RETSINA: Reproducibility and Experimentation Testbed for Signal-Strength Indoor Near Analysis

Anna Baskin*, Brian T. Nixon*, Panos K. Chrysanthis
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA, USA
afb39@pitt.edu, {nixon.b, panos}@cs.pitt.edu

Christos Laoudias
KIOS Research Center
University of Cyprus
Nicosia, Cyprus
laoudias.christos@ucy.cy

Constantinos Costa
Rinnoco Ltd
Limassol, Cyprus
costa.c@rinnoco.com

2023 13th International Conference on Indoor Positioning and Indoor Navigation (IPIN) | 979-8-3503-2011-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/IPIN57070.2023.10332500

***Abstract*—Reproducibility is a core component of any scientific discovery. A step towards reproducibility within the IPIN community is the contribution of this paper, our software-based testbed, called RETSINA (Reproducibility and Experimentation Testbed for Signal-strength Indoor Near Analysis). RETSINA enables the repeatability, reproducibility and comparison of approaches that use machine learning to detect proximity. We demonstrate RETSINA's functionality by repeating and extending the findings of a recent case study on Wi-Fi signal strength based contact tracing accuracy. Furthermore, we leverage RETSINA to experimentally compare the results for detecting close encounters produced by the original Wi-Fi signal strength readings study and our study using Bluetooth signal strength readings.**

***Index Terms*—Reproducibility, Repeatability, Contact Tracing, Proximity Detection, Indoor Localization, Machine Learning**

I. Introduction

Soon after the outbreak of the COVID-19 pandemic, Digital Contact Tracing (DCT) solutions were rapidly developed and deployed nation-wide to alleviate the cumbersome task of conventional contact tracing. The vast majority of DCT solutions leverage smartphone mobile apps that rely on Bluetooth scanning for power efficient and privacy-preserving proximity sensing [1]. Despite the evidence-based effectiveness of Bluetooth DCT apps [2], their behavior in complex indoor environments is not well studied yet, while the limited experimental setups are hard to replicate in order to reproduce the tracing accuracy results. Alternative technologies that are readily available on modern smartphones are already being explored to infer close contacts indoors in next generation DCT apps. For instance, uncertain positioning data were recently used for indoor contact tracing [3]; however, using location information as a proxy to determine the distance between users and detect proximity raises privacy concerns. To address this limitation, authors in [4] consider Wi-Fi signal strength fingerprints to detect proximity through Machine Learning (ML), thus removing the need to estimate user locations. While promising results are reported, they need to be validated by further studies and reproduced to compare against current solutions based on Bluetooth.

Research reproducibility is attracting increasing attention due to the need for more transparent, useful, and trustworthy scientific publications. Several computer science and engineering research communities and professional organizations, including ACM and IEEE, have identified the gap long ago and initiated targeted awareness raising activities [5]–[8]. Reproducibility is thereafter recognized as an essential scientific skill [9] and there is a limited number of publications addressing this issue across different areas, including Jupyter notebooks [10], ML platforms [11], signal processing [12], and computing research [13] among a few others. However, to achieve the ultimate goal of *fully reproducible* publications, it seems that improvements are still needed [14]. The situation is similar within the IPIN community including sparse related works, e.g., a 2015 survey of experimental evaluation in indoor localization research [15] and the offline evaluation of indoor positioning systems in the context of the IPIN 2020 indoor competition [16], as well as a few notable examples on reproducibility such as a reproducible comparison of clustering and optimization rules in Wi-Fi fingerprinting [17] and an effort to benchmark the methods of dynamic accuracy estimation of localization [18].

To the best of our knowledge, this work is the first attempt to introduce a testbed for reproducibility and experimentation that focuses jointly on i) digital contact tracing, rather than user localization and ii) indoor environments with far more challenging radio signal propagation conditions, compared to outdoor settings. To this end, our contribution is threefold.

