Cellular-Assisted, Deep Learning Based COVID-19 Contact Tracing

FAN YI, Princeton University, USA
YAXIONG XIE, Princeton University, USA
KYLE JAMIESON, Princeton University, USA

The coronavirus disease (COVID-19) pandemic has caused social and economic crisis to the globe. Contact tracing is a proven effective way of containing the spread of COVID-19. In this paper, we propose CAPER, a Cellular-Assisted deeP lEaRning based COVID-19 contact tracing system based on cellular network channel state information (CSI) measurements. CAPER leverages a deep neural network based feature extractor to map cellular CSI to a neural network feature space, within which the Euclidean distance between points strongly correlates with the proximity of devices. By doing so, we maintain user privacy by ensuring that CAPER never propagates one client's CSI data to its server or to other clients. We implement a CAPER prototype using a software defined radio platform, and evaluate its performance in a variety of real-world situations including indoor and outdoor scenarios, crowded and sparse environments, and with differing data traffic patterns and cellular configurations in common use. Microbenchmarks show that our neural network model runs in 12.1 microseconds on the OnePlus 8 smartphone. End-to-end results demonstrate that CAPER achieves an overall accuracy of 93.39%, outperforming the accuracy of BLE based approach by 14.96%, in determining whether two devices are within six feet or not, and only misses 1.21% of close contacts. CAPER is also robust to environment dynamics, maintaining an accuracy of 92.35% after running for ten days.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing**.

Additional Key Words and Phrases: COVID-19, Proximity estimation, Contact tracing, Cellular network, LTE, Neural networks

ACM Reference Format:
Fan Yi, Yaxiong Xie, and Kyle Jamieson. 2022. Cellular-Assisted, Deep Learning Based COVID-19 Contact Tracing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3, Article 150 (September 2022), 27 pages. https://doi.org/10.1145/3550332

1 INTRODUCTION

The ongoing coronavirus disease (COVID-19) pandemic has already resulted in millions of deaths all around the world, causing unprecedented social and economic crisis. Except vaccines, a proven effective way of containing the spread of COVID-19 is accurate, complete and timely *contact tracing*. According to the world health organization (WHO) [54], a *close contact* is a person who has been within six feet to someone that is COVID-19 positive for more than 15 minutes. Contact tracing is the process of identifying, assessing, and managing people who are close contacts with the contagious positive cases. Currently, contact tracing is accomplished through manually interviewing each COVID-19 positive cases by the health authorities, which, however, is extremely labor intensive, time consuming, and unscalable, motivating techniques that can accelerate and automate the process of contact tracing.

The core task of contact tracing is to identify the close contacts of the positive cases, which requires comparing the locations of two citizens. The ubiquity of smartphones, *e.g.*, more than 81% of Americans own a smartphone

Authors' addresses: Fan Yi, Princeton University, Princeton, USA, fanyi@princeton.edu; Yaxiong Xie, Princeton University, Princeton, USA, yaxiongx@princeton.edu; Kyle Jamieson, Princeton University, Princeton, USA, kylej@princeton.edu.

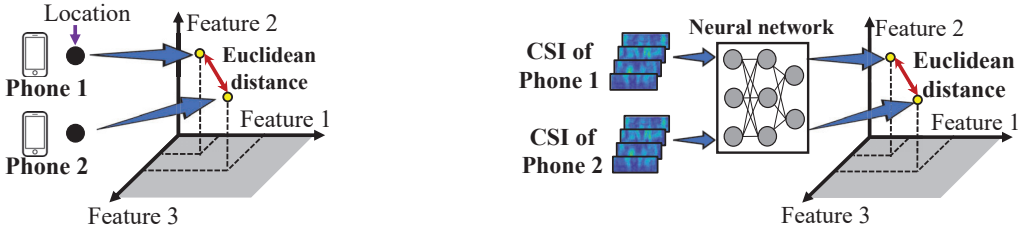
2474-9567/2022/9-ART150
https://doi.org/10.1145/3550332

in 2019, and their capability of performing device localization and proximity estimation make them ideal devices for building an automatic and fast contact tracing system that scales.

Identifying close contacts by comparing the exact locations of smartphones is a straightforward solution. There exists many mature techniques we can leverage to accurately localize the smartphones, including GPS [21], Wi-Fi based [27, 56–59], cellular based [13], Bluetooth based [5, 28], acoustic signal based [26, 49] and visible light based techniques [30, 61, 65]. Exposing the location to any third party, however, hinders the privacy of both healthy citizens and the people who have been infected with the virus, making the location based solutions impractical to implement.

