In other cases, the three models performed similarly.

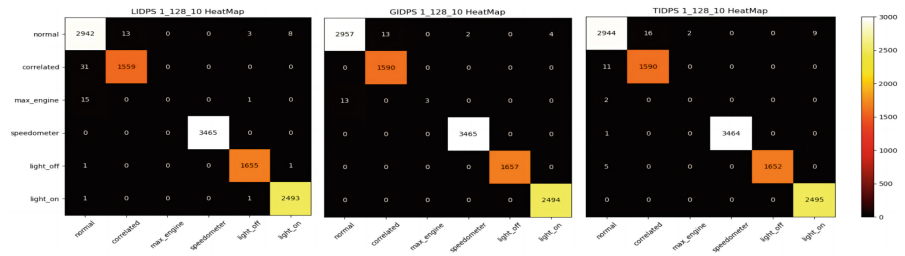


Fig. 4. ROAD dataset classification heat map

6 Conclusion

GIDPS and TIDPS models are applied for intrusion detection algorithms. We have compared and trained the proposed GIDPS and TIDPS models with previous LSTM-based LIDPS models. The time required by the proposed GIDPS and TIDPS models is much less than the previous LSTM-based LIDPS model, and the accuracy is slightly higher than that of LIDPS. It can be seen that the proposed model is efficient. Furthermore, we also migrated the model to a different dataset and still got this result. We can see that All models still are not perfect in the detection of fuzzy attacks, but under the premise of multi-classification, the effect could be slightly improved compared to binary classification. In the future, we will continue to practice and explore further improvements in algorithm performance, generalization, and resource consumption comparison.

Acknowledgements. This work was supported by the Science and Technology Planning Project of Guangdong (2020B0909030005, 2020ZDZX3013, 2023ZZ03), the Science and Technology Planning Project of Guangzhou (202206030007), the Opening Project of Key Laboratory of Safety of Intelligent Robots for State Market Regulation (GQI-KFKT202205), Guangdong Key Laboratory of Data Security and Privacy Preserving (2023B1212060036), Guangdong-Macao Advanced Intelligent Computing Joint Laboratory (2020B1212030003).

References

1. Koscher, K.: Experimental security analysis of a modern automobile. In: IEEE Symposium on Security and Privacy, vol. **2010**, pp. 447–462. IEEE (2010)
2. Kong, Q., et al.: Blockchain-based privacy-preserving driver monitoring for MaaS in the vehicular IoT. IEEE Trans. Veh. Technol. **70**(4), 3788–3799 (2021)
3. Zhang, T., et al.: VSRQ: quantitative assessment method for safety risk of vehicle intelligent connected system (2023). arXiv: 2305.01898 [cs.AI]
4. Tomlinson, A., Bryans, J., Shaikh, S.A.: Towards viable intrusion detection methods for the automotive controller area network. In: 2nd ACM Computer Science in Cars Symposium, pp. 1–9 (2018)
5. Cheng, X., et al.: Influence-aware successive point-of-interest recommendation. In: World Wide Web, vol. 26, no. 2, pp. 615–629 (2023)
6. Qaddoura, R., et al.: A multi-layer classification approach for intrusion detection in IoT networks based on deep learning. Sensors **21**(9), 2987 (2021)
7. Xie, G., et al.: Threat analysis for automotive CAN networks: a GAN model-based intrusion detection technique. IEEE Trans. Intell. Transp. Syst. **22**(7), 4467–4477 (2021)
8. Guo, J., et al.: TROVE: a context-awareness trust model for VANETs using reinforcement learning. IEEE Internet Things J. **7**(7), 6647–6662 (2020)
9. Javed, A.R., et al.: CANintelliIDS: detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU. IEEE Trans. Netw. Sci. Eng. **8**(2), 1456–1466 (2021)
10. Müter, M., Groll, A., Freiling, F.C.: A structured approach to anomaly detection for in-vehicle networks. In: 2010 Sixth International Conference on Information Assurance and Security, pp. 92–98. IEEE (2010)

11. Haichun, Z., et al.: Research on entropy-based vehicle CAN bus anomaly detection. Autom. Eng. **43**(10), 1543–1548 (2021)
12. Seo, E., Song, H.M., Kim, H.K.: GIDS: GAN based intrusion detection system for in-vehicle network. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–6. IEEE (2018)
13. Song, H.M., Woo, J., Kim, H.K.: In-vehicle network intrusion detection using deep convolutional neural network. Veh. Commun. **21**, 100198 (2020)
14. Hu, R., et al.: Multi-attack and multi-classification intrusion detection for vehicle-mounted networks based on mosaic-coded convolutional neural network. Sci. Rep. **12**(1), 1–16 (2022)
15. Seo, E., Song, H.M., Kim, H. K.: GIDS: GAN based intrusion detection system for in-vehicle network. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–6, August 2018. https://doi.org/10.1109/PST.2018.8514157.
16. Verma, M.E., et al.: Road: the real ornl automotive dynamometer controller area network intrusion detection dataset (with a comprehensive can ids dataset survey & guide). arXiv preprint arXiv:2012.14600 (2020)
17. Chung, J., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
18. Cortes, C., et al.: Advances in neural information processing systems 28. In: Proceedings of the 29th Annual Conference on Neural Information Processing Systems (2015)
19. Taylor, A., Leblanc, S., Japkowicz, N.: Anomaly detection in automobile control network data with long short-term memory networks. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 130–139. IEEE (2016)
20. Hossain, M.D., et al.: LSTM-based intrusion detection system for invehicle can bus communications. IEEE Access **8**, 185489–185502 (2020)
21. Hanselmann, M., et al.: CANet: an unsupervised intrusion detection system for high dimensional CAN bus data. IEEE Access **8**, 58194–58205 (2020)
22. Chung, J., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling. English (US). In: NIPS 2014 Workshop on Deep Learning, December 2014

Urban/Mobile Crowdsensing

HAUM^3: A Height Aware Urban Map Matching Mechanism

Jie Tang[1], Sunjian Zheng[1(✉)], Bo Yu[2], and Shaoshan Liu[2]

[1] South China University of Technology, Guangzhou, China
cstangjie@scut.edu.cn, sunjian_zheng@163.com
[2] PerceptIn, Fremont, USA
{bo.yu,shaoshan.liu}@perceptin.io

Abstract. Map matching is an essential component of urban intelligent transportation, providing fundamental data for technologies such as path planning, traffic analysis, and trajectory analysis. However, our commercial deployments showed that existing map matching algorithms behave ineffectively in complex urban scenarios such as elevated and interchange roads due to their insensitivity to the road height change. To improve the map matching for urban map services, we proposed and deployed **HAUM^3**, a novel Height Aware Urban Map Matching Mechanism with Spatio-Temporal Correlation. Firstly, starting from the dynamic driving process of vehicles, an urban driving scene classification model is designed based on the images and sensor data generated synchronously by the vehicles. This model correlates the spatio-temporal state changes of vehicles on roads of different heights. After that, according to the characteristics of different types of map matching algorithms, the classification model is fused into traditional map matching algorithms, enabling them to aware roads at different heights and improving their performance. Evaluation results show that this mechanism significantly improves the accuracy of different types of map matching algorithms in complex urban roads, with an average accuracy increase of over 10%, enhancing the usability of traditional map matching algorithms.

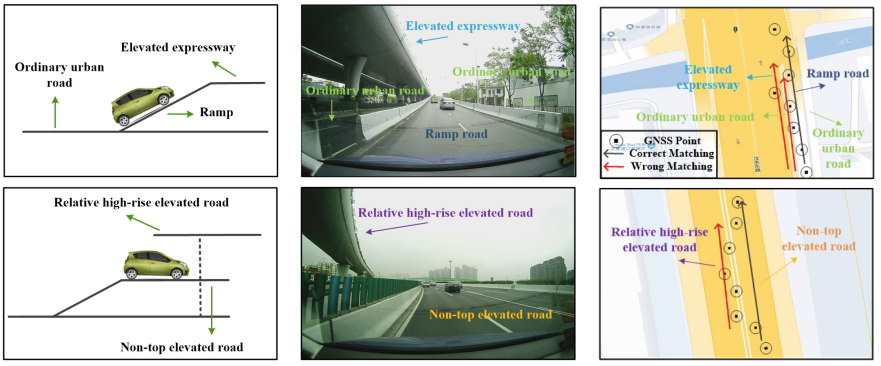
Keywords: Map matching · Driving scene classification · Images and sensor data · Height awareness

1 Introduction

Map matching is a technique that associates a series of GNSS (Global Navigation Satellite System) points with a road network by using maps to constrain GNSS data. GNSS sensors have been widely deployed in cars, smartphones, and other mobile devices to collect movement information. These devices accumulate vast amounts of trajectory data, which can be pre-processed using map matching technology to assign semantic information about the road network to raw data. This can then be applied in various fields such as managing urban traffic congestion, identifying functional areas in cities, and urban route planning [1].

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 505–525, 2024.
https://doi.org/10.1007/978-3-031-63989-0_27

However, with the rapid increase in automobiles, the traffic infrastructure has been fully developed and then urban road networks get complicated dramatically. Cities worldwide have constructed many elevated and interchange roads to alleviate the enormous traffic pressure on the roads. For example, Shanghai has more than 622 km of elevated roads, including multi-level elevated roads. As a result, in a two-dimensional map application, elevated roads may overlap with ground-level roads. The accuracy of position and movement direction provided by GNSS is even lower under the influence of elevated roads, with a maximum error of over 15 m [2]. Therefore, it is difficult to determine a vehicle's trajectory when traveling on elevated and nearby roads. That leaves a big problem for urban map matching both in accuracy and robustness.



(a) Side cut of the vehicle position (b) Images of the road from the driving viewpoint (c) Visualization of matching results

Fig. 1. Map mis-matching cases due to height difference. Subplot (c) shows the matching results of vehicle trajectories in complex urban roads obtained by applying the map matching algorithm. The overlap of roads at different heights and the noisy GNSS sampling points will lead to different degrees of mis-matching: the upper part of subplot (c) indicates that the matching algorithm mis-matches a normal urban road and a elevated expressway; the lower part of subplot (c) indicates that the matching algorithm mis-matches a relative high-rise elevated road.

When we dive into deployment details, we can find that in complex urban road scenarios, erroneous map mismatches of traditional map matching methods based on GNSS sampling points and road network structures can be blamed on their blindness to the road height differences. According to our experiments from commercial deployment, mismatches often occur on elevated and nearby roads, and Fig. 1 demonstrates the typical map mis-matching cases. When the vehicle is driving on the ramp road, the GNSS trajectory may be shifted to the nearby ordinary urban road as well as the elevated road, which leads to the map matching algorithm mis-matching the ordinary urban road and the elevated road, as shown in the upper part of Fig. 1; when the vehicle is driving on the elevated expressway, the GNSS trajectory may be shifted to the nearby elevated road on the upper or lower level, as shown in the lower part of Fig. 1, which demonstrates the map matching algorithm mis-matching the elevated road on

the upper level. Even though, currently, there have some methods for detecting elevated roads, which are based on altitude [3] or features [4,5]. However, they are challenged in the accuracy or robustness, therefore, they cannot effectively work for urban roads with heights difference.

The erroneous map matching problems in complex urban roads generate great need for height information in the GNSS guided map matching. With that observation, based on a large amount of deployment data, we have developed and deployed **HAUM^3**, a novel Height Aware Urban Map Matching Mechanism with Spatio-Temporal Correlation. By combining the dynamic running process of vehicles with spatio-temporal images and sensor data generated by vehicles, it can obtain the information of road types at different heights and incorporate them into existing map matching methods. It therefore enables map matching algorithms to distinguish roads of different heights effectively and can improve the matching performance of existing map matching methods on complex urban roads. The main contributions of our work can be concluded as follows:

1. A map matching mechanism named **HAUM^3** has been proposed. This mechanism utilizes the images and sensor data of vehicles, combined with the spatio-temporal state of the vehicle, to overcome the issue of low sensitivity in map matching algorithms for complex urban roads. It effectively enhances the matching effectiveness of elevated areas, such as expressways, ramps, and auxiliary roads.
2. A spatio-temporal correlated urban road scene classification model has been devised by integrating real-time in-vehicle images and sensor data. This model amalgamates the temporal and spatial attributes of the dynamic vehicle operation process and achieves effective discrimination among five distinct levels of urban roads: high-rise elevated roads, non-top elevated roads, ramps, ordinary urban roads, and tunnels, through the simultaneous utilization of vehicle-captured images and sensor data.
3. A map matching algorithm with superior performance can be obtained. By deploying **HAUM^3** on different types of map matching algorithms, they all can obtain some performance improvement. For global map matching algorithms, stronger robustness can be obtained, while for local/incremental map matching algorithms, faster convergence can be achieved.

2 Related Work

2.1 Map Matching

Scholars have proposed various map matching algorithms and classification systems for map matching algorithms to address different characteristics of GNSS data and map data. Based on the range of sample points considered when matching trajectories, map matching algorithms can be divided into local/incremental and global methods [6].

Local/incremental algorithms use the local features of the current trajectory point to match the road network, using a greedy strategy to gradually move

towards the global solution from the existing matching solution. For example, Brakatsoulas et al. [7] proposed a map matching algorithm based on the similarity and turning angle between trajectory segments and candidate paths. Fu et al. [8] improved the map matching algorithm using the extended Viterbi algorithm and the adaptive sliding window mechanism, considering drivers' travel preferences, road class, network topology, and other factors. Local/incremental algorithms are fast in calculation speed and strong in real-time performance, so they are often used in applications with online demands.

Global algorithms consider the entire sampling trajectory and find the closest matching path from the road network. A classic method is the map matching algorithm based on the Hidden Markov Model (HMM) [9], transforming the map matching problem into a decoding problem. Gui et al. [10] proposed a trajectory segmentation Hidden Markov Model map matching method based on heading homogeneity. Compared with local/incremental algorithms, global algorithms are more accurate, show greater robustness to reduced sampling rates, and can better handle map matching problems with long sampling intervals. Nevertheless, they have higher computational costs and are only applicable for offline matching [11].

2.2 Detection of Elevated Roads

As cities continue to develop and electronic map technology advances, urban electronic maps are becoming increasingly complex, especially on trunk roads and elevated expressways where roads at different heights overlap and interweave on a 2D map. The number of candidate road segments that can be found from a single sampling point increases rapidly. Therefore, the problem of elevated road matching has attracted wide attention from scholars. Ho et al. [3] used an altitude-based method to identify the position and status of vehicles, but using the elevation directly for position detection is not feasible due to the very low accuracy of barometric pressure sensors. In contrast, Gong et al. [4] proposed a feature-based method better to capture the differences between elevated and ground roads. They extract features, such as variance in vehicle speed and the number of tracked satellites, and input them into a classification model to identify whether a vehicle is on an elevated road. Based on this, Zhang et al. [5] proposed an elevated road network, which learns two feature descriptors and uses four features (satellite plane projection feature, group ID, distance of elevated road, and sequence speed) to distinguish between surface roads and elevated roads. Furthermore, Fu et al. [12] use image data from driving recorders to classify driving scenes and assist map matching in complex urban road environments.