- We offer to the community a unique software-based playground for research on indoor contact tracing, coined RETSINA (Reproducibility and Experimentation Testbed for Signal-strength based Indoor Near Analysis), which allows not only to assess the repeatability of scientific results reported in the literature, but importantly to extend those results through configurable experimentation.
- We leverage RETSINA for reproducing the results presented in the recent case study [4], which relies on Wi-Fi signal strength readings for detecting close encounters, in terms of contact tracing accuracy.
- We go beyond the reproducibility case study and present various extensions, including the provision of Bluetooth signal strength data, and improvements to the ML models attained by re-configuring the RETSINA testbed.

*These authors contributed equally.

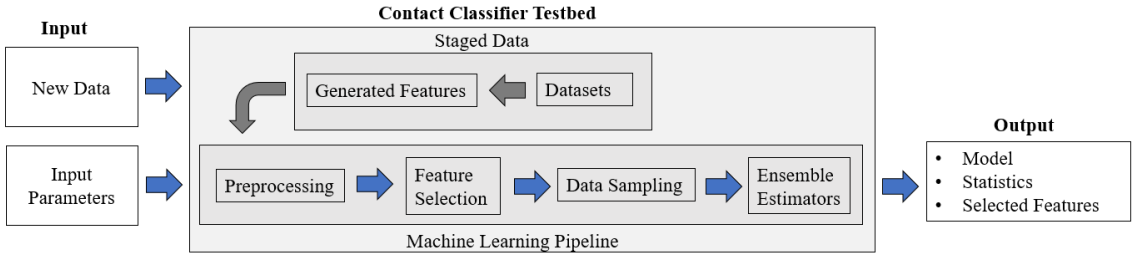


Fig. 1. RETSINA Contact Classifier Testbed.

The rest of the paper is structured as follows. Section II describes the RETSINA testbed and provides the details of the various system parameters pertaining to data selection and sampling, feature selection, ML estimators' configuration, and computational resources. We reproduce the results of the case study and discuss our findings in Section III. Next, RETSINA is used to perform extensive experimentation by re-configuring the system parameters and we report the improved accuracy results in Section IV. Finally, Section V, provides concluding remarks and directions for future work.

II. Testbed

Our RETSINA testbed[1] is a software experimental platform for evaluating proximity detection methods based on signal-strength readings through machine learning. Specifically, it allows the reproducibility of proposed classifiers that accept pairs of signal-strength fingerprints and a proximity distance (e.g., 2.5m apart for COVID-19 exposure based on CDC guidelines) and measures their contact accuracy.

```
How many datasets should the classifier be trained on? [all, one] all
Using all of the provided datasets as training data

What is the directory name for the testing dataset? (It is assumed that the directory exists within the datasets directory) 10-JUIndoorLoc
Using the 10-JUIndoorLoc dataset under the datasets/ directory.

How many features would you like extracted using pymRMR? 12
pyMRMR will extract 12 features

What is the balance of Close and Far Samples? Please enter the balance as Close:Far 40:60
The system currently supports the following classifiers:

        0. KNeighborsClassifier(3)

        1. DecisionTreeClassifier(max_depth=5)

        2. AdaBoostClassifier()

        3. MLPClassifier(alpha=1, max_iter=1000)

Please enter the number of the classifier you wish to include or -1 to finish choosing classifiers 1
Please enter the number of the classifier you wish to include or -1 to finish choosing classifiers 2
Please enter the number of the classifier you wish to include or -1 to finish choosing classifiers 3
Please enter the number of the classifier you wish to include or -1 to finish choosing classifiers -1
Please enter the number of processes you wish to use as an integer 20
Input Summary
Training with all datasets
Testing with the 10-JUIndoorLoc dataset
Selecting 12 features
Using a 40/60 split
Using the following classifiers: []
Using 20 processes

***Writing RETSINA updates to log files under /output_figs/*.log***
```

Fig. 2. The RETSINA interface for gathering configuration parameters.