Essentially, identifying close contacts only requires the distance between two people, so localizing the smartphones is an overkill. Knowing the proximity of devices is enough for contact tracing, which also preserves user privacy. Proximity estimation using RSSI of Bluetooth has drawn a significant amount of attention from the research community [11, 44, 44, 46, 51], and the industry [2, 9, 20].

A common approach, including the Google and Apple Exposure Notifications System [20], relies on signal strength of Bluetooth beacons to identify close contacts, which is defined as a distance of 6 feet [8]. However, such RSSI based approaches suffers from errors [64] and fail to provide accurate close contact estimation, as there are many factors other than the distance that can affect the received signal strength, including hardware imperfections, interference from other signals that share the ISM band and multipath effect. Furthermore, these systems require the device to frequently transmit beacons to detect each other, which consumes a large amount of energy and also makes the device trackable by malicious third parties. Techniques like MAC address shuffling have been used to hide the identity of the users, so the malicious third party cannot link the location it obtained to a physical device, which, however, can be easily hacked using techniques like RF fingerprinting [39].



(a) Location mapping. (b) CSI mapping.

Fig. 1. Mapping the physical locations of smartphones (**a**), and CSI measured from smartphones (**b**) to a high-dimensional feature space, within which the Euclidean distance between points indicates the proximity of smartphones.

We propose a location based proximity estimation algorithm that is accurate in estimating proximity and at the same time preserves user privacy by hiding the exact user location. We plot the intuition of our algorithm in Figure 1(a), from which we see that, instead of directly comparing the physical locations to get the proximity of two devices, we propose to map the locations to a point inside a high-dimensional feature space and then calculate the Euclidean distance inside the feature space to derive the proximity. To achieve this goal, we have two requirements for such mapping. First, we require that the Euclidean distance of two points in the feature space indicates the proximity of their corresponding physical locations. Second, to preserve user privacy, the mapping should be hard to reverse so any third party cannot reconstruct the exact user location using the exposed location inside the feature space.

To realize our idea in Figure 1(a), two tasks remain unsolved: localizing the smartphone and finding a mapping that that satisfies the aforementioned two requirements.

For the localization task, we rely on the channel state information (CSI). Using CSI, we can estimate the parameters of all the paths the signal travels along, such as angle of arrival (AoA) and time of flight (ToF), which

helps the proximity estimation from two aspects. First, the parameters of direct path can be directly used to localize the smartphone [27, 57, 59]. Second, the reflections from nearby furniture, floor or walls describe the surrounding environment near the device, so the similarity in parameters of reflection paths estimated by two smartphones also indicates their proximity, as nearby devices share similar propagation environment.

According to the above analysis, the algorithm that translates the CSI into user locations and then performs the mapping loses the information of the reflection multipaths. Therefore, to fully make use of the information conveyed by CSI, we directly map the CSI to a point in the feature space, as shown in Figure 1(b). In our implementation, we mainly rely on cellular CSI instead of Wi-Fi CSI because of two reasons. First, comparing with Wi-Fi, the cellular signal is ubiquitous, providing national wide coverage. Second, cellular base station broadcast reference signals at the frequency of millisecond (§2), so the smartphone is able to collect densely sampled, uniformly distributed CSI by decoding the reference signal. Leveraging downlink CSI also eliminates the peer-to-peer transmissions between smartphones, which prevents the smartphone from being tracked by any malicious third parties. We use Wi-Fi as a complementary information source to further extend the coverage of CAPER to challenging indoor scenarios, such as the basement.

For the mapping task, we propose to leverage a deep learning based feature extractor to find the mapping that satisfies our two requirements, as shown in Figure 1(b). Recent advances in deep learning has proven that convolutional deep neural network (CNN), is powerful in selecting representative features for diverse tasks. Therefore, after training, a CNN based feature extractor would automatically select the set of features that forms the desired high-dimensional feature space within which the Euclidean distance indicates the proximity. Furthermore, the extracted features become incomprehensible when the neural network goes deep.