None of the map matching algorithms based on traditional methods take into account the adverse effects caused by urban roads of different heights. Methods specifically designed for detecting elevated roads have drawbacks such as low accuracy and poor robustness and cannot effectively distinguish between the different heights of urban roads. The mechanism proposed in this paper combines images and sensor information from the dynamic driving process of a

vehicle with spatio-temporal state, endowing the map matching algorithm with height awareness, thus improving the accuracy of the map matching algorithm in complex urban road environments.

3 Problem Definition and Mechanism Overview

In order to address the issue of mis-matching due to height differences of map matching algorithms in complex urban roads, this paper proposes utilizing coherent image and sensor information to extract the vehicle's height variation during its journey. The data are derived from vehicle devices with travel records or driverless cars. During the driving process, the vehicle is continuously positioned according to a particular sampling frequency while corresponding video frames and sensor data are recorded. By combining the trajectory variations of the vehicle on different height roads and the images and sensor information generated during the process, the perception and differentiation of roads at different heights are achieved, ultimately obtaining a height aware map matching algorithm. The relevant definitions are as follows:

Definition 1: Urban Road Network. $G = (V, E)$ represents a directed graph of the urban road network. The vertices V denotes intersections or endpoints in the road network, while the edges E correspond to directed road segments. Each segment has its corresponding road attribute information.

Definition 2: Sampling Points. $t = (timestamp, lon, lat, image, sensor\ data)$ represents a set of data collected by a single device using its positioning module to obtain the current location information while synchronously recording the video and sensor data of the current moment. The *timestamap* denotes the time at which the sampling point is taken, and (lon, lat) denotes the geographic location of the traveling device at that moment. The *image* and *sensor data* refer to the images and other vehicle-related sensor data synchronized with the sampling point collection.

Definition 3: Vehicle trajectory. $T = \{t_1, t_2, t_3, \ldots, t_n\}$ represents a set of samples obtained by the exact vehicle at a specific time interval in a period, where n denotes the number of sampling points.

Definition 4: Actual Path. $P = [e_1, e_2, \ldots, e_n]$ represents a sequence of actual paths that a vehicle traverses during its motion, where $\forall e_n \in E$ and n denotes the number of actual paths.

Definition 5: Distinction of Different Height Roads. $R = [r_1, r_2, r_3, \ldots, r_n] = S[t_1.(i,s), t_2.(i,s), \ldots, t_n.(i,s)]$ represents the road type results obtained from processing images and sensor data collected at different times and heights. Here, r_k denotes the road type result calculated at the kth sampling point, S is the processing mechanism, and $t_k.(i,s)$ denotes the image and sensor data obtained at the kth sampling point, where $k \epsilon [1, n]$.

Definition 6: Matching Path. $P' = M(G, T, R) = [e_1', e_2', \ldots, e_m']$ represents the matching path obtained by mapping vehicle trajectory T onto the urban road network G through the height aware map matching algorithm M, combining the results of distinction of different height roads R, where $\forall e_m' \in E$ and m denotes the number of matched road sections. The height aware map matching algorithm M aims to obtain a matching path P' closer to the actual trajectory P.

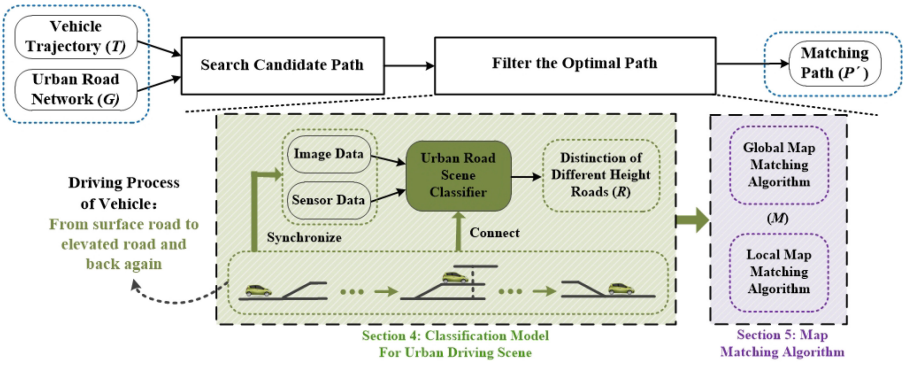


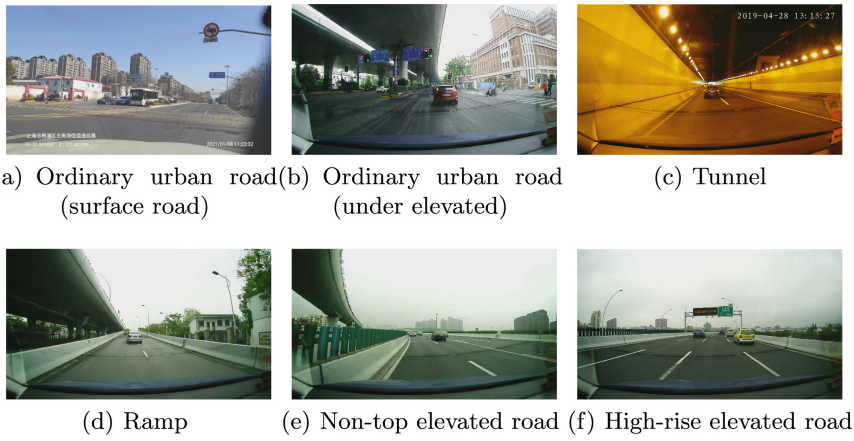
Fig. 2. The overall framework of the **HAUM³**. The upper part shows the overall operation of the map matching algorithm with height awareness; the lower part shows the schematic diagram of filtering the optimal path. Firstly, the urban road scene classifier model that links the dynamic driving process of vehicles obtains the distinction of different height roads. Then the different types of map matching algorithms are fused with the distinction of different height roads, endowing the map matching algorithm with height awareness. The optimal path is finally filtered.

The above definitions lead to **HAUM³**, a Height Aware Urban Map Matching Mechanism with Spatio-Temporal Correlation we have developed and deployed in the past year, and the overall framework is shown in Fig. 2. The map matching algorithm mainly consists of two steps. First, it searches for candidate paths in the map using the input trajectory information. After that, it selects the optimal matching path based on the rules and algorithms, and finally obtains the matching path. The proposed mechanism fuses the spatio-temporal correlation urban road scene classifier into searching for the optimal path, so that the optimal path searched is the one that matches the current actual road type. The classifier associates the vehicle's spatio-temporal state with the urban road scenes from the perspective of the vehicle's dynamic driving process. During the driving process, the images and sensor information collected synchronously by the vehicle are inputted into the urban road scene classifier. The obtained results are fused with the map matching algorithm to endow the map matching algorithm with height awareness, which leads to a better optimal matching path. The details are described in the following Sects. 4 and 5.

4 Classification Model for Urban Road Driving Scene

4.1 Scene Classification and Inspiration

The driving scene refers to the road environment in which the vehicle is moving. Due to current deficiencies in the map matching algorithm in complex urban roads, this paper first divides the vehicle driving scene into three main categories: elevated roads (including elevated expressways and ramps), ordinary urban roads (including elevated auxiliary roads and ordinary road scenes near elevated roads), and indoor roads (including tunnels and indoor parking area). In the elevated road scenes, they are further divided into ramp roads and elevated expressways. The elevated expressway scenes are further divided into relatively high-rise elevated roads (relatively high-rise interchanges) and non-top elevated roads (non-top interchanges). Figure 3 shows the images corresponding to the above five driving scenes.



(a) Ordinary urban road (surface road) (b) Ordinary urban road (under elevated) (c) Tunnel

(d) Ramp (e) Non-top elevated road (f) High-rise elevated road

Fig. 3. Various driving scenes recorded images.

Based on the above classification, the following urban driving scene indicator matrix is defined:

$$\boldsymbol{P}_{i,rtype} = [P_{i,ordinary} \quad P_{i,indoor} \quad P_{i,ramp} \quad P_{i,high} \quad P_{i,low}]$$

In the above equation, i denotes the ith sampling point, and $P_{i,ordinary}$, $P_{i,indoor}$, $P_{i,ramp}$, $P_{i,high}$, $P_{i,low}$ denotes the probability that the current sampling point belongs to the ordinary urban road, indoor road, ramp road, relative high-rise elevated road and non-top elevated road, respectively.

Note that in the following, O, I, R, H, and L denote the above five road types, respectively.

Through a large number of observations, this paper finds that vehicles driving in complex urban road scenarios satisfy the following laws: vehicles can reach any elevated road from ordinary urban roads through ramp roads, and then from elevated roads through ramp roads can go up or down to elevated roads, or return to ordinary urban roads. The specific diagram is shown in Fig. 4.

4.2 Urban Road Scene Classification Model with Spatio-Temporal Correlation

Inspired by the abovementioned laws, this paper proposes an urban road scene classification model based on images and sensor information with spatio-temporal correlation. The model inputs time-series images and sensor data, producing a matrix of driving scene indicators for each moment. Given the hierarchical relationship among different driving scenes, the model generates three sub-classifiers based on the granularity of the scenes:

The sub-classifier C_1 takes in image information and outputs the probabilities of three types of road scenes: elevated road, ordinary road, and indoor road, which are defined as P^{1_1}, P^{1_2}, and P^{1_3}, respectively. Specifically, $P_{i,normal} = P_i^{1_2}$ and $P_{i,indoor} = P_i^{1_3}$.

The sub-classifier C_2 confines the driving scene to elevated road, taking in sensor information such as vehicle speed and pitch angle (roll angle/lateral angle) information. It then outputs to assist in determining the probability of the following two road scenes: ramp road as well as elevated expressway, defined as P^{2_1} and P^{2_2}. Specifically, $P_{i,ramp} = P_i^{1_1} \cdot P_i^{2_1}$.

The sub-classifier C_3 confines the driving scene to elevated expressway, taking in image information, and outputs to assist in determining the probability of the following two road scenes: relative high-rise elevated road and non-top elevated road, defined as P^{3_1} and P^{3_2}. Specifically, $P_{i,high} = P_i^{1_1} \cdot P_i^{2_2} \cdot P_i^{3_1}$, and $P_{i,low} = P_i^{1_1} \cdot P_i^{2_2} \cdot P_i^{3_2}$.

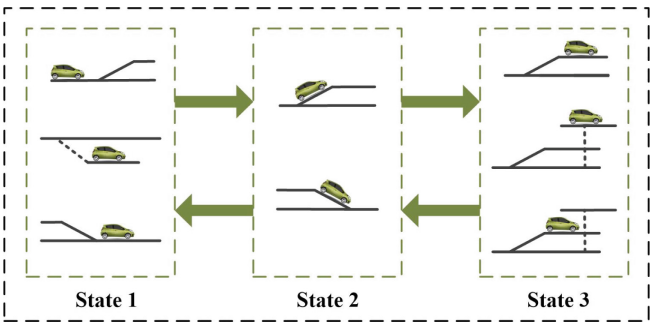


Fig. 4. Schematic diagram of the switching of vehicle driving states in different space-time states. The scenes of state 1 include: ordinary urban roads and indoor roads; the scenes of state 2 include: ramp roads; the scenes of state 3 include: relatively high-rise elevated roads and non-top elevated roads. The combination of scenes in various states can portray all the vehicle dynamic driving process.

The calculation process of the three sub-classifiers derived from the urban driving scene classification model based on images and sensor information with spatial-temporal correlation is shown in Fig. 5.

When the vehicle travels on the urban road network or the current map matching algorithm finds multiple matching roads, the mechanism process is initiated, and images and sensor data are collected synchronously. Firstly, the

collected image data is input to classifier C_1. Suppose the computed matching result is an ordinary urban or indoor road, corresponding to state 1. In that case, the probability result can be directly output, and the subsequent process is not continued. If the computed matching result is an elevated road $\left(P^{1_1} > V_1,\ simultaneously\ activating\ classifier\ C_2\right)$ and combined with the output of classifier C_2, it can be judged whether the vehicle is on a ramp section $\left(P^{1_1} \cdot P^{2_1} > V_2\right)$, corresponding to state 2. The probability result is output, and the process continues to run. Assuming that the vehicle has passed through the ramp section and is driving on the road above the elevated road, combined with the output of classifier C_2, it can be determined that the vehicle is currently on the elevated expressway $\left(P^{1_1} \cdot P^{2_2} > V_3,\ simultaneously\ activatingclassifier\ C_3\right)$, corresponding to state 3.

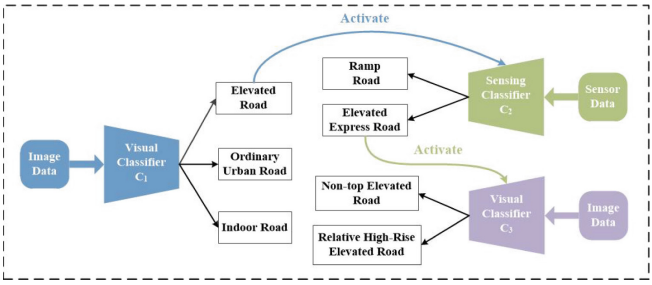


Fig. 5. The running process of three sub-classifiers.

Likewise, in the scene of the elevated expressway, if the current vehicle is about to exit the elevated road, i.e., the current section is a ramp section, corresponding to state 2, combined with the output of classifier C_1 and classifier C_2, it can be determined whether the vehicle is on a ramp section. After this, there are two possible scenarios. In scenario 1, the vehicle enters the elevated expressway located below (or above), corresponding to state 3, and the mechanism continues to run. In scenario 2, the vehicle enters an ordinary urban road, corresponding to state 1, and the process ends, with only classifier C_1 continuing to output. The specific schematic of this process is shown in Fig. 4 and Fig. 5.

Note that when the classifier is not activated, its output results are equiprobable. That is, when classifier C_2 is not activated, P^{2_1} and P^{2_2} are both equal to $\frac{1}{2}$; when classifier C_3 is not activated, P^{3_1} and P^{3_2} are both equal to $\frac{1}{2}$.

