Proximity Sensing: Modeling and Understanding Noisy RSSI-BLE Signals and Other Mobile Sensor Data for Digital Contact Tracing

Sheshank Shankar **Rishank Kanaparti** **Ayush Chopra** **Rohan Sukumaran**

Parth Patwa **Myungsun Kang** **Abhishek Singh** **Kevin P. McPherson**

Ramesh Raskar

PathCheck Foundation MIT Media Lab
Cambridge, MA
{first.lastname}@pathcheck.org

Abstract

As we await a vaccine, social-distancing via efficient contact tracing has emerged as the primary health strategy to dampen the spread of COVID-19. To enable efficient digital contact tracing, we present a novel system to estimate pair-wise individual proximity, via a joint model of Bluetooth Low Energy (BLE) signals with other on-device sensors (accelerometer, magnetometer, gyroscope). We explore multiple ways of interpreting the sensor data stream (time-series, histogram, etc) and use several statistical and deep learning methods to learn representations for sensing proximity. We report the normalized Decision Cost Function (*nDCF*) metric and analyze the differential impact of the various input signals, as well as discuss various challenges associated with this task.

1 Introduction

As economies open up, digital contact tracing is emerging as an important tool to help contain the spread of COVID-19 by providing exposure notification to individuals who come in close proximity to infected individuals [3, 6, 16, 27]. There have been several proposals for proximity sensing varying across different modalities, including WiFi signals [5, 21, 25], GPS [20], Ultrasound [15], and QR codes [17]. However, Bluetooth is the most widely accepted technology for digital contact tracing, as it is the most feasible across the available options. Moreover, it is supported by the Google Apple Exposure Notification (GAEN) API [8]. Current approaches for automated exposure notification use BLE signals emanating from smartphones (chirps) to detect if a person has been in close proximity to an infected individual. However, there are certain limitations with the accuracy of Bluetooth, as described in [13, 14, 23]. Researchers have tried augmenting Bluetooth-based protocols with other modalities (such as Ultrasound [15] and GPS [20]), however, that does not circumvent the signal processing limitation present in Bluetooth based systems. Existing attempts at calibrating Bluetooth data only use device level information [7] to partially account for hardware level differences among the devices. In addition, the received signal strength indicator (RSSI) value of Bluetooth chirps sent between phones is a noisy estimator of the actual distance between the phones as they can be dramatically affected by real-world conditions [9].

In this paper, we predict the distance between two phones using the RSSI values from BLE signal logs, along with data from other mobile sensors (as described in Table 1). The additional sensors help account for the real-world complexities that affect the variations in BLE signals, such as the position of the phone, movement, etc. Using datasets collected by NIST and MITRE, we experiment with various machine learning models to learn the representations of the data, achieving the most favorable results by using a temporal Conv1D. Additionally, to understand the contribution of various sensors on the overall task as well as the extent of noise present in this data distribution, we perform ablation studies and discuss various challenges associated with the data distribution.

2 Preliminaries

In our experiments, we use the NIST dataset (collected in coordination with the MIT PACT project) and the MITRE Range Angle Structured (further referred to as MITRE) dataset. Both the datasets are collected using the Structured Contact Tracing Protocol V 2.0 [12] and provide a stream of Bluetooth RSSI data between two phones at the following distances, 1.2, 1.8, 3.0, and 4.5 meters. We use these distances as classes. In addition, the dataset collects other phone sensor data (listed in Table 1), as well as experiment level metadata (such as device models, power, carriage states, etc).

The dataset is formatted as multiple-segmented intervals of continuous sensor readings (between transmitter and receiver) at the same distance. This is done for multiple 4-second device interactions per experiment. We primarily use the MITRE and a set-aside subset of the NIST datasets for training and evaluation respectively. However, results of models trained on a separate subset of the NIST dataset are also discussed.

3 Methodology

As can be seen in Figure 2, the raw Bluetooth RSSI and other sensor data is extremely noisy. Hence, we exploit the temporal characteristics of the dataset by modeling each experiment as a time series, breaking each 4-second interval into multiple time-steps. To minimize noise and the need for over-sampling and under-sampling readings, we choose the mean number of samples per each 4-second interval (150) as the fixed length of each time series. Every time-step is represented as a normalized fixed-length feature vector representing the most recent values obtained from each sensor. In addition, the metadata is one-hot encoded and concatenated to each timestep's vector. Figure 1 shows the data processing pipeline for the non-temporal models that do not make use of a time-series input.