A. Architecture

Our testbed pipeline consists of five components, as shown in Figure 1: *Generated Features, Preprocessing, Feature Selection, Data Sampling* and *Ensemble Estimator*. It takes as input fingerprint datasets for training/testing and configuration parameters as shown in Figure 2 and outputs a trained classifier and relevant statistics regarding the classifier's performance on each testing dataset (accuracy, "close" accuracy, "far" accuracy, balanced accuracy, precision, and recall). It also outputs the features selected through feature selection to be used by the classifier.

The testbed also provides staged datasets to be used as additional data for reproducibility and experimentation. These staged datasets for training/testing are from Miskolc [19], JUIndoorLoc [20], UJIIndoorLoc [21], and IPIN-Tutorial [21].

The input fingerprints data is expected to be of the following tabular form and stored in CSV files:

$$< \text{APs, lon, lat, FLOOR, BUILDINGID, SPACEID,}$$
$$\text{RELATIVEPOSITION, USERID, PHONEID, TIMESTAMP} >$$

where all access points (APs) appear before the longitude (lon) and latitude (lat). The relative position (RELATIVEPOSITION) is a binary attribute that has value one if the fingerprint was recorded inside the corresponding space and zero if captured outside (in front of the door in our case) of the space.

Each sample in the generated data represents a combination of two fingerprints from the original datasets. Features for the generated data can be placed into several categories shown in Table I as described by Hyfe et al. [4], with the addition of a new category G. *AP Detection Features* which is described in Section IV-D.

TABLE I
Feature Categories for Generated Data

Feature Category
A. AP Detection-Based Features
B. Basic RSSI Value-Based Features
C. Redpin Score-Based Features
D. Correlation-Based Features
E. Difference-Based Features
F. Device Heterogeneity Mitigation Features
G. AP Detection Features

Preprocessing on the generated data includes normalization and filtering samples to remove samples outside the designated range for distance. Next, features are selected using minimum Redundancy Maximum Relevance (mRMR) feature selection, as in Hyfe et al. [4]. The data is sampled to train/test according to the designated sampling balance of close/far. Finally, the testbed trains an attribute bagging classifier with multiple options for the base estimator.

[1]RETSINA is publicly available at https://doi.org/10.5281/zenodo.8143917.

B. Configuration Parameters

Tuning the testbed parameters will affect the performance of the classifier. There are four categories of configuration parameters:

1) **Data Selection and Sampling Parameters:**
   - *Training Datasets*: indicates which dataset/datasets to use in training.
   - *Testing Datasets*: indicates which dataset/datasets to use in testing.
   - *Number of samples*: indicates the total number of samples to consider from each dataset.
   - *Percentage Close*: indicates the percentage of the samples that should be in close proximity (to tune the close/far balance of the sampled data) for the training data.
   - *Excluded middle range*: designates a range of distances to exclude from the training/testing data (e.g., ignore samples which are 2-2.5m apart).
   - *Max range*: designate a maximum distance apart for the model to consider (e.g., ignore samples greater than 20m apart).
2) **Feature Selection Parameters:**
   - *Number of pymRMR features*: designates the number of features selected during feature selection.
   - *Add num_aps (boolean)*: if true, uses the features specifying the number of access points in each fingerprint, alongside the features generated through feature selection.
3) **Machine Learning Parameters:**
   - *Base Estimator*: specifies options for the base estimators (Nearest Neighbors, Decision Tree, Neural Network, and AdaBoost). These were chosen to represent diverse methodologies in machine learning.
   - *Bagging Tree Depth*: indicates the number of features per base estimator in the bagging classifier.
   - *Number of Bagging Tree Estimators*: indicates the number of base estimators in the bagging classifier.
4) **Control Parameters:**
   - *Number of processes*: indicates the number of processes to use for parallelization.

C. Implementation

RETSINA was implemented with Python 3.10 using the *scikit-learn* 0.22.1 library for generating the Bagging Classifier and the base estimators (e.g., Decision Tree, Adaboost, KNeighbors, and MLP) as discussed further in Section IV as well as for standard feature scaling during preprocessing steps.

We utilized versions 1.24.1 and 1.5.3 of the *numpy* and *pandas* Python packages for processing and evaluating all data used by RETSINA and performing class rebalancing through the pandas DataFrame sample method.