According to the above, using the visual images and sensor information collected synchronously during vehicle travel, it is possible to effectively distinguish between the four types of roads at different heights: underground roads, ground-level roads, ramp roads, and elevated expressways. In addition, visual image information can be used to distinguish between elevated expressways at different heights, further refining the granularity of the scene in elevated scene. At the same time, by combining with the dynamic driving process of the vehicle, it is possible to reduce the system error caused by the sub-classifier. That is, the

mutual switching of state 1 and state 3 must pass through state 2, which can be corrected when the classifier C_1 has an erroneous output.

5 Height Aware Map Matching Algorithm

The urban road driving scene classification model is applied to obtain a driving scene indicator matrix for each moment. A height aware map matching algorithm can be obtained by merging this matrix with the map matching algorithm. Currently, there are two types of map matching methods for GNSS trajectory data: global algorithms and local algorithms. However, regardless of the type of map matching algorithm, candidate paths are first searched in the map, and then the optimal matching path is selected through different methods. This paper takes this as a starting point. It uses the classic HMM algorithm (global algorithm) and Incremental algorithm (local algorithm) as examples to illustrate how to integrate the above scene classification model into the existing map matching algorithm, giving it the ability to perceive different heights of roads.

5.1 Height Aware Global Map Matching Algorithm

Currently, global algorithms are mostly derived from HMM map matching algorithm [9]. Based on HMM map matching, this is a decoding problem that seeks to find the hidden sequence that generates the observed sequence under certain observations. The hidden sequence represents the specific location of the vehicle equipped with a positioning device. In contrast, the observed sequence is a series of coordinate points generated by the GNSS trajectory produced by the positioning device.

This algorithm has multiple candidate paths for a GNSS point within a certain road network distance. Each candidate path is represented as a vertex in the Markov chain and is characterized by its observation probability. The candidate road segment set is defined as C. For the jth candidate road segment C_i^j in the candidate road segment set C_i of the ith sampled point P_i in the trajectory, its observation probability is calculated as follows:

$$P\left(C_i^j\right) = \frac{1}{\sqrt{2\pi}\delta} e^{-\frac{(x_i^j)^2}{2\delta^2}} \cdot \boldsymbol{P}_{i,rtype}^j \tag{1}$$

where x_i^j is the vertical distance between the current sampling point t_i and the candidate road segment C_i^j, and δ is the GNSS positioning error.

The weights between adjacent GNSS candidate points are used as edges in a Markov chain and are represented by the transfer probabilities. The transfer probability V is calculated as:

$$V\left(C_{i-1}^l \to C_i^s\right) = \frac{d_{i-1\to i}}{w_{(i-1,l)\to(i,s)}} \tag{2}$$

where, the numerator denotes the spatial distance between two adjacent GNSS points; the denominator denotes the shortest road network distance between two adjacent candidate points.

Finally, the dynamic programming algorithm is used to find the optimal path in the road network that maximizes the product of observation probability and transfer probability as the final matching path. The specific solving process is shown in Algorithm 1.

GNSS points are more affected by noise in complex urban roads, and adding the driving scene indicator matrix makes the global map matching algorithm more robust in complex urban roads. At the same time, a more correct, comprehensive view of the trajectory can be obtained.

Algorithm 1: Dynamic Programming

Input: Trajectory $\boldsymbol{T} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_n}\}$ and Urban Road Network $\boldsymbol{G}$.
Output: Matching Path $\boldsymbol{P'} = \{\boldsymbol{e'_1}, \boldsymbol{e'_2}, \ldots, \boldsymbol{e'_n}\}$.

Let $Func[\,]$ be the joint probability corresponding to the current state and $Prev[\,]$ be the previous state of the current state.
for *i=2 to n* **do**
  Get the sets of candidate paths C_{i-1} and C_i corresponding to sampling points t_{i-1} and t_i from G;
  for *each* C_i^s *in* C_i **do**
    `// Observation probability.`
    Calculate $P(C_i^s)$
    for *each* C_i^l *in* C_{i-1} **do**
      `// Initialization.`
      if *i=2* **then**
        Calculate $P(C_{i-1}^l)$
        $Func[C_{i-1}^l] = P(C_{i-1}^l)$
      end if
      `// Transfer probability.`
      Calculate $V_l^s = V(C_{i-1}^l \longrightarrow C_i^s)$
    end for
    $Func[C_i^s] = P(C_i^s) \cdot Max_{C_{i-1}^l \in C_{i-1}} \left\{Func[C_{i-1}^l] \cdot V_l^s\right\}$
    $r = Max_{C_{i-1}^l \in C_{i-1}} \left\{Func[C_{i-1}^l] \cdot V_l^s\right\}$
    $Prev[C_i^s] = r$
  end for
end for
`// Get the global optimal path endpoint.`
$End = Argmax_{C_n^k \in C_n} \left\{Func[C_n^k]\right\}$
$P'.append(End)$
for *i=2 to n* **do**
  `// Backtracking process.`
  $End = prev[End]$
  $P'.append(End)$
end for
$P'.reverse(\,)$
return $\boldsymbol{P'}$

5.2 Height Aware Local Map Matching Algorithm

The local algorithm must obtain a local optimal solution and gradually advance toward the global solution. The Incremental algorithm [7] uses a weighted calculation of the similarity of candidate paths based on the distance and angle difference. The distance measurement from the sampling point t_i to each candidate road segment C_i^j and the direction difference are defined respectively as:

$$S_d\left(t_i, C_i^j\right) = \mu_d - a \cdot d\left(t_i, C_i^j\right)^{n_d} \tag{3}$$

$$S_\alpha\left(t_i, C_i^j\right) = \mu_\alpha \cdot cos\left(\alpha_{i,j}\right)^{n_\alpha} \tag{4}$$

where μ and n are scale factors, a is the distance scale, and α is the angular difference.

Algorithm 2: Look-Ahead

Input: Trajectory $\boldsymbol{T} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_n}\}$ and Urban Road Network $\boldsymbol{G}$.
Output: Matching Path $\boldsymbol{P'} = \left\{\boldsymbol{e'_1}, \boldsymbol{e'_2}, \ldots, \boldsymbol{e'_n}\right\}$.

for *i=1 to n-k* **do**
  Get the candidate sets $\{C_i, C_{i+1}, \ldots, C_{i+k}\}$ of sample point t_i and its next k sample points from G;
  for *each* C_i^j *in* C_i **do**
    `// Composite score.`
    Calculate S_i^j
    if $Type\left(Max\left(P_{i,rtype}^j\right)\right) == Type\left(C_i^j\right)$ **then**
      `// Conformance to the scene.`
      Calculate A_i^j
      `// Composite look-ahead score.`
    end if
  end for
  $e'_i = Max_{C_i^j \in C_i}\left\{A_i^j\right\}$
  $P'.append(e'_i)$
end for
`// Processing residual candidates.`
for *i=n-k to n* **do**
  for *each* C_i^j *in* C_i **do**
    Calculate S_i^j
  end for
  $e'_i = Max_{C_i^j \in C_i}\left\{S_i^j\right\}$
  $P'.append(e'_i)$
end for
return $\boldsymbol{P'}$

The urban driving scene indicators were integrated into the calculation of the composite results, defining the composite scores as follows:

$$S_{\Theta}\left(t_i, C_i^j\right) = (S_d + S_{\alpha}) \cdot \boldsymbol{P}_{i,rtype}^j \tag{5}$$

the higher the score of this metric, the better the match.

Finally, a look-ahead algorithm is used to calculate the total composite score after advancing k steps along all candidate road sections, from which the one with the highest similarity is selected as the matching result. The look-ahead score A is defined as:

$$A\left(t_i, C_i^j\right) = \sum_{k,l=0}^{depth} S_{\Theta}\left(t_{i+l}, C_{i,k}^j\right) \cdot \boldsymbol{P}_{i,rtype}^j \tag{6}$$

For the convenience, $S_{\Theta}\left(t_i, C_i^j\right)$ is shortened to S_i^j, $A\left(t_i, C_i^j\right)$ is shortened to A_i^j.

The specific solving process is shown in Algorithm 2.

With the addition of the driving scene indicator matrix, only candidate paths that match the driving scene are considered in the look-ahead process, which reduces unnecessary calculations and makes the local map matching algorithm converge faster.

5.3 Buffer Zones

All the fusion mechanisms mentioned above are based on absolute trust in the scene classification model. However, when vehicles switch between different road types while passing through the ramp road, there is an unavoidable delay and fluctuation in the classification results of the classifier. In order to ensure that the output results of driving scene indicator matrix do not mislead the process of map matching, therefore a particular buffer zone can be set before and after the road section passing through the ramp section, and the road section in this buffer zone will not be fused with the driving scene indicator matrix.

The situations where the road type is switched when passing through the ramp road are divided into two cases. Case 1: switching between ramp roads and ordinary city roads; Case 2: switching between ramp roads and elevated expressways. The experimental results of the above classifier show that the delay in case 1 is slightly smaller than in case 2. According to [13], the length of the ramp road is generally more than 150 m, with a speed limit of 60 km per hour. Therefore, the buffer zone for case 1 is set to 20 m, and the buffer zone for case 2 is set to 30 m.

6 Evaluation

6.1 Dataset

To evaluate the effectiveness of the mechanism proposed in this paper in real-world urban scenarios, OpenStreetMap data and three sets of driving trajectories

located in Shanghai [12] were selected for the experiments. Specific information can be found in Table 1 and Fig. 6. As OSM is a two-dimensional map and does not provide information on the number of layers of elevated roads, this attribute was manually added. Trajectories 1 and 2 were collected under normal conditions with good weather, suitable lighting, and clear images. Trajectory 3 was collected between 5–6 am, with soft lighting, poor weather conditions, and heavy rainfall throughout some sections, resulting in poor visibility. Trajectories 1–3 passed through complex urban road areas such as the Inner Ring Elevated Road, Middle Ring Elevated Road, Yixian Elevated Road, Dabaishu Interchange, Handan Road Tunnel, Outer Ring Interchange, Xinzhuang Interchange, and Gonghexin Road Interchange in Shanghai.

Table 1. Experimental Trajectory Specifics.

Parameter	T_1	T_2	T_3
Total Points	1441	1337	3695
Sampling Frequency $/s$	1	1	1
Track Length $/km$	24.89	23.73	73.91
Number of Road Sections	145	92	160
Light Conditions	Bright	Bright	Dim

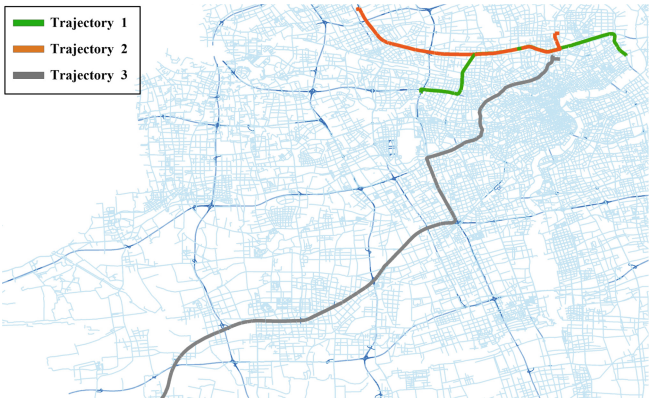


Fig. 6. Experimental Trajectory.

6.2 Map Matching Evaluation Methodology

In order to evaluate the improvement in performance of different types of map matching algorithms using this mechanism, the present paper selected the following three algorithms as baselines:

(1) The shortest distance map matching algorithm [14] (referred to as Point to Curve algorithm). This algorithm uses only the distance metric and chooses the path closest to the sampling point as the matching result.
(2) The local increment map matching algorithm [7] (referred to as Incremental algorithm). It uses a look-ahead algorithm to calculate the total similarity after moving k steps along all candidate paths, choosing the path with the highest similarity as the matching result.
(3) The Hidden Markov Model map matching algorithm [9] (referred to as HMM algorithm). The matching path with the highest global possibility is output by calculating the output probability and transition probability between adjacent states.

In this paper, match rate, precision, recall, and error rate are selected as the evaluation metrics for the algorithms.

Match rate is defined as the proportion of correctly matched points among all localized points in a trajectory [12]. For the HMM-based algorithms, since the algorithm output is a set of segments, the matching result for a single point is the nearest segment in the matching result.

$$Match\ Rate = \frac{N_{correct}}{N_{all}} \tag{7}$$

where $N_{correct}$ denotes the number of correctly matched positioning points with images and N_{all} denotes the total number of positioning points for the experimental trajectory.

Precision is defined as the percentage of the correct length of the matched path P' is calculated.

$$Precision = \frac{length\,(P \cap P')}{length\,(P')} \tag{8}$$

Recall is defined as the percentage of matching paths P' that cover the actual paths is calculated.

$$Recall = \frac{length\,(P \cap P')}{length\,(P)} \tag{9}$$

The above length (X) is the total length of the path X, not the number of edges. For example, length (P') is the length of the matching path P'.

The F_1 score is the summed mean of precision and recall, which combines precision and recall, and is defined as follows:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

Using the F_1 score to define the error rate [6]:

$$Error\ Rate = 1 - F_1 \tag{11}$$

6.3 Driving Scene Classification Model

In order to give the model some degree of generalization, this paper collected driving recorder image data and sensor data from different vehicles under different weather conditions (sunny, light rain, moderate rain, heavy rain). Each image's urban driving scene category was manually labeled, and the sensor data under different elevated road scenes were extracted. The images of the elevated expressway were labeled again. 80% of the data was randomly selected as the training set, and the remaining 20% of the data and the images and sensor data of the three trajectories were set as the test set.

Since image data is relatively simple and easily distinguishable, and there is little correlation between sensor data, classifiers C_1 and C_3 used the ResNet-18 [15] for fine-tuning transfer learning. The input is image data, and specific data augmentation is performed, including random rotation, horizontal and vertical flipping, and Gaussian blur. Classifier C_2 used the SVM [16] for learning from scratch, with the input being vehicle speed, pitch angle, roll angle, and lateral angle information. Combining the dynamic running process of the vehicle and the test results of the three sub-classifiers, the calculated results obtained by the urban road driving scene classification model are shown in Table 2.

Table 2. Classification results of urban driving road Scenes.

Indicator	O	I	R	H	L
$\boldsymbol{P}$	98.8%	100%	96.7%	98.0%	98.3%
$\boldsymbol{R}$	90.7%	100%	94.1%	95.1%	95.4%
$\boldsymbol{F}_1$	94.5%	100%	95.4%	96.6%	96.9%

6.4 Performance on Complex Urban Roads

This subsection evaluates the performance of the map matching algorithm on complex urban road sections. To ensure that the candidate set contains all relevant roads, d in the Incremental and HMM algorithms is taken as $50m$, and other parameters are recommended in the original paper. The baseline method and the method incorporated with the mechanism M of this paper are applied to three trajectories and evaluated experimentally at different sampling intervals (1–10 s). The results are shown in Table 3 and Fig. 7.