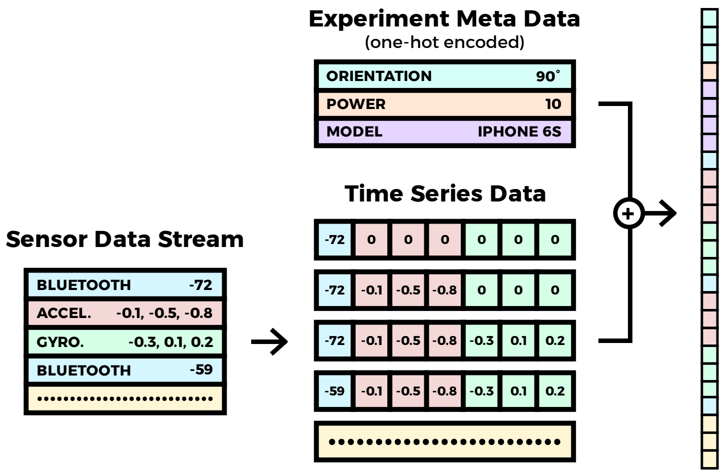

Figure 1: A *simplified* visualization of the data processing pipeline for the **concatenated** time-steps input. After the sensor data stream is converted into a time series of 150 timesteps, they are concatenated and merged with the one-hot encoded meta-data to create a single feature vector.

Inspired by [10], we also experiment with a histogram representation of the BLE signals. Here, we use the frequencies of RSSI values in various buckets instead of using the actual RSSI values. We also try using mix-up data augmentation [26], to reduce overfitting by increasing the variations present in the MITRE dataset. However, it did not provide a significant performance increase.

To model this complex data, we use classifiers derived from various DL models (feed-forward, LSTM, GRU, temporal one-dimensional convolutional network, and a convolutional GRU [22]), Decision Tree-based models (XGBoost [2], Random Forests [1]), as well as with support vector machines (SVM) [4] and a Naive Bayes Classifier. The majority of the models are trained on the entire MITRE dataset, however, the SVM and tree-based models are only trained on a smaller subset of the dataset. This is because training on the full dataset for these model architectures do not complete in a reasonable amount of time. To understand the variations caused by minor changes in the input, we also model the task as a regression problem, where we round the outputs to the nearest class.

We evaluate results on a test set from MITRE and NIST datasets with the *normalized Decision Cost Function (nDCF)* metric [24] that effectively combines the probability of a false negative and false positive into a single value using weights reflecting the relative cost of each type of error.

All experiments were run on a Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz server with 528 GB RAM, 48 cores, and on a single Nvidia 1080 Ti GPU. The temporal networks are built using PyTorch [18], whereas the Support Vector Machine and the Decision Tree based models are implemented using scikit-learn [19]. All the deep learning experiments use the Adam optimizer [11] and optimize Categorical Crossentropy loss. We plan to open source the code.

4 Results and Analysis

After rigorous parameter tuning, we present the best results on various model architectures evaluated on a subset of the NIST dataset and trained on either the MITRE dataset or a separate subset of the NIST dataset. As highlighted in Table 2, the temporal one-dimensional convolutional network shows the best performance. Surprisingly, the Nu-Support Vector Classifier is able to perform relatively well even with a substantially smaller training subset of the MITRE dataset (as described in the Methodology section). This suggests that there could be less variation in the MITRE dataset, contributing to lower levels of generalization. Additionally, for the random forest model architecture, using a regressor instead of a classifier decreases the nDCF by 0.19, possibly indicating that minor variations in the input could be relevant. The Convolutional GRU architecture performs extremely well when trained on the training subset of the NIST dataset, but significantly loses performance when trained on the MITRE dataset. Due to the complexity of the Convolutional GRU, we hypothesize that this model is overfitting to the conditions of the *location* that the NIST data was collected in.

Name	**Description**
Bluetooth	BLE strength
Accelerometer	linear acceleration
Gyroscope	angular velocity
Magnetometer	magnetic aberration changes
Attitude	roll, pitch, yaw
Gravity	Gravity along X, Y, Z axes
Altitude	atmospheric pressure changes
Compass	orientation & direction

Table 1: Descriptions of mobile sensors present in both datasets.

Network	**Train Set**	**nDCF**
ConvGRU	MITRE	1.02
RNN (GRU)	MITRE	0.97
Feed Forward	MITRE	0.77
Conv1D	**MITRE**	**0.58**
Conv1D (MaxPool)	MITRE	0.81
Conv1D (Dilation)	MITRE	0.70
C-SVC	MITRE	0.99
Nu-SVC	MITRE	0.77
XGBoost	MITRE	1.02
RF Classifier	MITRE	1.04
RF Regressor	MITRE	0.85
RF Histogram (Regressor)	MITRE	0.90
Naive Bayes	MITRE	0.92
GRU	NIST	0.28
ConvGRU	**NIST**	**0.16**

Table 2: Results for various model representations, evaluated on a subset of the MITRE and NIST dataset.