We used version 0.1.11 of the *pymRMR* Python package to select the top features with the "mutual information difference" (MID) selection method.

Lastly, to facilitate the parallel computation of experiments, data processing, and data generation, we utilized version 2023.5.0 of the dask Python package and Python's standard multiprocessing package. The *dask* Python package was used to convert datasets of generated features to Parquet files, allowing for parallel processing on input data, which performs significantly faster than the standard Pandas method for reading CSV files and processing data.

III. Reproducibility Case Study

As mentioned in Section I, our testbed was produced to get a better insight to the findings of Hyfe et al. [4]. In this section, we share these insights of reproducing the aforementioned findings. Hyfe et al. train and compare the results of a *generic classifier* and a series of *specialized classifiers*. Specifically, the generic classifier is trained on all available data, that is, trained using data aggregated from all datasets, while the specialized classifiers are trained on data grouped by the average number of access points in each dataset.

A. Generic Classifier

The first experiment for the reproducibility case study focuses on the generic classifier trained using data aggregated from all staged datasets. The parameters for the generic classifier which mimic the original paper's classifier training include: 53% "close" samples, decision tree base estimator, 8 pymRMR features, bagging tree depth of 3, 300 bagging tree estimators, 17,000 samples per dataset, an excluded middle range of 2.25-3.25 meters, a maximum range of 20 meters, and use all the available datasets to test, and all datasets but JUIndoorLoc to train.

Using these parameters, our testbed yielded higher balanced accuracies for each dataset from the generic classifier of Hyfe et al. [4], as displayed in Table II. Some changes that may have affected the replication of the data include using fewer datasets (we implemented only four of the six publicly available datasets, and had no access to data collected in-house by the original paper) and our decision to filter all samples which had no shared access points (as these could automatically be labeled as "Far"). Both of these changes could increase the accuracy of the generic classifier. Additionally, we were unclear as to the use of feature reduction in the generic classifiers, and so chose to use only 8 features selected by the mRMR algorithm to reduce runtime and overfitting. It is possible that these changes caused a higher balanced accuracy as we observed a higher balanced accuracy for each dataset in comparison to the original paper.

B. Specialized Classifiers

We had initially assumed that Hyfe et al. [4] trained and tested a specialized classifier for each collection of datasets, grouped by access point density. While they did divide the datasets into high, medium, and low density access point groups, rather than training a classifier on each group, they trained a classifier on one dataset of the group and tested it on another. They found that training a classifier on a single

TABLE II
GENERIC CLASSIFIER REPRODUCTION

Dataset Name	Original Balanced Accuracy	Testbed Balanced Accuracy
Miskolc [19]	59.11%	73.69%
JUIndoorLoc [20]	52.34%	55.43%
UJIndoorLoc [21]	61.06%	73.24%
IPIN 2016 Tutorial [21]	55.46%	65.00%
Average	**56.99%**	**66.84%**

dataset and testing the classifier on a different dataset with a similar number of access points yielded a higher balanced accuracy (average of 71% as opposed to 56.99%). This serves as a proof of concept that a trained classifier can yield good results on completely unfamiliar data, provided that the test data has a similar number of access points.

While much of the data to test this claim was not publicly available, we found that we could test this hypothesis by training a classifier on the Miskolc data (median 10 APs per fingerprint) and testing on the JUIndoorLoc data (median 15 APs per fingerprint). Both would be within what the original paper considered a "low AP density" of 10-15 mean access points per fingerprint. This resulted in a balanced accuracy of 55.46%, around the accuracy of the average balanced accuracy of the generic classifier, which does not support the original paper's hypothesis. This indicates that while grouping by access point density may improve performance, it is not a guarantee.