According to Table 3, at sampling intervals of 1–10 s, the three baseline algorithms exhibited an average precision increase of 11.9%, 10.1%, and 10.9%, respectively, after incorporating the result information of the proposed urban driving scene model with spatio-temporal correlation in this paper. The average recall rates also increased by 7.5%, 7.8%, and 8.9%, respectively, while the average error rates decreased by 10.4%, 9.1%, and 10.1%. These indicate that in complex urban road scenarios, combining different map matching algorithms with the mechanism proposed in this paper can rectify mis-matchings in vertical road matching caused by differences in road height, thereby improving the matching performance of the algorithms.

Table 3. Average matching results of the 3 methods.

Method	Match Rate	Precision	Recall	Error Rate
Point to Curve	75.7%	62.4%	81.7%	29.5%
$M(P)$	**85.6%**	**74.4%**	**89.2%**	**18.9%**
Incremental	83.7%	77.9%	87.5%	17.7%
$M(I)$	**93.3%**	**88.1%**	**95.3%**	**8.5%**
HMM	83.9%	77.6%	87.5%	18.3%
$M(H)$	**93.7%**	**88.5%**	**95.5%**	**8.2%**

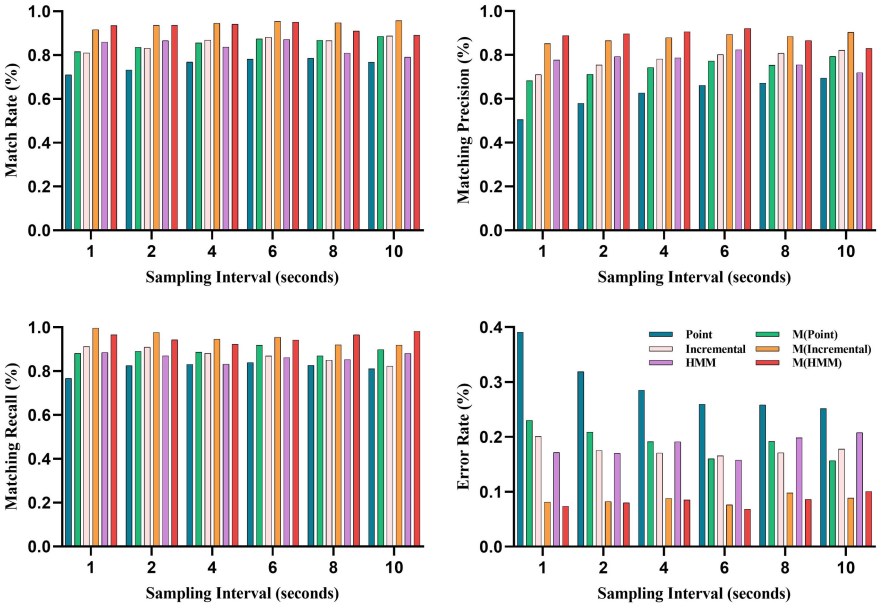
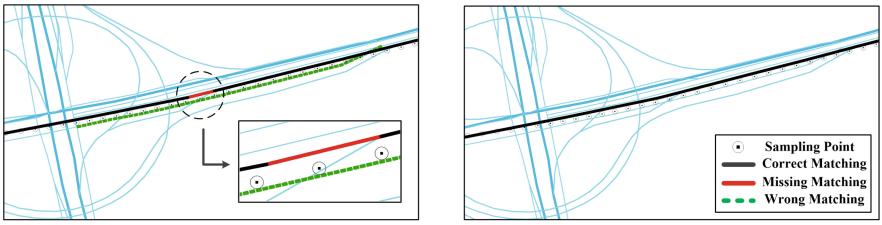


Fig. 7. Experimental results of the 3 methods at 1–10 s sampling interval.

Figure 8 compares the baseline algorithm and the boosted algorithm using the mechanism in the complex area. The fused mechanism proposed in this paper can better distinguish the elevated road from the parallel ground road, thus achieving better matching in the complex intersection and elevated road area. At the same time, the baseline algorithm has different degrees of wrong matching and missing matching.

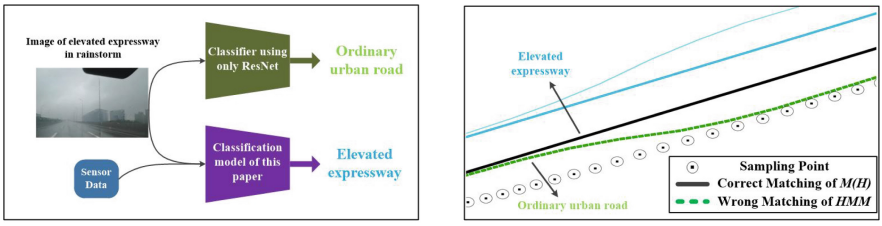
The mechanism proposed in this paper also provides the map matching algorithms with some resistance to weather effects. As shown in subplot (a) of Fig. 9, in the stormy road section of Trajectory 3, the vehicle images are blurred due to the severe weather conditions. At this time, it is very likely that the image of the elevated expressway will be classified as an ordinary urban road using only the image classifier of ResNet; while the classification model proposed in this

paper, which combines the different spatio-temporal states of the vehicle, can classify the image correctly. Since the classification model can transfer the previous correct classification results to the subsequent process, which reduces the impact of weather conditions on map matching. This situation is also reflected in the map matching algorithm. As shown in subplot (b) of Fig. 9, the map matching algorithm $M(H)$, which incorporates the mechanisms proposed in this paper, correctly matches the elevated expressway even in the stormy section of Trajectory 3; while the baseline algorithm HMM obtains an incorrect match.



(a) Incremental algorithm (b) Mechanism of this paper: $M(I)$

Fig. 8. The matching performance of the algorithm in complex urban roads with a sampling interval of 1s on Trajectory 2.



(a) Classification result (b) Matching result

Fig. 9. Demonstration of the results of different classifiers and different matching algorithms on a complex urban road with a sampling interval of 1s on Trajectory 3 under bad weather. Subplot (a) shows the recognition results of different classification models for elevated expressways under bad weather conditions. Subplot (b) shows the matching results between the map matching mechanism $M(H)$ in this paper and the baseline algorithm HMM.

6.5 Performance of Map Matching Algorithms

In terms of the performance of the map matching algorithm, the HMM algorithm, after incorporating the mechanism in this paper, maintains a better accuracy and lower error rate at different sampling rates, as shown in Fig. 7. Therefore, the mechanism of this paper makes the global algorithm more robust. The

Incremental algorithm, too, has improved the convergence speed by incorporating the mechanism of this paper, as shown in Table 4.

The performance of the Incremental algorithm is better at high sampling rates, as the local algorithm only obtains the matching section corresponding to the sampling point. As the distance between sampling points increases, there may be a situation where some intermediate sections are skipped, resulting in a decrease in performance as the sampling rate decreases. On the other hand, the HMM algorithm performs better at low sampling rates, assuming that each GNSS point in the global context is equally important. Due to noise in GNSS data, the error rate is higher when sampling frequently. As the distance between sampling points increases with increasing sampling intervals, a small number of GNSS points can provide a better overall direction of travel. Similarly, after incorporating information from the spatio-temporal correlated urban road scene classifier, the overall trend of map matching algorithms does not change at different sampling rates, as seen in Fig. 7. This suggests that the proposed mechanism relies on the baseline map matching algorithm, which improves the performance of different map matching algorithms, especially in complex urban sections such as elevated bridge sections, while still relying on the matching ability of the baseline algorithm itself in ordinary sections.

Table 4. Average single point matching time.

Sampling interval	1	2	4	6	8	10
Incremental ms	31.3	42.2	46.7	59.1	76.7	87.5
$\boldsymbol{M}$ (I) ms	**26.8**	**38.1**	**42.8**	**55.2**	**73.6**	**84.6**

7 Conclusion and Future Work

This paper proposes **HAUM^3**: A Height Aware Urban Map Matching Mechanism with Spatio-Temporal Correlation. This addresses the current lack of height information in map matching algorithms. On the one hand, the mechanism starts from the dynamic driving process of vehicles and designs a spatio-temporal correlation urban driving scene classification model based on the synchronized images and sensor data generated by vehicles, which can well distinguish five different heights of roads from a fine-grained level. On the other hand, according to the characteristics of different types of map matching algorithms, the classification model is integrated into traditional map matching algorithms, giving them height awareness. This improves the performance of the map matching algorithms and resists the influence of weather conditions to a certain extent. According to the evaluation results in the urban area of Shanghai, this mechanism significantly improves the accuracy of different map matching algorithms on complex urban roads, injecting new vitality into different traditional map matching algorithms.

There are still some ways to improve the mechanism in the future. First, this paper only distinguishes road scenes with different heights from the vertical direction, and the distinction of parallel roads is also an important issue. Second, future work could use image information as well as other on-board data to identify more specific scenes, such as side roads, sidewalks, etc. Thirdly, based on the road level matching achieved in this paper, lane level map matching methods for vehicles can also be further analyzed by using lane lines and other information in the images.

Acknowledgements. This work was supported by National Natural Science Foundation of China (62372188), GuangDong Natural Science Foundation (2021 A1515011755), as well as thanks to Yiren Technology Co. (Shanghai) for providing the experimental dataset.

References

1. Kubicka, M., Cela, A., Mounier, H., Niculescu, S.-I.: Comparative study and application-oriented classification of vehicular map-matching methods. IEEE Intell. Transp. Syst. Mag. **10**(2), 150–166 (2018)
2. GPS. Gps accuracy. https://www.gps.gov/systems/gps/performance/ (2020)
3. Ho, P.F., Hsu, C.C., Chen, J.C., Zhang, T.: Using barometer on smartphones to improve GPS navigation altitude accuracy. In: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, pp. 741–743 (2018)
4. Gong, Y., Zhu, Y., Yu, J.: Deel: detecting elevation of urban roads with smartphones on wheels. In: 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 363–371. IEEE (2015)
5. Zhang, X., Xu, H., Yang, J., Sun, J., Chen, F., Li, L.: Elevated road network: a metric learning method for recognizing whether a vehicle is on an elevated road. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2925–2932 (2020)
6. Li, H., Kulik, L., Ramamohanarao, K.: Robust inferences of travel paths from GPS trajectories. Int. J. Geogr. Inf. Sci. **29**(12), 2194–2222 (2015)
7. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 853–864 (2005)
8. Xiao, F., Zhang, J., Zhang, Y.: An online map matching algorithm based on second-order hidden markov model. J. Adv. Transp. **1–12**, 2021 (2021)
9. Newson, P., Krumm, J.: Hidden markov map matching through noise and sparseness. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 336–343 (2009)
10. Cui, G., Bian, W., Wang, X.: Hidden markov map matching based on trajectory segmentation with heading homogeneity. GeoInformatica **25**, 179–206 (2021)
11. Huang, Z., Qiao, S., Han, N., Yuan, C., Song, X., Xiao, Y.: Survey on vehicle map matching techniques. CAAI Trans. Intell. Technol. **6**(1), 55–71 (2021)
12. Fu, C., Huang, S., Tang, Y., Wu, H., Liu, C., Yao, L., Huang, W.: A real-time map matching method for road network using driving scenario classification. Acta Geodaetica et Cartographica Sinica **50**(11), 1617 (2021)

13. Baidu. Ramp (2023). http://baike.baidu.com/view/414889.htm?fr=aladdin
14. White, C.E., Bernstein, D., Kornhauser, A.L.: Some map matching algorithms for personal navigation assistants. Transp. Res. Part c: Emerg. Technol. **8**(1-6), 91–108 (2000)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
16. Chauhan, V.K., Dahiya, K., Sharma, A.: Problem formulations and solvers in linear SVM: a review. Artif. Intell. Rev. **52**(2), 803–855 (2018). https://doi.org/10.1007/s10462-018-9614-6

A Resource-Efficient Approach of GNSS Activation for Pedestrian Monitoring

Asiimwe Paddy Junior[1,3(✉)], Luis Enrique Díez[1], Alfonso Bahillo[2], and Odongo Steven Eyobu[3]

[1] Faculty of Engineering, University of Deusto, Avenue Universidades, 24, 48007 Bilbao, Spain
{pjasiimwe,luis.enrique.diez}@deusto.es

[2] Department of Signal Theory and Communications, University of Valladolid, Campus Miguel Delibes, Paseo de Belén, 15, 47011 Valladolid, Spain
alfonso.bahillo@uva.es

[3] Makerere University, Kampala, Uganda
odongo.eyobu@mak.ac.ug

Abstract. Remote monitoring provides effective solutions that bridge the gap between healthcare and where the elderly actually desire to live. In underdeveloped countries, the adoption rate of such platforms is exceptionally low due in part to the lack of resources that these systems take for granted. In other words, these systems are designed for rich countries but are also needed in resource-constrained environments. Global Navigation Satellite Systems (GNSS) are one of the main technologies for remote monitoring, though they consume relatively high amounts of energy. GNSS activation, i.e., varying the position update rate and turning the receiver on/off, is one of the main GNSS power optimization strategies, which has two existing methods: user activity-based and position-based. The former is usually too simple, suitable only for objects such as cars but not pedestrians. The latter requires installing beacons around users' premises, which is more expensive. We are looking for a method that is smarter than checking the acceleration and cheaper than using beacons for GNSS activation. This work proposes a position-based method using a Pedestrian Dead-Reckoning (PDR) system. PDR is a navigation method that uses inertial sensors (accelerometer, gyroscope, magnetometer) to estimate position.

Experimental evaluations show that this proposed GNSS activation method achieves higher power optimization (by more than 90% in and outside the safe zone) than acceleration-based ones without installing any beacon, making it better for resource-constrained environments. Also, since the proposed method depends on the user's estimated position, the

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the MarieSkłodowska-Curie grant agreement No. 847624. In addition, a number of institutions, including the Government of Uganda, through the Makerere University Research Innovation Fund (RIF), back and co-finance this project. This paper reflects only the author's view, and the Research Executive Agency is not responsible for any use that may be made of the information it contains.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 526–546, 2024.
https://doi.org/10.1007/978-3-031-63989-0_28

effect of the PDR's performance is assessed by evaluating two different PDR implementations.