We implement a prototype of CAPER using USRP as frontend to collect CSI from commercial cell towers. Extensive experimental results show that CAPER achieves an overall accuracy of 93.39%, outperforming the accuracy of BLE based approach by 14.96%, in identifying close contacts, *i.e.*, determining whether two devices are within six feet or not, and only misses 1.21% of close contacts. CAPER is also robust to environment dynamics, achieving an accuracy of 92.35% after running for ten days. Microbenchmarks show that our neural network model runs in 12.1 microseconds on the Oneplus 8 smart phone.

2 LTE PRIMER

In this section, we introduce the LTE primer. We focus on the frequency division duplexing (FDD), the most widely used mode in current deployed commercial cellular network.

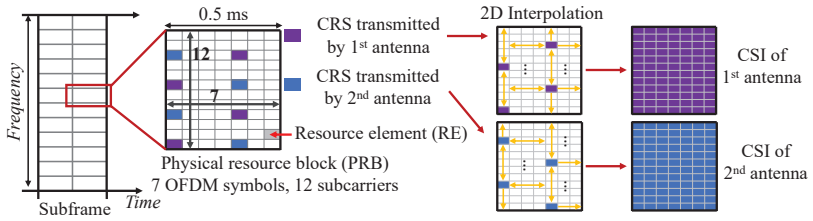


Fig. 2. LTE groups multiple REs into a PRB. Inside each PRB, the base station transmits CRS at specific REs, whose channel can be directly estimated. Mobile client performs interpolation to derive channel estimations of all REs.

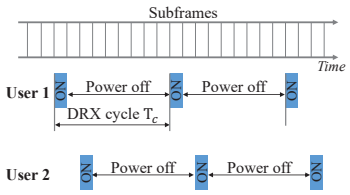


Fig. 3. A mobile client in DRX mode wakes up periodically to receive data from base station.

Physical layer frame structure. LTE adopts OFDM in the physical layer, so the smallest time-frequency unit is one subcarrier in frequency and one OFDM symbol in time, which is also denoted as one *resource element* (RE). LTE groups all REs inside a block spanning seven OFDM symbols and 12 subcarriers into a *physical resource*

block (PRB), as shown in Figure 2. To support multiple access, LTE divides the time into one millisecond length *subframes* and allocates the PRBs inside each subframe to one or multiple mobile users for data transmission.

Channel estimation in LTE network. To facilitate channel estimation, the base station transmits predefined *cell specific reference signal* (CRS), inside several specific REs of a PRB, as shown in Figure 2. The CRSs transmitted by multiple antennas of one base station are non-overlapping with each other, so a mobile client can separate them and estimate the channel between each transmitting antennas and its receiving antenna independently. Since the sequence of CRS is known, a mobile client is able to estimate the channel of all REs that carry CRS. To obtain the channel estimation of every RE, the mobile client performs a two-dimensional interpolation, *i.e.*, over time and frequency, as shown in Figure 2. We note that the base station always broadcast the CRS no matter it has data to transmit or not, so a mobile user can estimate the downlink channel at any time point.

Discontinuous reception (DRX). To save energy, the cellular network supports *discontinuous reception* (DRX) mode, in which the mobile user wakes up periodically, *e.g.*, every 10 subframes or milliseconds, to receive, if there is any, data from the base station. The interval between two waking up events is one DRX cycle, as shown in Figure 3.

Radio resource control (RRC) modes. A cellular mobile client is in RRC *connected mode* when it is communicating with the cell tower, and enters RRC *idle mode* when the transmission finishes. A mobile client in idle mode adopts DRX to periodically wake up to receive any possible messages from the base stations. To prepare for possible handover, each mobile client also periodically measures the channel quality of neighbouring cells.

3 RELATED WORK

Contact tracing is widely used to slow down the spread of COVID-19 [8]. Traditional contact tracing methods involve case investigation, where the authority collects trajectory information of positive cases, then manually identify their possible close contacts, and at last contact and give guidance to those potentially exposed individuals. Case investigation has demonstrated its effectiveness in many countries, but is also extremely labor intensive, time consuming and unscalable, which motivates technology-empowered automatic and accurate contact tracing methods.