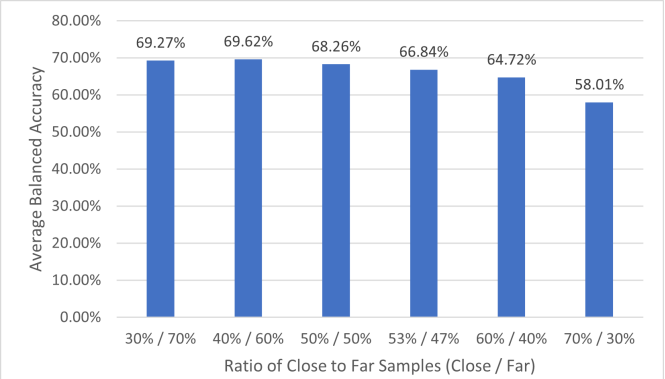
IV. EXPERIMENTAL EVALUATION CASE STUDY

In addition to reproducing the results of Hyfe et al. [4] in a publicly available testbed, this paper extended their results through experimentation. Beyond training/testing on new data, the testbed allows users to change parameters relating to sampling, preprocessing, and machine learning hyperparameters.

Specifically, first we manually tuned the bagging classifier estimator, the close/far balance, and the number of features of the original generic classifier (Section IV-A). Next, we observed whether additional features describing the number of access points in each fingerprint improve the accuracy of the classifier (Sections IV-B, IV-C and IV-D). Finally, we compared our resulting model statistics to the traditional Bluetooth-based contact classifiers to explore whether fingerprint-based comparison represents an improvement (Section IV-E).

A. Base Estimator Comparison

To begin, we trained a generic classifier for each possible base estimator. Beyond changing the base estimator of the bagging classifier, we used the original parameters of the generic classifier. However, Hyfe et al. [4] chose to drop the cases where fingerprints are collected 2.25-3.25 meters and greater than 20 meters away from each other. We did not include a middle range or maximum distance to be dropped for the IPIN 2016 Tutorial's evaluation dataset, because this resulted in the removal of all samples labeled as "Far," which would produce a dataset where the target label is always "Close." Table III shows that the Decision Tree performs best as the base estimator for the bagging tree classifier, with the highest balanced accuracy averaged across all test data.



Close/Far Split	Average Precision	Average Recall	Average FPR	Average FNR
30% / 70%	0.9082	0.7041	0.5234	0.2958
40% / 60%	**0.9161**	**0.6476**	**0.2552**	**0.3524**
50% / 50%	0.9236	0.6385	0.1235	0.5113
53% / 47%	0.9239	0.4451	0.2905	0.5549
60% / 40%	0.9237	0.3581	0.0638	0.6419
70% / 30%	0.9317	0.1826	0.0224	0.8174

Fig. 3. (top) A spectrum of close/far balances measuring the average balanced accuracy from a generic Decision Tree estimator, (bottom) The average precision, average recall, average false positive rate (FPR), and average false negative rate (FNR) of the close/far balances from the generic Decision Tree.

TABLE III
BASE ESTIMATOR COMPARISON

Base Estimator	Balanced Accuracy
Nearest Neighbors	66.52%
Decision Tree	**66.84%**
Neural Network	66.65%
AdaBoost	63.47%

B. Close/Far Tuning

Next, we trained a generic classifier using a Decision Tree as the base estimator for a spectrum of close/far balances, ranging from 30% close to 70% close. The results, shown in Figure 3, indicate that a 40% close, 60% far split provides the best results in terms of balanced accuracy.

An interesting observation from this experiment is that the close/far splits with at least 50% close performed with a lower average balanced accuracy than the splits where less than half of the samples were close. One potential reason for this trend is the class imbalance present in all of the testing data which included: a 5% / 95% split for the Miskolc dataset, a 38% / 62% split for the JUI dataset, a 4% / 96% split for the UJILoc dataset, and a 2% / 98% split for the IPIN dataset.