Keywords: GNSS activation · Pedestrian Dead-Reckoning · Resource-constrained environments

1 Introduction

The shift in a country's population distribution towards older ages started in highly developed countries, but now low and middle-income countries are also experiencing this change. By 2050, two-thirds of the world's population over 60 years will live in low- and middle-income countries [8]. This consistent increase in the number of elderly people represents a likelihood of a higher burden on healthcare, family, and social services and already resource-constrained low-income governments. Non-communicable diseases like cardiovascular diseases, diabetes, dementia, hypertension, and cancer are mainly a result of old age. [4]. Therefore, many of these older adults need support from caregivers as they cannot live independently. Several studies show that people with dementia are at least 60% likely to wander in open areas. Wandering as one of the challenges facing dementia patients can be dangerous and life-threatening. The stress of this risk weighs heavily on caregivers and families [19].

In order to save costs, health care policy should shift from institutionalization to aging in place (in the community). A recent American Association of Retired Persons (AARP) survey has confirmed that most people over 65 prefer to stay home, even if they require continuous assistance and health care [3]. Therefore, there is an increase in the demand for technologies to address the urgent service concerns of the aging population. Remote monitoring solutions using non-intrusive, non-invasive actuators, wearable sensors, and communication technologies offer efficient solutions. These technologies bridge the gaps between healthcare and where older adults prefer to live. When properly applied, such technologies will not only assure an appropriate quality of life among older people in their homes but will also support the family and caregivers in delivering adequate services to these older adults in society. [24].

Several remote localization platforms are available on the market, but their adoption rate is extremely low in low-developed countries and rural areas. One primary reason is the scarcity of resources that these systems take for granted. They presume the availability of resources and infrastructures such as cellular networks, Wi-Fi networks, the Internet, digital literacy, and access to the power grid. In other words, these systems are created for rich countries but are also vitally needed in poor and middle-income countries with resource-constrained environments [13].

Location information (positioning) is one the key variables as it allows us to remotely monitor the older person's behavior and infer some basic physical activity information relevant to their health status (step counter, walking speed, fall detection) [11]. Global Navigation Satellite Systems (GNSS) have been

the de facto standard for many remote outdoor infrastructure-free positioning applications for decades. GNSS can give dependable and accurate position data anywhere in the world, but it is not the ideal solution for all localization challenges. This is because GNSS technology has a very high energy consumption in its typical use case, which doesn't align well with the stringent constraints of battery-powered devices such as IoT devices, despite energy consumption being critical in resource-constrained environments. Additionally, remote location applications require the ability to monitor devices and data seamlessly. Therefore, a communication network is required, as the GNSS location must be communicated to the application server. From the systematic literature review conducted by authors in [13] about the state-of-the-art remote pedestrian monitoring systems for resource-constrained environments, it is evident that GNSS is still the best choice for remote long-range infrastructure-free localization. However, it is necessary to reduce the GNSS receiver energy consumption further. Therefore, research should aim to mitigate its challenges rather than eliminate its use in a resource-constrained environment.

Different GNSS power optimization strategies have been employed [2,12]. On the one hand, techniques such as assisted GPS (AGPS) [6,16], extended and autonomous ephemeris prediction or snapshot positioning [7,17,23] achieve great power reduction by using external information or delegating part of the GNSS processing, but require a communication channel (NB-IoT, GSM, LTE) to access the Internet with sufficient capacity and data rates, which are not available in resource-constrained environments.

The other type of power optimization strategy is varying the position update rate and turning on/off the receiver based on the user activity or position. By putting the receiver in sleep mode between location updates, the total energy consumption can be reduced significantly, especially in IoT use cases where a location update is only required every few hours, days, weeks, months, or even in instances where an event is triggered [9]. Two methods exist, i.e., user activity-based and position-based. Under activity-based, user activity parameters, such as acceleration or angular rate, have been used by different authors [1,5,10,15] to control the position update rates and GNSS activation. This method may work on objects that are either static or changing their position, such as cars, but may not be an optimal solution to optimize power in the case of human users as subjects because, by nature, human beings exhibit frequent motions that can be easily captured and interpreted as an activity to trigger the turn-on of the GNSS and yet it is a false alarm.

The position-based method uses the user position to eliminate unnecessary GNSS activations and/or position updates. Beacons such as Infrared and Bluetooth devices are commonly installed at the user's premises to determine the user's position around their premises. For example, authors in [5] used, in addition to an accelerometer to detect static periods, Bluetooth beacons and GNSS signal strength to determine if the user is indoors and then turn off the GPS. It should be noted that installing additional infrastructure comes at a higher cost

and works effectively well in planned, structured homes, commonly in urban centers.

By the nature of the resource-constrained environment use case, we are looking for a method smarter than just checking the acceleration and cheaper than using beacons for GNSS activation. We are, therefore, proposing a position-based method using the PDR system. This method does not need to install beacons at the user premises, making it the best fit for a resource-constrained environment, especially in poor rural areas in Africa with poor housing infrastructure. The proposed method is based on the definition of a geofence around the user's home such that the GNSS is only turned on when the user's position is estimated to be outside the geofence (safe zone). Our experiments were conducted in an open environment with limited interference to simulate the rural, low-resourced home scenario. LoRaWAN technology was used for ultra-low power requirements and long-range bidirectional communication of up to 15km or more in rural areas under Line-of-sight circumstances.

Experimental evaluations show that this proposed activation method achieves higher power optimization (by more than 90% in and outside the safe zone) than acceleration-based ones without installing any beacon. Also, in order to assess how position accuracy affects the performance of the proposed method, two different PDR systems have been implemented and evaluated.

The rest of this paper is organized as follows. Section 2 discusses relevant work related to the state of the art. Section 3 describes the proposed methodology. Details about the experiments and the results are presented in Sect. 4. Finally, this work is summarized and concluded in Sect. 5.

2 Related Work

The position update rate is the first thing to consider when attempting to lower a GNSS receiver's power consumption because some typical use cases only require position updates when an event is triggered, and thus, there is no need for continuous tracking [2]. This allows the GPS receiver to be turned off when not in use, minimizing the technology's power consumption.

Authors in [21] introduced unique techniques for cheaply determining whether and when GPS activations are required. At the heart of their approach is a method for estimating user velocity based on a history of previously measured velocities at the exact location and the same time of day; intuitively, this velocity estimation was leveraging consistency in user behavior, and when the estimated distance traveled approached a user-specified accuracy bound, their system (known as RAPS, the rate-adaptive positioning system) activated GPS. A duty-cycled accelerometer was used to estimate user movement/velocity efficiently. This proposal is designed for urban environments, where GNSS is generally less accurate, so it suffices to turn on GPS only as often as necessary to achieve this accuracy. In addition, the building of a location-time database makes the user device more complex and with higher communication requirements.

Similarly, authors in [20] developed an accelerometer-based algorithm for correctly detecting the user's mobility state. The novel algorithm was designed,

built, and tested on Android-powered mobile devices. It was used to control the process of turning on and off smartphone-based location-sensing technologies like GPS. Like in [21], just acceleration determines when to turn on/off the GPS, and as already explained in the introduction, it's not effective, especially with pedestrians.

The authors in [10] adopted and implemented a method called the Accelerometer Assisted GPS Model as designed by authors in [20]. The method utilized the embedded 3-axis accelerometer to distinguish whether the user kept stationary. To adjust the GPS sampling rate and reduce power consumption, the authors considered two motion patterns: movement and stationary. If a pedestrian is kept stationary, the system suspends the GPS sampling for positioning. In contrast, whenever a pedestrian started to move, their system adjusted the GPS sample rate based on the correlations between the maximum searching range, the service region, and the person's current location. Their work used the user position rather than just the mobility state. The user's position was determined in relation to the point of interest, which denotes a heritage, a landmark, a scenic spot, a neighboring restaurant, or a popular store, and to obtain neighboring information, pedestrians continuously update their current geographical locations via the cellular network, regardless of whether they are moving or stationary. Even though the authors use the idea of a geofence and user position, the system's design does not fit a resource-constrained environment as it requires a network and a reference point, like a restaurant or a popular store, to determine the user position, and these resources don't exist available in these rural resource-constrained environments.

The authors in [5] combined activity and position methods. An accelerometer was utilized to detect motion and count steps. If no motion was observed over a certain period, the pedestrian may have been stationary. To conserve power, the GNSS module was turned off unless it was activated by movement. Furthermore, Bluetooth beacon devices installed at the exit and entry of the house were used to determine the indoor position. The GNSS was turned off when the user was indoors with a poor signal. Installing beacons is not ideal for most resource-constrained environments because of the costs involved and the nature of the housing infrastructure found in these environments. Plus, the GPS receivers can continue getting a good signal even when the user is indoors because of the nature and materials used to set up those houses.

Therefore, to fill this open gap, a method smarter than user acceleration but cheaper than installing beacons is needed. We have reviewed the state of the art and identified the research gap; as far as we know, no work has been done on that.

3 Methodology

3.1 Pedestrian Monitoring System Overview

The proposed GNSS activation method will be included in a pedestrian monitoring system. This section describes the main blocks of that system as shown in

Fig. 1. Our proposed system has four blocks, i.e., power, sensor, control, and communication modules. The sensor module contains the positioning sensor (GNSS) and the inertial sensors (accelerometer and gyroscope). The communication module has a RAK11720 LoRaWAN, and the control module has an ESP32 microcontroller. A lithium-ion battery was used to power all the modules.

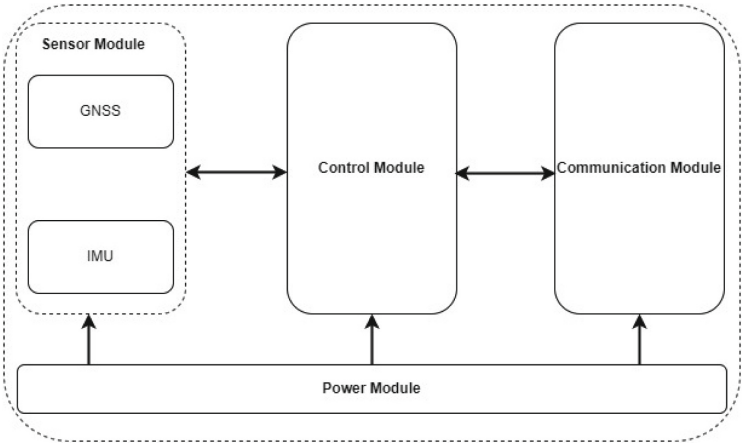


Fig. 1. Proposed system block diagram.

3.2 The Proposed Position-Based GNSS Activation Method

Figure 2 is the Workflow diagram of the proposed system. To easily monitor elderly persons around their homes, a geofence equivalent to a safe area is defined first. We assumed the user's home was at the center of the geofence (safe zone). When the user turns on the system, the inertial sensors and GNSS receiver are turned on and then off immediately after a reference point (reading of the coordinates for the initial user position) is acquired and stored permanently. The user is then requested to define the radius of the geofence (safe zone), and it's also stored permanently. We used the geofence idea to define the safe area around the user's home where less or no monitoring is needed. In our environment, we assumed the user could still get a GNSS signal even if indoors. A safe zone of a radius of 20 m around the user's home was assumed in this experiment.

In real-time, the user position is continuously monitored and estimated using a PDR system and checked to see if it is in the safe zone, and when the user position is estimated to be outside the safe zone radius, the GNSS receiver is turned on, and the reading is taken and compared with the initial position.

If the position is found to be inside the safe zone, the GNSS is turned off and marked as a false alarm. The user position on the PDR system is updated. Still, if the position is confirmed to be outside the safe zone, a message is sent through the LoRaWAN communication system to the caretaker for a quick response.

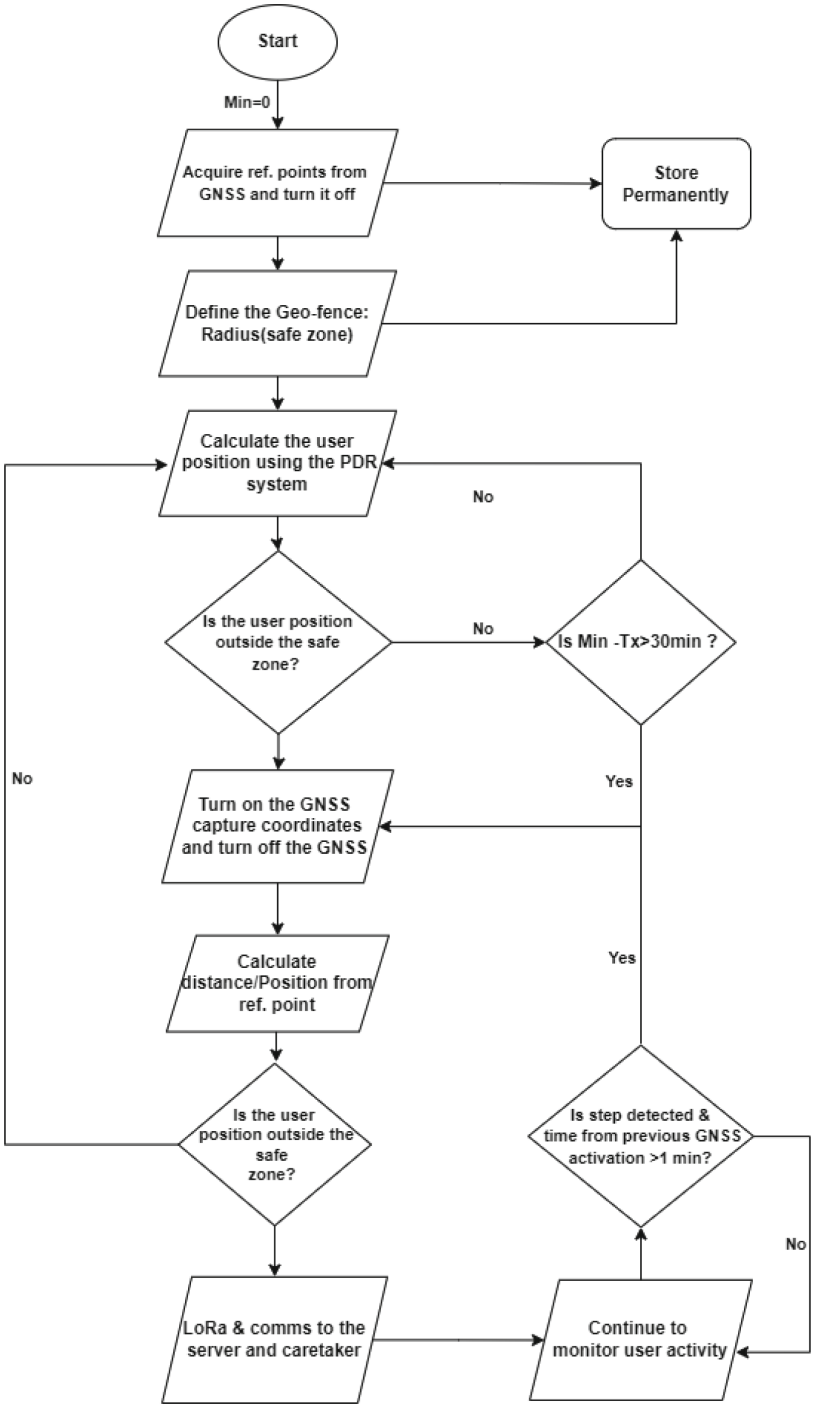


Fig. 2. Workflow diagram of the proposed system.