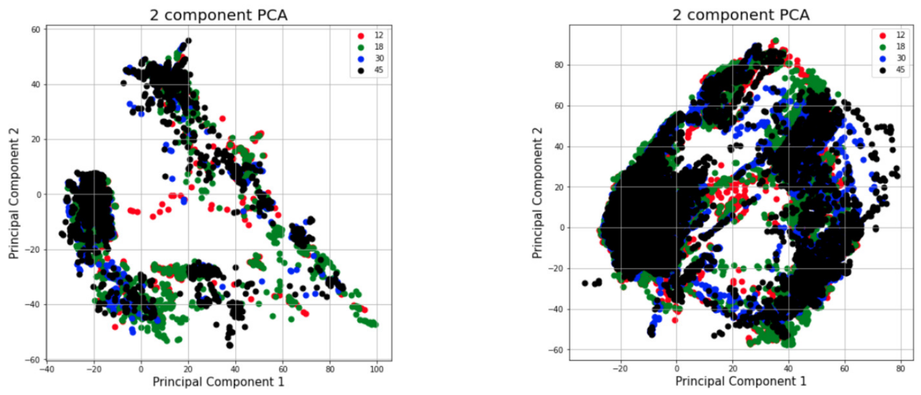

Figure 2: The PCA of the MITRE (left) & NIST (right) datasets shows there are no clear decision boundaries present among the classes in this low dimensional representation.

4.1 Ablation studies

We perform preliminary ablation studies by excluding data from input data streams to estimate the role of different sensors in proximity sensing. Due to the extensive number of potential subsets, we limit our experiments to only subsets based on a physics intuition. For example, we find using only the RSSI-BLE readings increases the divergence in the final training and testing accuracy, indicating higher susceptibility to overfitting. After training with multiple other combinations of sensors, we achieve the best performance using the gyroscope, accelerometer, magnetometer, and the RSSI-BLE readings. When excluding experiment-level metadata, we do not observe any significant performance change. We find that training is actually susceptible to overfitting on two classes when we include the experiment-level metadata.

5 Discussion

In this section, we discuss the feasibility of proximity sensing with respect to the distribution of data. We perform low dimensional projections of the dataset to identify underlying target class clusters in the feature space. Figure 2 shows the Principal Component Analysis (PCA) plots for both datasets. In both plots, clusters are heavily overlapping, without a clear decision boundary for any two labels. This could be caused by external factors such as, physical barriers present between phones, the number and time spread of observed chirps, and multi-path signals reflected from surfaces (e.g. indoor vs outdoor). However, higher dimensional hyper-planes dividing the classes may exist.

A major challenge we encounter across experiments is the lack of generalization of the models trained on MITRE dataset to the NIST dataset. To assess the generalizability of the NIST dataset, we train on NIST dataset and evaluation on the MITRE dataset, but it does not yield any notable improvement in generalization. Informed by these results, we try to estimate the gap between the MITRE and NIST distributions. The inconsistencies between the two distributions can be observed by the high values of ℓ_2 between the pair-wise distances of any two feature vectors. Looking at the cross-dataset nearest neighbour pairs, we find that a significant number of the pairs had different classes. Further proving this data discrepancy, we receive similar results when training on an *optimal training subset* that includes only the 2 nearest points in the MITRE dataset for each point in the NIST dataset is included. We also find that the average distance between a nearest-neighbor pair have an ℓ_2 distance of 24; however, if we isolate nearest-neighbor pairs with different classes, the average ℓ_2 distance is around 200. This supports the argument that the training and evaluation data distributions are not similar enough to capture any generalizable information. However, a thorough statistical analysis is needed to confirm this argument, as highly non-linear manifolds that can fit both distributions may exist.

6 Conclusion and Future Work

In this paper, we present our approach for detecting proximity between phones for digital contact tracing. Compared to current systems that only rely on RSSI values, our approach attempts to fuse knowledge from additional sensors available on smartphones to mitigate the noise present in the Bluetooth signal. Out of all the models we try, Conv1D performs the best. We report our findings and analysis over different data streams and methods. Effectively sensing proximity was marked by several challenges due to the noise in the data distribution and hence the lack of generalization.