As shown in Figure 3 (bottom) close/far balances with at least 50% of the samples labeled "Close" had a marginally

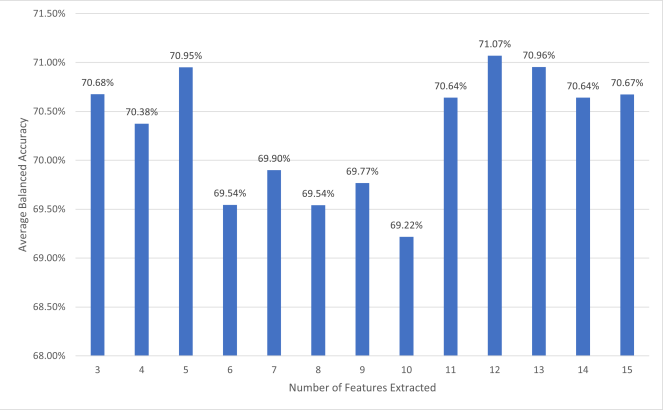


Fig. 4. The results of measuring the average balanced accuracy from a generic Decision Tree estimator using 3 to 15 features.

higher average precision but significantly lower average recalls that rapidly declined as the percentage of far labels in the split decreased. Furthermore, the figure shows a tradeoff as the percentage of close labels increases, the average false positive rate (FPR) decreases while the average false negative rate (FNR) increases.

C. Number of Features Tuning

We trained a generic classifier using a Decision Tree as a base estimator and 40% close using a variable number of features. We had the mRMR algorithm select 3-15 features during feature selection. The 15 most relevant features from the generic classifier's aggregated training dataset, as identified by the pymRMR package, are: (1) Pearson coefficient of the shared AP pair ratio vectors (Double-fingerprint least squares), (2) Non-shared APs count, (3) Percent of APs shared within 15dB (No transformation), (4) Shared AP pair difference vector standard deviation (Single-fingerprint 50% least squares), (5) Top APs shared within 13dB (No transformation), (6) Same Device Model, (7) Pearson Coefficient of the Rank vector (No transformation), (8) Jaccard ratio, (9) Percent of APs shared within 9dB (No transformation), (10) Shared AP pair difference vector standard deviation (No transformation), (11) Shared top 4 APs (No transformation), (12) Pearson coefficient of shared AP vector (No transformation), (13) Shared top 6 APs (Single-fingerprint 50% least squares), (14) Percent of APs shared within 1dB (Single-fingerprint least squares), and (15) Percent of APs shared within 13dB (No transformation).

The results, shown in Figure 4 indicate selecting 12 features yields the highest balanced accuracy. In the cases of features 13, 14, and 15, it's possible that the average balanced accuracy was lower (despite the increase in features) due to overlapping with similar features that were already chosen by mRMR and introducing bias towards the same correlated feature. For instance, feature 13 is a stricter variant of feature 11 as it checks if the fingerprints share the same top 6 APs rather than the top 4 APs. Features 14 and 15 have narrower RSSI ranges of 1dB and 13dB compared to feature 3 which finds the percentage of APs that are shared within a range of 15dB.

D. Adding Number of Access Points

Hyfe et al. [4] demonstrated that grouping the data by average number of access points significantly increases the accuracy of the models. Even if we found that this is not always the case, it is still evident that grouping by access point density can significantly improve performance.

However, grouping by access point density disregards the main benefit of fingerprint comparison. Fingerprint comparison is useful because it can estimate distance without any required context about the location. Using the average number of access points in the area to determine which classifier to use would require context about the area, and so the classifiers could not be deployed in new, unfamiliar locations.

TABLE IV
GENERIC CLASSIFIER WITH NUMBER OF ACCESS POINTS COMPARISON

Dataset Name	Without APs Balanced Accuracy	With APs Balanced Accuracy
Miskolc	77.47%	76.60%
JUIndoorLoc	56.16%	57.54%
UJIndoorLoc	81.68%	81.74%
IPIN 2016 Tutorial	68.98%	68.65%
Average	**71.07%**	**71.13%**

Therefore, we decided to generate two additional features describing the number of access points in each fingerprint, in the hopes that their addition would improve the accuracy of the generic classifier. In this experiment, we found that including the number of access points of each fingerprint to the above generic classifier, in addition to the 12 selected features, yielded 71.13% balanced accuracy as shown in Table IV, which is a mild improvement over the above classifier. In addition, this new generated feature could in future works be used to cluster the fingerprint comparisons before training specialized classifiers, which would once again allow the generic classifier to be location unaware.