Localization based contact tracing. Diverse techniques have been proposed to localize mobile devices. GPS can localize devices with meter-level accuracy in outdoor scenarios [21], but is also known to be inaccurate in indoor environments. To address the problem of indoor localization, techniques that leverage Wi-Fi signals [27, 56–59], Cellular signals [13], Bluetooth beacons [5, 28] and visible light [30, 61, 65] have been proposed and achieve centimeter-level accuracy. Therefore, conducting contact tracing via localizing every citizen that carries a mobile device is possible from a technical standpoint [2, 29, 43, 45], which, however, raises serious concerns about the user privacy. Without getting the users' consent to frequently upload their locations, localization-based contact tracing becomes practically impossible. [52] propose a Wi-Fi MAC based contact tracing system, and deploy it at two campuses. This paper also designs a graph-based model to reduce the memory usage and index update overheads when the the number of users becomes large.

Proximity based contact tracing. Contact tracing using device proximity hides the exact user location and thus preserves user privacy, which makes it a promising solution. Diverse techniques have been proposed to estimate the proximity of mobile devices. WiFi based proximity estimation using RSSI of observed signal from AP has been proposed [37, 47] and its application in contact tracing has been preliminarily investigated in [15], which, however has error of meters to tens of meters and cannot work in outdoor scenarios where there exists no WiFi AP. WiFi proximity using direct signal transmission between two device has also been proposed [40], which, however, only works when the devices are in close proximity, *i.e.*, centimeters apart. Bluetooth based proximity estimation has been well studied [35, 36], and its application to contact tracing has been explored by

both the research community [11, 44, 46, 51] and commercial companies like Google and Apple [20]. Most of these contact tracing systems estimates the proximity using Bluetooth RSSI, whose value is affected by several factors other than distance, including the hardware, interference from signals that share the 2.4 GHz ISM band and the multipath effect, introducing significant error in the proximity estimation [64]. [60] proposes a cellular signal based contact tracing system that applies deep neural network for close contact identification. Comparing with [60], CAPER proposes a novel data preprocessing technique to preserve the continuity and periodicity in the CSI phase. Furthermore, to overcome overfitting in the received wireless data, we propose a customized network network structure by integrating multitask learning with Twin-net. CAPER also considers the problem of model adaptation to environmental changes and apply deep adaptation networks (DAN) to continuously update the model.

Neural networks as feature extractors. Neural networks, especially CNN, have been widely used as feature extractors because of their advantages in capturing patterns on structured data. Therefore, researchers have applied neural networks to address various communication and sensing tasks in wireless domain. RF-Pose3D [63] uses CNN to estimate 3D poses from RF signals. RF-EATS [22] applies fully-connected neural networks to address the liquid sensing problem. [34] runs RNN to realize room-scale hand tracking. DLoc[4] achieves a high accuracy for indoor localization by applying a CNN based localization algorithm. Our work different from the above systems leverages CNN to extract proximity features from pair of CSIs.

4 CAPER DESIGN

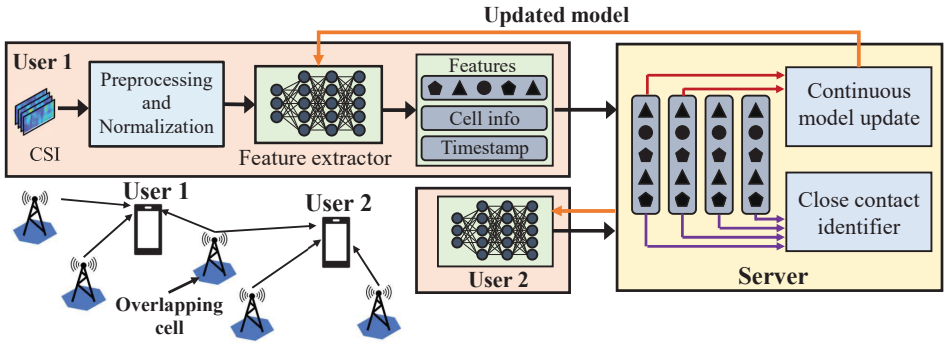


Fig. 4. The system architecture of CAPER. The mobile clients run feature extractor to map CSIs to feature vectors, and then send feature vectors packed with cell information and timestamps to the server. The server stores feature vectors reported by all mobile clients, and run close contact identifier to find all exposed users.

In this paper, we propose CAPER, a cellular-assisted, deep learning based contact tracing system, whose architecture is plotted in Figure 4. Generally, CAPER consists of the mobile client side data preprocessing module in Section 4.1 along with feature extractor module in Section 4.2, and server side close contact identifier in Section 4.3. Continuous model updating module and coverage extension are detailed in Section 4.4 and 4.5, respectively.