When the user is outside the safe zone and continues walking, the sampling rate for the GNSS receiver is set to 1 min, and the GNSS is turned off whenever the user is stationed to minimize energy consumption.

In that case, the user returns to the geofence as estimated by the PDR system; the GNSS is turned off until the user position is estimated to be outside the geofence or after 30 min. We used the 30-minute position update rate inside the safe zone for our work to prevent the GNSS receiver from going to the cold start. Power outages cause GNSS receivers to lose their position fix and all downloaded time and GNSS orbit data (ephemeris data), forcing them to undergo a complete cold start (up to 12.5 min) as soon as power is returned. Many GNSS receivers include a backup battery and a built-in EEPROM to boast the Time To First Fix (TTFF) [18]. So, by saving this data in backup RAM, GNSS receivers with a backup battery can recover faster (less than 30 s). This only works when the position update period is not longer than two hours, which roughly corresponds to the validity of the ephemeris data [2]. Therefore, the proposed method assumes that the GNSS receiver includes a backup battery and memory.

3.3 PDR System

The proposed GNSS activation and position update rate depend on the user position estimated by the PDR system. So, the accuracy with which the PDR system determines the position is key to the effectiveness of the proposed method. To assess this, two different PDR systems were implemented and evaluated. PDR systems are generally divided into four different blocks: step detection, step length estimation, heading estimation, and, finally, position integration. The two implementations differ only in the first two blocks, both in the algorithms and the input signal they use. The first one uses the pitch [22] of the inertial sensor, while the second one uses the magnitude of the sensor's acceleration. We chose those two implementations because the different step detectors are suitable for different mounting points. The performance of position estimation methods is determined by the accuracy of step detection and the pedestrian's step length (SL) estimation. The final blocks of this proposed algorithm, i.e., heading estimation and position integration, were implemented as explained in [22].

Pitch-Based PDR Implementation. Figure 3 represents the block diagram of the selected pitch-based PDR implementation, as explained in [22]. This proposed pitch-based approach detects steps using the pitch angle and estimates step length based on the pitch amplitude. We used a Kalman filter for accelerometer and gyroscope fusion [14] to calculate the roll and pitch of the inertial sensor. The step detection algorithm analyses the pitch and detects the step when local maximum and minimum pitch values occur in a certain interval. To avoid false step detection, the maximum and minimum values should be above or below a certain threshold, respectively.

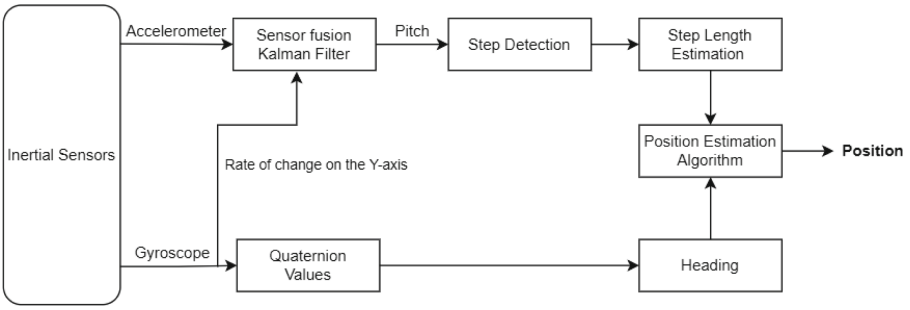


Fig. 3. Pitch-based PDR implementation.

Then, the step length is estimated using a first-order linear regression model on the pitch amplitude ($\Delta\theta$):

$$SL = a \times \Delta\theta + b \tag{1}$$

where $\Delta\theta$ is the difference between highest positive peak ($\theta_{\max}$) and lowest negative peak ($\theta_{\min}$), in degrees. The constants b and a are the personalized parameters fitting each regression line.

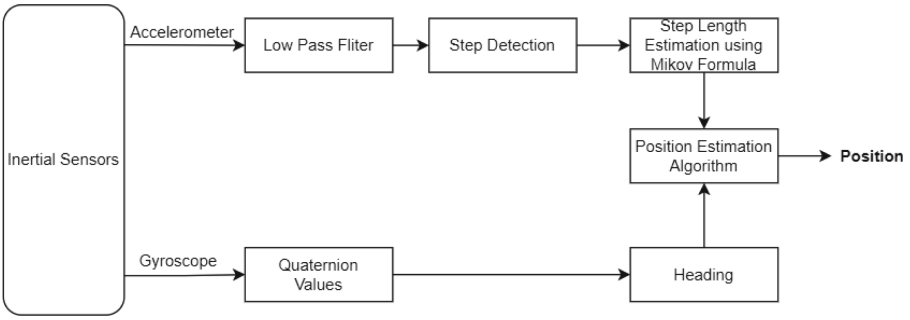


Fig. 4. Acceleration-based PDR implementation.

Acceleration-Based sPDR Implementation. Figure 4 shows the second selected PDR implementation, which utilizes the acceleration values of the accelerometer sensor's X, Y, and Z axes to count steps in real time. Then, the magnitude value is computed as the square root of the three accelerometer values, as shown in the equation below [22].

$$mag = \sqrt{x^2 + y^2 + z^2} \tag{2}$$

The magnitude value represents the vibration on the three-axis coordinates. The accelerometer sensors are inaccurate and suffer from various problems. Therefore, the derived data must be filtered to exclude the noise and the outliers' values. The derived data were filtered in this work using a low pass filter. Smoothing the data helps improve the system's accuracy, as the accelerometer sensor is very sensitive to the movements.

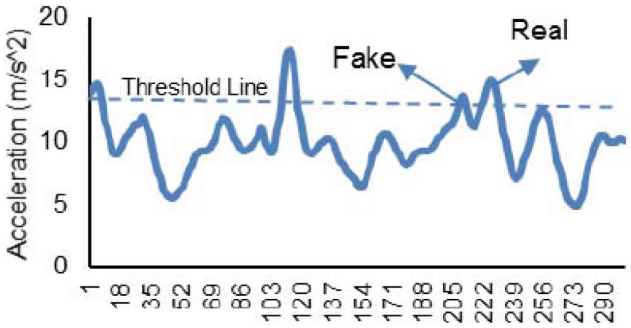


Fig. 5. Fake vs Real peaks.

Most current step detection algorithms rely on counting the peaks, as each peak represents a user's stride. A significant problem is the fake peaks. A fake peak is a peak caused by an irrelevant movement. Figure 5 shows the real and the fake peaks. Counting the fake peak as a real peak increases the error in the distance estimation. In this research, a peak detection method for the detection of the steps, as explained in [25], was used, and the following Mikov formula was used to compute step length.

$$L_{Mikov} = K_5 \cdot t_{step} \cdot \sqrt[4]{a_{\max} - a_{\min}} \tag{3}$$

where $a_{\min}$ and $a_{\max}$ are the minimum and maximum acceleration values as shown in Fig. 6, respectively, measured on the Z-axis in a single stride, t_{step} is step duration, and K1 is a constant for unit conversion (i.e., feet or meters traveled).

4 Experimental Evaluation

4.1 Experiments Description

In our experiments, we try to show that using just acceleration or steps to activate the GNSS is not good enough, especially with human subjects. So, this work proposes a smarter and more efficient algorithm for GNSS activation, using the user position around their home, as demonstrated in the experiments we conducted. In our experiments, we assumed that the user's home was at the center of our defined geofence.

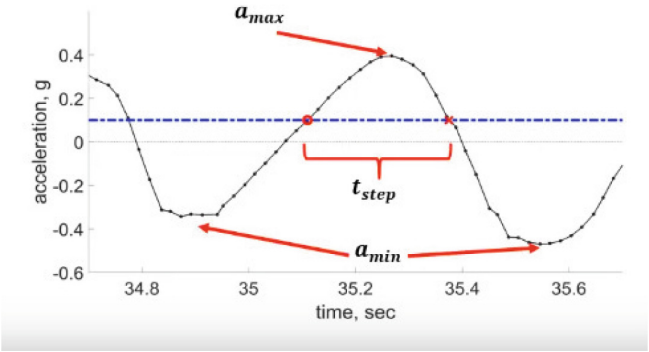


Fig. 6. Acceleration chart.

We conducted different experiments to assess the effectiveness of our proposed method, including experiments to evaluate the GNSS activation and the effect of position performance for the two PDR implementations. Two mounting positions were used to collect data, i.e., on the chest and in the swinging hands while walking.

(a). GNSS Activation evaluation

Here, our GNSS activation method was compared with a basic method we implemented inspired by work from the literature [20]. This simple system activates the GNSS whenever a step is detected. In this test, two evaluation metrics were used, i.e., when the GPS is activated/deactivated and power consumption.

(i). Experiment one:
Walking continuously inside, outside, and back inside the safe zone: This experiment aims to show that in our method, GPS is activated only outside the safe zone, hence getting some power saving against the other method.

(ii). Experiment two:
Walking continuously inside the safe zone for at least more than 30 min (to simplify the experiments, we used 5 min), and the objective of this experiment is to show that our method periodically activates the GPS if the user stays inside the safe zone, i.e., every after 30 min to refresh the position error accumulated by the PDR, and the satellite data to avoid a cold start.

(iii). Experiment three:
Walking and stopping inside, outside, and back inside the safe zone: This experiment aims to show that whenever the user stops outside the geofence, the GPS gets off until a motion/step is detected again.

(b). Effect of position performance

In this experiment, data were collected using two mounting positions: on the chest and in the hands.

(iv). Experiment four: Effect of length estimation and step detection.
In this experiment, the volunteer walked in a straight line inside, outside, and back inside with each PDR implementation for two different mounting points. This experiment aims to assess how the performance of step detection, step length estimation, and, therefore, position estimation affects the GNSS activation method performance.

(v). Experiment five: Effect of heading estimation
In this experiment, the volunteer walked in a zigzag motion from the center of the geofence to the outside and back. This was intended to assess the effect of the heading estimation on the final user position and, therefore, on the GNSS activation. The acceleration-based PDR implementation mounted on the chest was used for this experiment.

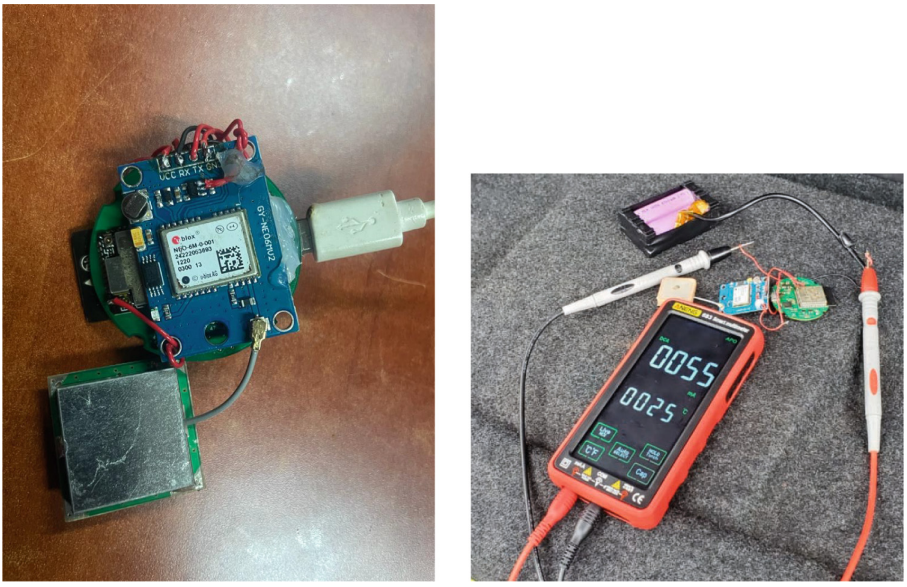
4.2 Data Collection

An embedded system on a custom-designed Printed Circuit Board (PCB) was designed and assembled for the tests as shown in Fig. 7a. A custom design allowed for miniaturization of the system as it was intended to be carried or worn by the user. The PCB integrated the **ESP32 microcontroller** (ESP32 integrates a power amplifier, RF balun, an antenna switch, filter, low-noise amplifier, and power management module. The chip's sleep current is less 5 μ A, making it suitable for battery-powered wearable electronic devices), **MPU6050 IMU** (a low-cost, low-power 6-axis motion tracking chip that integrates a 3-axis accelerometer, 3-axis gyroscope and a Digital Motion Processor (DMP) into a tiny 4 mm $\times$ 4 mm package), **NEO6 GPS** module (A u-blox module that was used to get the positioning information. This module only supports GPS, and its compact architecture, memory, and power options make NEO-6 modules excellent for battery-powered mobile devices with very strict cost and space constraints. It also comes with a small battery for hot-start and built-in EEPROM to save configuration settings when turned off), and **RAK3172 LoRaWAN** module in a tiny 40 mm diameter board and configured to operate at 868 MHz. It's an Ultra-low-power consumption module with an active power consumption of less than 6 μ A/ MHz and 2.7 μ A in sleep mode. The PCB also supports lithium-ion battery charging from the onboard USB connector for programming the microcontroller.

We conducted our experiments in an open ground (Makerere University rugby ground) to simulate rural resource-constrained environments with scattered, inadequate semi-structured buildings and less interference with GNSS signals.

During data collection, the Volunteer mounted the embedded system in the two mentioned mounting points and walked in the reference paths. This was done to test if the device could track the number of steps made, distance traveled, and user position accurately, regardless of the device's mounting position. Our test set a radius of 20 m as the required geofence radius (safe zone), outside which the GNSS module would be turned on and coordinates retrieved. Note that the safe

zone radius is not fixed but is determined based on the user's safe surroundings. Only the MPU6050 IMU sensor was used for tracking inside the geofence radius.



(a) PCB. (b) Current reading from multi-meter.

Fig. 7. Designed embedded system.

The system printed out logs over a serial monitoring application; an Arduino serial monitor was used, after which the data was saved as a log of the experiment for analysis purposes. The logged data included current, relative location, number of steps made, heading of the person, and distance moved by the person. If the person has moved out of the specified geofence radius, the logs then include the status of the GPS and the current coordinates from it.