E. Bluetooth Comparison

Finally, we compared our best model to classifiers generated using only Bluetooth signal strength to predict close or far. In order to generate our Bluetooth models, we used an open-source Bluetooth range-estimation dataset deliberately collected in a complex office environment by Pascacio et. al [22]. We split the dataset into training and testing, using 30% of the original data as testing data and undersampling the training data to create a balanced class. We trained classifiers using the same machine learning algorithms that formed the base estimators for the generic classifier: Nearest Neighbors, Decision Tree, Neural Network, and AdaBoost, with the addition of Naive Bayes, QDA, Linear SVM, and RBF SVM. The Decision Tree had the highest balanced accuracy. Next, we manually tuned the depth and minimum number of samples per leaf, finding that the best parameters were a max depth of 6 and a minimum sample value of 2. In Table V we compare the tuned Decision Tree Bluetooth classifier to our best performing testbed classifier.

TABLE V
BLUETOOTH COMPARISON

Metrics	Contact Classifier Model	Bluetooth Model
Balanced Accuracy	71.13%	60.33%
Accuracy	63.74%	54.41%
Precision	92.93%	11.71%
Recall	62.32%	67.45%
F1-score	74.38%	19.96%
True Positive Percent	62.32%	67.45%
False Positive Percent	21.04%	46.79%
True Negative Percent	78.97%	53.21%
False Negative Percent	37.68%	32.55%

V. CONCLUSION

Reproducibility is an essential component that can help facilitate further discoveries within the scientific community such as that of IPIN. In this paper we proposed our software-based testbed, referred to as RETSINA, to enable the repeatability, reproducibility and comparison of approaches that utilize machine learning techniques for the problem of proximity detection. We demonstrated how RETSINA could be leveraged to repeat the results presented in the recent case study [4] and presented various extensions, including the the provision of Bluetooth signal strength data and tuning of machine learning, feature selection, and sampling parameters achieved by reconfiguring the RETSINA testbed. As part of our future work, we hope we will be able to repeat our reproducibility case study using the two public datasets that were not included as part of our experiments as well as the private data that was examined by Hyfe et al.

In this paper, we have only shown the potential of our RETSINA testbed and its capability to repeat and extend the findings of the recent case study on Wi-Fi signal strength based contact tracing by Hyfe et al. Additional extensions could utilize clustering algorithms to cluster the fingerprint comparisons based on the new *AP Detection Features* category before training specialized classifiers. This clustering technique would allow classifiers to benefit from training on the number of APs in each fingerprint as shown in Section IV-D while remaining unaware of the recorded location, which would allow the classifiers to be deployed in new, unfamiliar locations. Further experimentation could involve different machine learning, feature selection, and data sampling algorithms. Finally, Wi-Fi signal strength readings could be combined with Bluetooth signal strength readings to improve overall balanced accuracy, though this requires the development of a new hybrid system that balances when to utilize classifiers trained on Bluetooth or Wi-Fi signal strength readings.

ACKNOWLEDGMENTS

This work was partially supported by NSF SES-2017614 and NIH R01-HL-159805 grants and reflects only the authors' opinions. The work was also supported by the European Union's Horizon 2020 research and innovation Programme under grant agreement No 739551 (KIOS CoE) and from the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

REFERENCES

[1] C. Laoudias, S. Meyer, P. Isaia, T. Windisch, J. Benzler, and M. Lenkeit, "Mobile applications for privacy-preserving digital contact tracing," in *23rd IEEE International Conference on Mobile Data Management (MDM)*, 2022, pp. 1–4.

[2] C. Wymant *et al.*, "The epidemiological impact of the nhs covid-19 app," *Nature*, vol. 594, no. 7863, pp. 408–412, 2021.

[3] T. Liu, H. Li, H. Lu, M. A. Cheema, and H. K.-H. Chan, "Contact tracing over uncertain indoor positioning data," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2023.