Mobile client side. Each cellular-connected mobile client first measures the CSI of the channels between itself with multiple cell towers, then maps the CSI to a vector of contact features via a deep-learning based feature extractor, and at last tags each feature vector with timestamp and information about the cell from which the CSI is measured, including the cell ID, antenna number, and bandwidth. We note that, each user reports feature vectors of multiple detected cell towers, so it is difficult to accurately localize the user using the cell IDs associated with

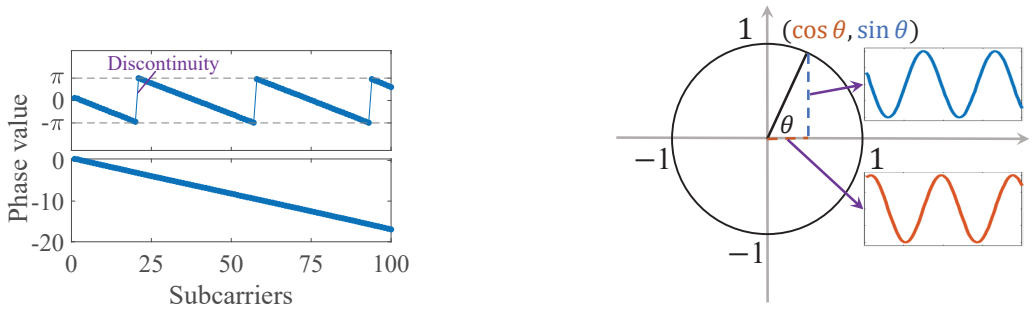
the features vectors as the combination of cell towers usually covers a large area of tens of square kilometers. The mobile client regularly uploads the extracted feature vectors together with the tagged information to the server.

Server side. The server stores feature vectors reported by all mobile clients. If one user tests positive for COVID-19, this user may choose to send a warning message to the server and unveil its identity. Upon receiving the message, the server starts the proximity estimation process to find all potentially exposed users, which consists of two step. First, to narrow down the search space, the server identify all the users who appear at the same cell coverage area at the same time with positive COVID-19 users, by checking the timestamp and cell information list associated with positive COVID-19 users against the list of all other users. Second, the server runs close contact identifier with feature vectors of potentially exposed users and that of positive COVID-19 users to further find out users who have close contacts with positive cases. To handle environment dynamics, the server continuously updates the feature extractor and then distributes the updated model to the mobile clients.

4.1 Data Preprocessing

The extracted CSI has a granularity of 14 samples per millisecond in RRC connected mode, which has information redundancy in time domain. To reduce the number of CSI being processed, we downsample the CSI to one sample per 32ms, if the extracted CSI is finer than this granularity. In addition, before feeding the downsampled CSI into the deep learning based feature extractor, we need to preprocess the CSI to eliminate the impact of noise and then transform the raw complex CSI matrix into a form that deep neural networks can interpret.

4.1.1 CSI Error Handling. The measured CSI is noisy because of hardware imperfections. We clean the CSI by performing CSI amplitude error correction and CSI phase offset calibration separately before feeding them into the neural network model. For CSI amplitude, we run Hampel filter to identify and remove outlier CSIs. To eliminate the phase error introduced by central frequency offset (CFO), sampling frequency offset (SFO) and symbol timing offset (STO), we feed the phase difference across antennas to the deep neural network. Phase difference captures the relative relationship between phase across antennas but loses the absolute value. We, therefore, input the sanitized phase of the first antenna [27], to compensate for the information loss.



(a) Phases across subcarriers. (b) Unit circle.

Fig. 5. Continuity is lost in wrapped phase and periodicity is lost in unwrapped phase **(a)**. Both continuity and periodicity is kept in the polar coordinates **(b)**.

4.1.2 Continuity and Periodicity of CSI Phase. Directly feeding the raw phase may confuse the neural network because of two reasons. On one hand, wrapped phase data loses its continuity. The wrapping operation restrains phase value in the range of $[-\pi, \pi]$, so it introduces frequent sudden jumps across subcarriers when the phase value rolls back. Neural networks are sensitive to sudden value changes, since jumps may convey important

information about the input data, *e.g.*, spikes in time series or edges in images. The phase jump caused by phase rollbacks are meaningless in phase data, but could draw significant attention of the neural network during training and thus mislead the network to select features that explains these jumps. On the other hand, the unwrapped phase data loses its periodicity. Theoretically, the phase value θ equals the phase value $\theta + 2\pi$. We lose such a relationship in the unwrapped data, as shown in Figure 5(a), so the neural network may misinterpret the phase value and thus miss the meaningful pattern conveyed inside the phase value.