To assess the power consumption, current readings were taken using a multimeter in series with the system and the battery. Measurements showed that with the GPS off, an average of 55mA was drawn from the battery and 144mA when the GPS was turned on, as shown in Fig. 7b.

4.3 Results and Discussion

To evaluate the energy performance of our proposed GNSS activation method, five experiments were conducted. Three were conducted to evaluate the GNSS activation, and two to assess the effect of position performance. The data was collected as explained in 4.2. The following represents the results and analysis of the conducted experiments.

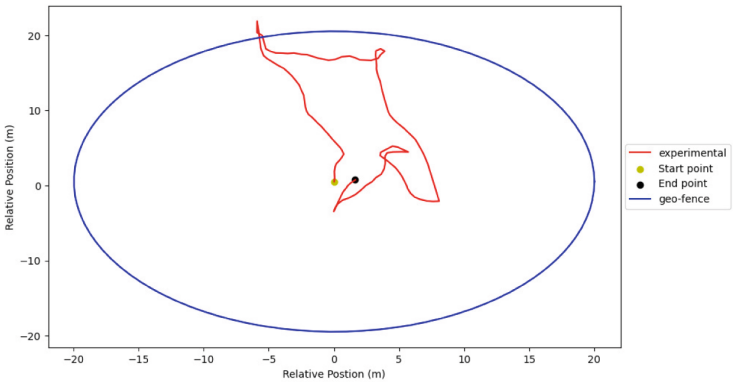
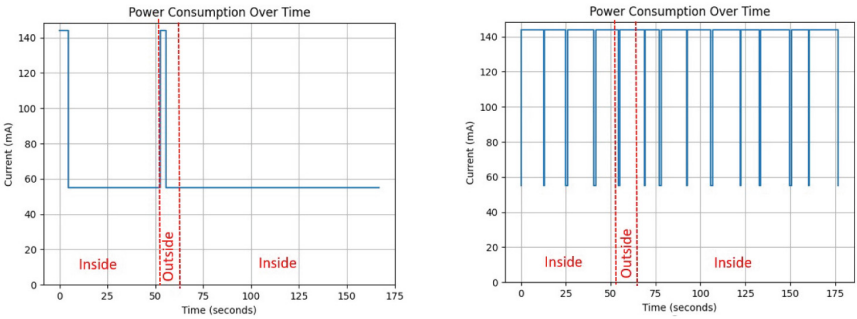


Fig. 8. Shows the path taken by the volunteer during the experiment one.



(a) Power consumption of the proposed method.

(b) Power consumption of the activity-based method.

Fig. 9. GPS activation outside the geofence

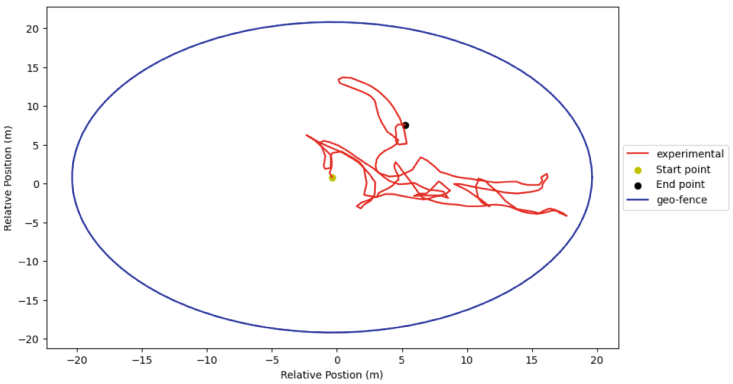
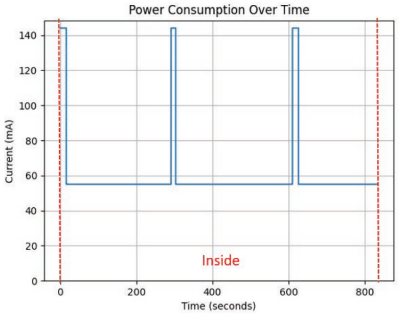


Fig. 10. Shows the user movements and position for the 14 min walk in experiment two.

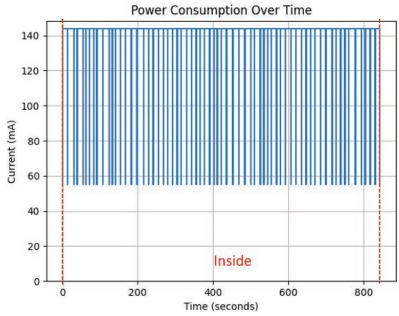
(i). Experiment one:
Figure 9 represents the analysis of data collected in experiment one for a user walking continuously inside, outside, and back inside the safe zone, as illustrated in Fig. 8. Figure 9a shows that the GPS was activated only once when the user briefly stepped out of the safe zone(section marked outside on the graph) and remained turned off for the entire period the user was walking in the safe zone (section marked inside). This demonstrates that our system turns on the GPS only when the user walks out of the geofence, maximizing the device's battery life. Contrary, Fig. 9b shows GPS continues to be turned on while the user moves, regardless of his/her position for the activity-based method.

(ii). Experiment two:
Figure 11 represents the analysis of data collected in experiment two, where the user walked continuously inside the safe zone, as illustrated in Fig. 10. Figure 11a shows that the GPS was activated only two times each after approximately 5 min for 14-minute continuous walks within the safe zone. As already explained in 4.1, we used 5 min instead of 30 min to simplify the experiment. This periodic GPS activation is done to avoid a cold start of the receiver, as already explained in 3.2. Because of this periodic activation, GPS took an average of less than 14 s to get a position fix in most cases, hence less power consumption. Contrary, Fig. 11b shows that the GPS continues to be turned on while the user moves, even though the user is inside the safe zone, hence unnecessary power consumption.

(iii). Experiment three:
Figure 13 represents the analysis of data collected in experiment three, where the user walked and stopped inside, outside, and back inside the safe zone, as represented in Fig. 12. This experiment demonstrates that in



(a) Our method periodically activates the GPS if the user stays inside the safe zone to avoid a cold start



(b) GPS continues to be turned on while the user moves in the safe zone for the activity-based method.

Fig. 11. Periodic GPS activation within the geofence

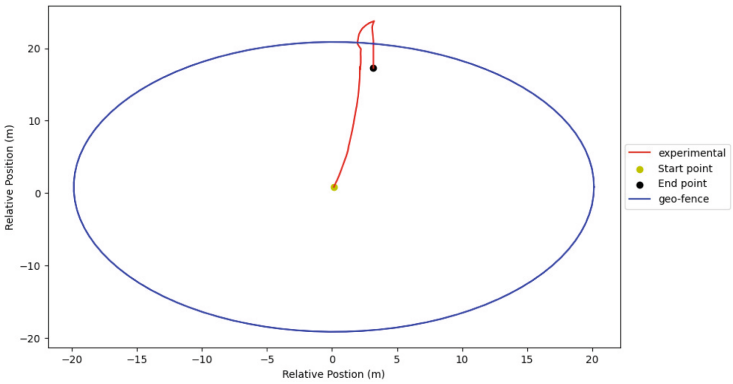
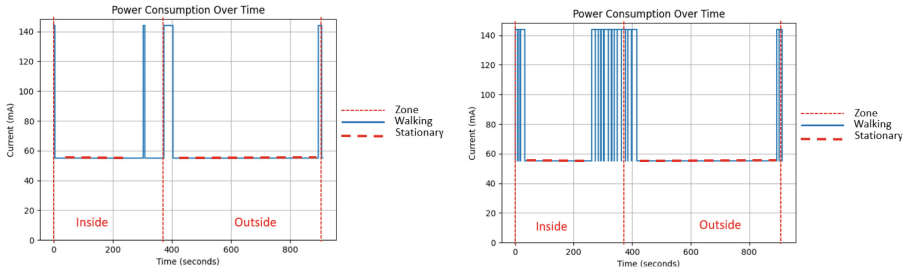


Fig. 12. Shows the path taken by the volunteer during the experiment two.



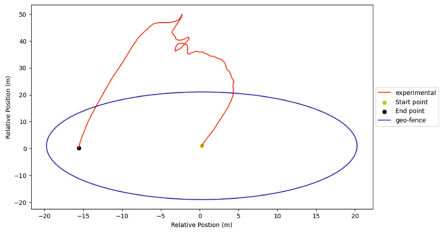
(a) Shows that GPS is turned off whenever the user is stationary even though outside the safe zone, as illustrated in 13.

(b) Shows that GPS is turned on and off for all times apart from when the user is stationary

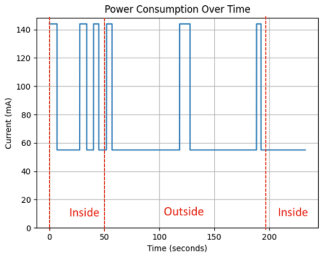
Fig. 13. GPS activation outside the geofence

our method, the GPS is turned off whenever the user is stationary until a step is detected again when the user is outside the geofence, as shown in Fig. 13a. Power consumption for both systems is comparable when the user is stationary regardless of user position but very high for the activity-based system when the user moves even though inside the geofence, as illustrated in Fig. 13b.

(iv). Experiment four:
Figure 14 shows the user positions estimated using pitch-based PDR implementation mounted on the chest. In this experiment, the user walked straight inside, outside, and back inside, as illustrated in Fig. 14a. This was done to demonstrate the effect of step detection and length estimation performance on the GPS activation method. Figure 14b shows several unnecessary GPS activations inside the safe zone resulting from wrong PDR positioning because of the mounting point used. Contrary, the same



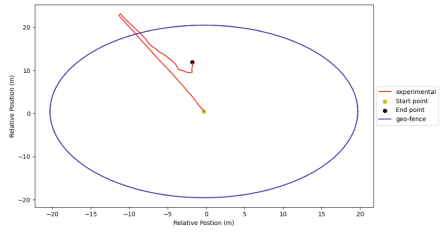
(a) Shows user movements and position using the Pitch-based PDR implementation with the device on the chest.



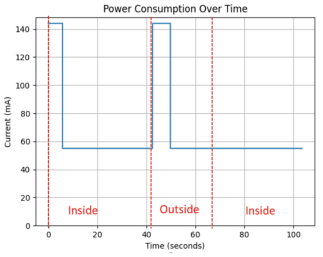
(b) Unnecessary GPS activations resulting from poor position performance.

Fig. 14. User position and power consumption using Pitch-based PDR implementation with the device on the chest

PDR implementation has a better positioning and even better power optimization when a different mounting point is used, as demonstrated in Fig. 15. Similarly, better positioning and power optimization are achieved as illustrated in Fig. 16 where acceleration-based PDR implementation is mounted on the chest, but contrary for the same implementation when the device is carried in the hands as shown in 17 From analysis it is clear the pitch-based PDR implementation works best with the body's swinging parts, but the acceleration-based method does not.



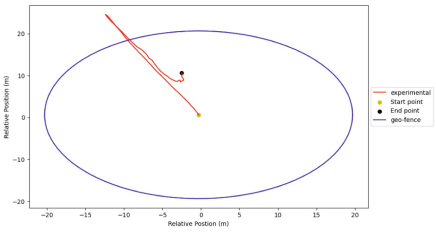
(a) Shows user movements and position using the Pitch-based PDR implementation with the device in hand.



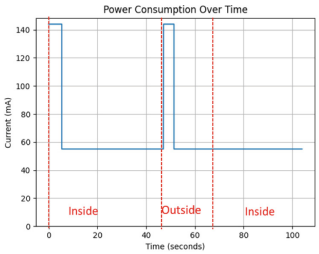
(b) GPS activations resulting from accurate position performance.

Fig. 15. User position and power consumption using Pitch-based PDR implementation with the device in hand

(v). Experiment five:
Figure 18 shows the effect of heading estimation on the user's final position estimation and power consumption. The acceleration-based PDR implementation mounted on the chest was used in this experiment, and the user walked in a zigzag from the center of the geofence to the outside and

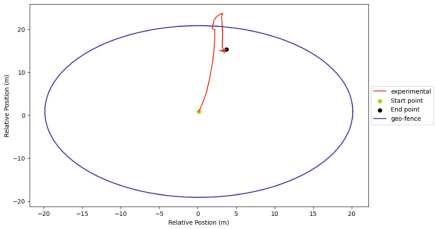


(a) Shows user movements and position using the Acceleration-based PDR implementation with the device on the chest.

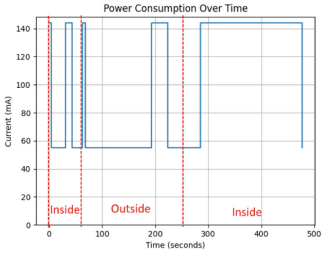


(b) GPS activations and power consumption resulting from accurate position performance.

Fig. 16. Shows the user position using Acceleration-based PDR implementation with the device on the chest

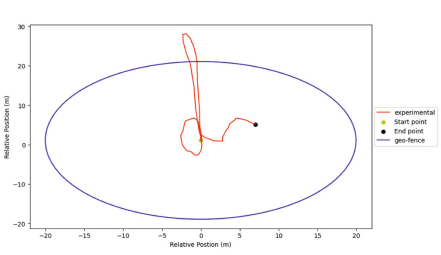


(a) Shows user movements and position using the acceleration-based PDR implementation with the device in hand.

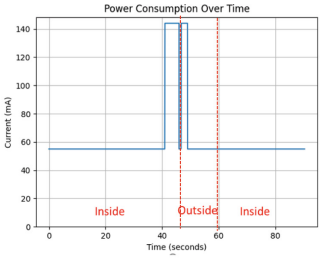


(b) Unnecessary GPS activations resulting from poor position performance.

Fig. 17. Shows the user position using Acceleration-based step detection method for different mounting points



(a) Shows the effect of heading estimation on the final user position.



(b) shows the GPS was activated when the user was still in the safe zone.

Fig. 18. Shows the effect of heading estimation on the user's final position estimation and power consumption

back to the center, but because of the effect of heading estimation, the user position is off by more than 3 m. Figure 18b shows the GPS was activated when the user was still in the safe zone, hence unnecessary power consumption.

5 Conclusion

This paper presents a position-based GNSS activation method using a PDR system as the main input source. The aim is to design an activation method better than usual activity-based methods for pedestrians (commonly acceleration-based) and cheaper than common position-based methods requiring beacons. From all experiments and results, our proposed method shows better power optimization (by more than 50%) compared to the activity-based method, as demonstrated using the current draw.