[4] Z. Van Hyfte and A. Zakhor, "Immediate proximity detection using wi-fi–enabled smartphones," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2021, pp. 1–8.

[5] J. Arguello, M. Crane, F. Diaz, J. Lin, and A. Trotman, "Report on the SIGIR workshop on reproducibility, inexplicability, and generalizability of results (RIGOR)," *SIGIR Forum*, vol. 49, no. 2, pp. 107–116, 2015.

[6] B. Baillieul, John *et al.*, "The first ieee workshop on the future of research curation and research reproducibility," 2017. [Online]. Available: https://open.bu.edu/handle/2144/39028

[7] B. R. Childers and P. K. Chrysanthis, "Artifact evaluation: Is it a real incentive?" in *13th IEEE International Conference on e-Science*, 2017, pp. 488–489.

[8] ——, "Artifact evaluation: FAD or real news?" in *34th IEEE International Conference on Data Engineering, ICDE*, 2018, pp. 1664–1665.

[9] W. Mauerer, S. Klessinger, and S. Scherzinger, "Beyond the badge: Reproducibility engineering as a lifetime skill," in *IEEE/ACM 4th International Workshop on Software Engineering Education for the Next Generation (SEENG)*, 2022, pp. 1–4.

[10] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, "A large-scale study about quality and reproducibility of jupyter notebooks," in *IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 507–517.

[11] R. Isdahl and O. E. Gundersen, "Out-of-the-box reproducibility: A survey of machine learning platforms," in *15th International Conference on eScience (eScience)*, 2019, pp. 86–95.

[12] T. Adali, R. C. Guido, T. K. Ho, K.-R. Müller, and S. Strother, "Interpretability, reproducibility, and replicability [from the guest editors]," *IEEE Signal Processing Magazine*, vol. 39, no. 4, pp. 5–7, 2022.

[13] W. Raghupathi, V. Raghupathi, and J. Ren, "Reproducibility in computing research: An empirical study," *IEEE Access*, vol. 10, pp. 29 207–29 223, 2022.

[14] J. Goodrich, "Study shows ensuring reproducibility in research is needed – the ieee computer society suggests improvements," *IEEE Spectrum*, 2021.

[15] S. Adler, S. Schmitt, K. Wolter, and M. Kyas, "A survey of experimental evaluation in indoor localization research," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2015, pp. 1–10.

[16] F. Potorti and et al., "Off-line evaluation of indoor positioning systems in different scenarios: The experiences from ipin competition," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 5011–5054, 2022.

[17] J. Torres-Sospedra, P. Richter, A. Moreira, G. M. Mendoza-Silva, E. S. Lohan, S. Trilles, M. Matey-Sanz, and J. Huerta, "A comprehensive and reproducible comparison of clustering and optimization rules in wi-fi fingerprinting," *IEEE Transactions on Mobile Computing*, vol. 21, no. 3, pp. 769–782, 2022.

[18] G. G. Anagnostopoulos and A. Kalousis, "Can i trust this location estimate? reproducibly benchmarking the methods of dynamic accuracy estimation of localization," *Sensors*, vol. 22, no. 3, 2022.

[19] G. M. Mendoza-Silva, P. Richter, J. Torres-Sospedra, E. S. Lohan, and J. Huerta, "Long-term wifi fingerprinting dataset for research on robust indoor positioning," *Data*, vol. 3, no. 1, 2018.

[20] P. Roy, C. Chowdhury, D. Ghosh, and S. Bandyopadhyay, "Juindoorloc: A ubiquitous framework for smartphone-based indoor localization subject to context and device heterogeneity," *Wireless Personal Communications*, vol. 106, 05 2019.

[21] E. Sansano, R. Montoliu, O. Belmonte, and J. Torres-Sospedra, "Uji indoor positioning and navigation repository," 2016.

[22] P. Pascacio, J. Torres-Sospedra, A. Jiménez, and S. Casteleyn, "Mobile device-based bluetooth low energy database for range estimation in indoor environments," *Scientific Data*, vol. 9, 06 2022.