In the future, we plan to work on more interpretable modeling and extensive breakdown of different sensor contribution to the prediction. A combination of physics based forward model and data driven predictor will also be a good step towards robustly detect proximity. We plan to incorporate other co-location technologies (such as WiFi, GPS, Ultrasound, and QR) with our current approach to provide more accurate results.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[3] Hao-Yuan Cheng, Shu-Wan Jian, Ding-Ping Liu, Ta-Chou Ng, Wan-Ting Huang, and Hsien-Ho Lin. Contact tracing assessment of covid-19 transmission dynamics in taiwan and risk at different exposure periods before and after symptom onset. *JAMA internal medicine*, 2020.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] Mikhail Dmitrienko, Abhishek Singh, Patrick Erichsen, and Ramesh Raskar. Proximity inference with wifi-colocation during the covid-19 pandemic, 2020.

[6] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 368(6491), 2020.

[7] Google. *Exposure Notifications BLE RSSI calibration procedure*, October 2020. `https://developers.google.com/android/exposure-notifications/ble-attenuation-procedure`.

[8] Google and Apple. Exposure notification: Bluetooth specification, 2020. `https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf`.

[9] Gary F Hatke, Monica Montanari, Swaroop Appadwedula, Michael Wentz, John Meklenburg, Louise Ivers, Jennifer Watson, and Paul Fiore. Using bluetooth low energy (ble) signal strength estimation to facilitate contact tracing for covid-19. *arXiv preprint arXiv:2006.15711*, 2020.

[10] Tianlang HE and Maximilian PRINTZ. A 2-stage classifier for contact detection with bluetoothLE and INS signals. *NIST TC4TL Challenge*, 2020.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] MIT Lincoln Laboratory. Structured contact tracing protocol, v. 2.0 (1.5), 2020. `https://mitll.github.io/PACT/files/Structured%20Contact%20Tracing%20Protocol,%20V.%202.0%20(1.5).pdf`.

[13] Douglas J. Leith and Stephen Farrell. Coronavirus contact tracing: Evaluating the potential of using bluetooth received signal strength for proximity detection, 2020.

[14] Douglas J Leith and Stephen Farrell. Measurement-based evaluation of google/apple exposure notification api for proximity detection in a commuter bus. *arXiv preprint arXiv:2006.08543*, 2020.

[15] Po-Shen Loh. Flipping the perspective in contact tracing. *arXiv preprint arXiv:2010.03806*, 2020.

[16] Chandini Raina MacIntyre. Case isolation, contact tracing, and physical distancing are pillars of covid-19 pandemic control, not optional choices. *The Lancet Infectious Diseases*, 20(10):1105–1106, 2020.

[17] Ichiro Nakamoto, Sheng Wang, Yan Guo, and Weiqing Zhuang. A qr code–based contact tracing framework for sustainable containment of covid-19: Evaluation of an approach to assist the return to normal activity. *JMIR Mhealth Uhealth*, 8(9):e22321, Sep 2020.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[20] Ramesh Raskar, Abhishek Singh, Sam Zimmerman, and Shrikant Kanaparti. Adding location and global context to the google/apple exposure notification bluetooth api. *arXiv e-prints*, pages arXiv–2007, 2020.

[21] Piotr Sapiezynski, Arkadiusz Stopczynski, David Kofoed Wind, Jure Leskovec, and Sune Lehmann. Inferring person-to-person proximity using wifi signals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2):1–20, 2017.

[22] Mennatullah Siam, Sepehr Valipour, Martin Jagersand, and Nilanjan Ray. Convolutional gated recurrent networks for video segmentation. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3090–3094. IEEE, 2017.

[23] Ashkan Soltani, Ryan Calo, and Carl Bergstrom. Contacttracing apps are not a solution to the covid-19 crisis. *The Brookings Institution, April*, 27, 2020.

[24] NIST TC4TL. Nist pilot too close for too long (tc4tl) challenge evaluation plan, July 2020. `https://www.nist.gov/system/files/documents/2020/07/01/2020_NIST_Pilot_TC4TL_Challenge_Evaluation_Plan_v1p3.pdf`.

[25] Amee Trivedi, Camellia Zakaria, Rajesh Balan, and Prashant Shenoy. Wifitrace: Network-based contact tracing for infectious diseasesusing passive wifi sensing. *arXiv preprint arXiv:2005.12045*, 2020.

[26] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[27] Juanjuan Zhang, Maria Litvinova, Yuxia Liang, Yan Wang, Wei Wang, Shanlu Zhao, Qianhui Wu, Stefano Merler, Cécile Viboud, Alessandro Vespignani, et al. Changes in contact patterns shape the dynamics of the covid-19 outbreak in china. *Science*, 2020.