The proposed GNSS activation and position update rate method depends on the user position estimated by the PDR system. So, the accuracy with which the PDR system determines the position is key to the effectiveness of our proposed method. To assess this, two different PDR systems were implemented and evaluated, i.e., acceleration-based and pitch-based PDR implementation. Two mounting positions were used to collect data, i.e., on the chest and in the hands. The experimental evaluation confirmed that the PDR's estimated position directly affects the proposed GNSS activation method, reducing its performance.

The approach defined in this study can be applied to children and pet monitoring in open-space scenarios such as homes, sports fields, and recreation parks.

In future work, we plan to improve the power measurement by adding a fuel gauge to get the exact power consumption reading. We also plan to explore improved GNSS activation methods by using machine learning to distinguish motion mode by users at home vs. moving to a different place. This way, we could avoid the error accumulation limitation of the PDR strategy.

References

1. Anacleto, R., Figueiredo, L., Almeida, A., Novais, P.: Person localization using sensor information fusion. In: Ramos, C., Novais, P., Nihan, C.E., Corchado Rodríguez, J.M. (eds.) Ambient Intelligence - Software and Applications. AISC, vol. 291, pp. 53–61. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07596-9_6
2. Heidtmann, B.: Low-power GNSS for tracking applications. White paper, U-blox (2021)
3. Binette, J., Vasold, K.: 2018 home and community preferences: a national survey of adults age 18-plus. Technical Report, AARP Research, August 2018. https://doi.org/10.26419/res.00231.001
4. Culberson, J.W., Kopel, J., Sehar, U., Reddy, P.H.: Urgent needs of caregiving in ageing populations with alzheimer's disease and other chronic conditions: support our loved ones. Ageing Res. Rev. **90**, 102001 (2023)
5. Dai, Z., Podd, F.J.: A power-efficient BLE augmented GNSS approach to site-specific navigation. In: 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), pp. 1305–1310. IEEE (2020)

6. van Diggelen, F.: Assisted GNSS. In: Position, Navigation, and Timing Technologies in the 21st Century, pp. 419–444. Wiley (2020). https://doi.org/10.1002/9781119458449.ch17
7. Dinh, T.N., La The, V.: A novel design of low power consumption GPS positioning solution based on snapshot technique. In: 2017 International Conference on Advanced Technologies for Communications (ATC), pp. 285–290. IEEE (2017)
8. of Economic, U.N.D., Affairs, S.: World Population Ageing 2020: Highlights: Living Arrangements of Older Persons. UN (2021)
9. Fernandes, C.D., et al.: Hybrid indoor and outdoor localization for elderly care applications with lorawan. In: 2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA), pp. 1–6. IEEE (2020)
10. Huang, C.M., Lee, C.H., Chen, W.S.: A power efficient pedestrian touring scheme based on sensor-assisted positioning and prioritized caching for smart mobile devices. In: 2013 21st International Conference on Software, Telecommunications and Computer Networks-(SoftCOM 2013), pp. 1–5. IEEE (2013)
11. Jacob Rodrigues, M., Postolache, O., Cercas, F.: Physiological and behavior monitoring systems for smart healthcare environments: a review. Sensors **20**(8), 2186 (2020)
12. Janssen, T., Koppert, A., Berkvens, R., Weyn, M.: A survey on IoT positioning leveraging LPWAN, GNSS and LEO-PNT. IEEE Internet Things J. **10**, 11135–11159 (2023)
13. Junior, A.P., Díez, L.E., Bahillo, A., Eyobu, O.S.: Remote pedestrian localization systems for resource-constrained environments: a systematic review. IEEE Access **11**, 36865–36889 (2023)
14. Kim, P.: Kalman Filter for Beginners: with MATLAB Examples. CreateSpace, Charleston, SC (2011)
15. Kjærgaard, M.B., Langdal, J., Godsk, T., Toftkjær, T.: Entracked: energy-efficient robust position tracking for mobile devices. In: Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, pp. 221–234 (2009)
16. Lehtinen, M., Happonen, A., Ikonen, J.: Accuracy and time to first fix using consumer-grade GPS receivers. In: 2008 16th International Conference on Software, Telecommunications and Computer Networks, pp. 334–340. IEEE (2008)
17. Molteno, T.C.: Estimating position from millisecond samples of GPS signals (the "fastfix" algorithm). Sensors **20**(22), 6480 (2020)
18. Morton, Y.J., van Diggelen, F., Spilker Jr, J.J., Parkinson, B.W., Lo, S., Gao, G.: Position, navigation, and timing technologies in the 21st century: Integrated satellite navigation, sensor systems, and civil applications, vol. 1. John Wiley & Sons (2021)
19. Neubauer, N.A., Azad-Khaneghah, P., Miguel-Cruz, A., Liu, L.: What do we know about strategies to manage dementia-related wandering? a scoping review. Alzheimer's Dement. Diagn. Assess. Dis. Monit. **10**, 615–628 (2018)
20. Oshin, T.O., Poslad, S., Ma, A.: Improving the energy-efficiency of GPS based location sensing smartphone applications. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 1698–1705. IEEE (2012)
21. Paek, J., Kim, J., Govindan, R.: Energy-efficient rate-adaptive GPS-based positioning for smartphones. In: Proceedings of the 8th International Conference on Mobile systems, Applications, and Services. pp. 299–314 (2010)
22. Poulose, A., Eyobu, O.S., Han, D.S.: An indoor position-estimation algorithm using smartphone IMU sensor data. IEEE Access **7**, 11165–11177 (2019)

23. Wang, M., Qin, H., Jin, T.: Massive terminal positioning system with snapshot positioning technique. GPS Solutions **23**, 1–14 (2019)
24. Wang, S., et al.: Technology to support aging in place: older adults' perspectives. Healthcare **7**(2), 60 (2019). https://doi.org/10.3390/healthcare7020060
25. Weiss, V., Bologna, G., Cloix, S., Hasler, D., Pun, T.: Walking behavior change detector for a "smart" walker. Procedia Comput. Sci. **39**, 43–50 (2014)

Author Index

A
Adam, Shuaibu Musa II-368
Agate, Vincenzo I-445
Agti, Nadia II-231
Alizai, Muhammad Hamad II-43
Amano, Tatsuya II-280
Araujo, Aleteia II-319
Arrotta, Luca I-375
Asghar, Farhan II-43

B
Bachiega Jr., João II-319
Bader, Lennart I-464
Bahillo, Alfonso I-526
Bai, Hongtao II-105
Banerjee, Abhik I-156, II-319
Banerjee, Rochishnu II-28
Barendregt, Rosaline II-473
Beigl, Michael II-15
Ben Fredj, Khalil I-201
Bettini, Claudio I-375
Bhatti, Naveed Anwar II-43
Bie, Yuhang II-485
Bodenhausen, Jörn I-333
Brunello, Andrea I-46
Bui, Van Vu II-368

C
Cai, Mengnan II-3
Carvalho, Leonardo R. II-319
Cenni, Daniele II-167
Chen, Hsiao-Yuan I-3
Chen, Niansheng I-297
Chen, Xin I-313
Chen, Zhi II-268
Cheng, Songlin I-297
Chibani, Abdelghani II-231
Chowdhury, Farida II-381
Christmas, Shaine I-175
Civitarese, Gabriele I-375
Costa, Breno II-319
Cui, Jiaxu II-105

D
De Paola, Alessandra I-445
Diallo, Alpha I-26
Díez, Luis Enrique I-526
Dolui, Koustabh I-97
Don, Jacob II-341
Du, Cuifeng I-489, II-268
Duan, Zhifei I-489

E
Edirisinghe, Sampath II-86
Eyobu, Odongo Steven I-526

F
Fan, Guangyu I-297
Fatima, Tehreem II-43
Fujii, Hisanori II-204

G
Gan, Qingqing I-429
Gani, Md Osman II-28
Garbinato, Benoît I-26
Georgakopoulos, Dimitrios I-119
Grafflage, Kolja I-333
Gu, Tao II-244
Guan, Quanlong I-489, II-268
Guo, Xiaolei II-511
Gupta, Neelima II-294

H
Han, Qi II-167, II-415
Hara, Takahiro II-219
Harland, James II-538
Hasan, Khalad II-381
Hasan, Mehedi II-428
Havinga, Paul I-201
Heijenk, Geert I-201
Heitmann, Martin II-473
Henze, Martin I-333
Hmissi, Fatma II-393
Hoo, Vichen II-268

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved

A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 547–550, 2024.
https://doi.org/10.1007/978-3-031-63989-0

Hu, Chunyu I-392
Huang, Guang-Li I-156
Huang, Xiujie I-489, II-268
Huang, Yanqiu I-201
Huang, Yiran II-15
Hughes, Danny I-97, II-368

I
Inan, Muhammad Sakib Khan I-119
Islam, A. B. M. Alim Al I-408
Islam, Md Fourkanul II-28
Islam, Nafisa I-408
Ivoghlian, Ameer I-136

J
Jabbari, Amir II-125
Jayaraman, Prem Prakash I-119, I-156, II-319
Jiang, Landu II-244
Jiao, Libo I-313
Jin, Xuhao II-192
Julien, Christine I-3
Junior, Asiimwe Paddy I-526
Jurdak, Raja II-125

K
Kamal, Uday I-408
Kanhere, Salil S. I-464, II-353
Karmakar, Sushanta II-63
Katsuma, Ryo II-204
Kazar, Okba II-231
King, Evan I-3
Krishna, Aneesh II-341
Kummerow, Arne II-503

L
Lee, Kevin I-175
Leung, Victor C. M. I-257
Li, Jinbao II-256
Li, Jing II-511
Li, Shanping II-445
Li, Wang I-313
Li, Zhefu I-489, II-268
Liao, Kewen I-119
Ling, Tianheng II-529
Liu, Changjiang I-489
Liu, Guozheng I-223
Liu, Hao II-179
Liu, Hong I-392
Liu, Meiling II-485
Liu, Mengyao II-368
Liu, Shaoshan I-505
Liu, Sheng I-239
Liu, Wenwen I-257
Liu, Xiaoguang I-257
Liu, Zhaoyi I-97
Liu, Zhen I-429
Lo Re, Giuseppe I-445
Lu, Kezhong II-244
Lunardi, Roben C. I-476
Luo, Cheng II-244
Lv, Shuanghuan II-268
Lyu, Lei I-392

M
Mackenzie, Brendan J. II-368
Madria, Sanjay II-294
Mahmud, Redowan II-341
Malik, Sidra II-125
Manchanda, Ashish I-156
Mazumder, Aritra II-381
Michiels, Sam I-97, II-368
Mistry, Avinaba I-68
Mistry, Sajib II-341
Montanari, Angelo I-46
Morshchavka, Sergii I-97
Mubassira, Taseen II-428
Muthu Kumara Swamy, Shneka II-415
Muzammel, Chowdhury Shahriar II-538

N
Namvar, Anahita II-353
Nguyen, Thomas II-28
Ning, Wangzhong I-313
Noyce, Hanna M. II-28

O
Olejniczak, Emily M. II-28
Ouni, Sofiane II-393

P
Pallamreddy, Akhil Reddy I-201
Paul, Souradyuti II-63
Peng, Bitao I-429
Pennekamp, Jan I-464
Phuong, Huynh Nguyen Bao II-368

Q
Qian, Chao II-529

R
Rabbi, Forhad II-381
Rademacher, Michael I-333
Rahaman, Masfiqur I-408
Rajashekar, Kolichala II-63
Ramachandran, Gowri II-125
Ranaweera, Chathurika II-86
Rao, Lei I-297
Rasheed, Raihan I-408
Ray, Suprio I-68
Raychoudhury, Vaskar II-28
Riedel, Till II-15
Rizk, Hamada II-280
Rosa, Nelson Souto II-368
Rötzel, Sebastian I-333

S
Sabri, Lyazid II-231
Saccomanno, Nicola I-46
Sakib, Md. Nazmul Hasan I-408
Salcic, Zoran I-136
Salim, Saiful Islam I-408
Schaffeld, Matthias II-520
Schiele, Gregor II-529
Schneider, Jean-Guy I-175
Seahra, Sanjeev I-68
Sharmin, Sadia II-428
Shen, Haifeng I-119
Shen, Xinling II-3
Shen, Zhonghui II-256
Shi, Huiling I-223
Siddiqui, Junaid Haroon II-43
Sidhanta, Subhajit II-63
Silveira, Rodrigo W. I-476
Smith, Roger O. II-28
Soha, Sabiha Tahsin II-381
Song, Xiaoyong I-297
Sorgatz, Christian I-333
Spichkova, Maria II-461, II-538
Spiß, Maria I-464
Su, Tongtong I-257
Sun, Bingyi II-105

T
Tan, Lizhuang I-223
Tang, Bo II-256
Tang, Chang I-223
Tang, Jie I-505
Tang, Ming Jian I-119
Thangarajan, Ashok Samraj I-97, II-368
Thapa, Chandra II-353
Thevaraj, Emildeon I-464
Torres-Sospedra, Joaquín I-46

V
Virga, Antonio I-445
Vogt, Thomas I-333

W
Wadhwa, Manisha II-294
Wang, Changji I-429
Wang, Gang I-257
Wang, Kevin I-Kai I-136, I-355
Wang, Nan II-256
Wang, Qing II-179
Wang, Ruoyu I-429
Wang, Yaoyidi I-297
Wang, Yixiao I-392
Wang, Zehua I-257
Wasson, Barbara II-473
Weerasinghe, Shakthi I-156
Wehrle, Klaus I-464
Weis, Torben II-503, II-520
Wu, Xiaofeng I-489, II-268

X
Xie, Rong II-145

Y
Yamada, Shota II-280
Yamaguchi, Hirozumi II-280
Yang, Bo II-445
Yang, Dingyu I-297
Yang, Jie II-179
Yang, Meihong I-223
Yao, Lina II-219
Ye, Hanting II-179
Ye, Xiaozhou I-355
You, Linlin I-239
Yu, Bo I-505
Yu, Zhaoyang I-257
Yuan, Lin I-392

Z
Zaslavsky, Arkady I-156
Zhang, Dian II-244
Zhang, Guanghui I-489
Zhang, Shuai I-281

Zhang, Shunliang II-511
Zhang, Tian I-489
Zhang, Wei I-223
Zhang, Xiangxie II-179
Zhang, Xichen I-68
Zhang, Yan II-145
Zhang, Yihong II-219
Zhang, Yonggang II-105
Zhang, Zhongmei I-281
Zhao, Haibin II-15
Zhao, Sinong I-257
Zheng, Sunjian I-505
Zhong, Yuansheng I-489
Zhou, Jiyun II-485
Zhou, Yexu II-15
Zhou, Yuren I-239
Zhou, Yuyu II-268
Zhu, Wenwu I-313
Zhuang, Xiaotian II-268
Zorzo, Avelino F. I-476