According the above analysis, we conclude that inputting the raw phase value can only keep one characteristic of the phase, either the continuity or periodicity. To simultaneously maintain both the continuity and periodicity, we propose to input the polar representation of the raw phase. Specifically, we map raw phase data θ to its corresponding point on the unit circle, as shown in Figure 5(b), and uses the coordinate $[\cos\theta, \sin\theta]$ of such a point to represent the raw phase. We could see from Figure 5(b) that both the x-coordinate value and y-coordinate value are continuous and periodic across subcarriers. To guarantee that no information is lost, we input both the x-coordinate $\cos\theta$ and y-coordinate $\sin\theta$ to our deep-learning based feature extractor.

4.1.3 Normalization. Data normalization is a widely used technique to make the training process faster and more effective [6, 48]. We calculate the maximum amplitude over a 10-minute sliding window and then normalize the CSI amplitude according to the amplitude maximum inside the window. For CSI phase, we could see from Figure 5(b) that all phase are naturally normalized to the range of $[-1,\ 1]$ after being mapped to the unit circle.

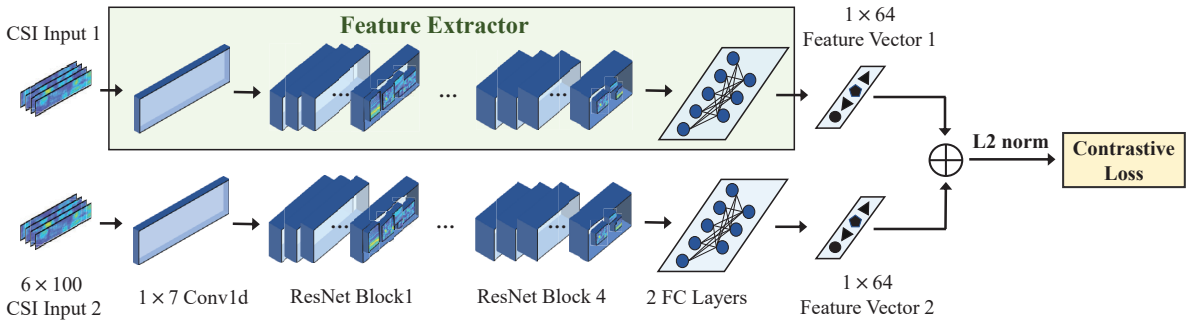


Fig. 6. The architecture of Siamese network, which runs two identical neural networks on two CSIs and outputs two feature vectors. It further feeds the L2 normalization of the two feature vectors into the contrastive loss function. The highlighted part shows the structure of our CNN based contact feature extractor.

4.2 CNN based Contact Feature Extractor

CAPER leverages a deep neural network to automatically select a set of features of the CSI. The requirement of the selected features is that the Euclidean distance between the feature vectors of any two CSIs represents the physical proximity of two mobile clients from which the CSIs are measured. We plot the structure of CAPER's deep learning based feature extractor in Figure 6. To fit deep learning model into mobile phones, we use half of the ResNet18 [24] layers in our feature extractor. The neural network initiates with one layer of 1×7 one-dimensional convolution, followed by 4 ResNet blocks, each consisting of 2 layers of 1×3 one-dimensional convolution. We add two fully connected layers after the ResNet block, which converts the intermediate outputs of convolutional layers into the final 1×64 feature vector.

4.2.1 Training the Contact Feature Extractor. Given the structure in Figure 6, we now introduce how to train the contact feature extractor to find the representative features.

Training with Siamese (twin) neural network. We adopt the Siamese (sometimes called the twin) neural network to train our feature extractor [7], which consists of two identical neural networks that run in parallel, as shown in Figure 6. These two neural networks share weights, and thus extract the same set of features of the input CSI. The Siamese network outputs the Euclidean distance of two feature vectors, which are extracted by the two parallel neural networks.

The goal of training is to minimize the loss function over the training dataset. We use *contrastive loss* [23] as our loss function, which is defined as:

$$\mathcal{L}_c = Y \cdot D^2 + (1 - Y) \cdot [\max(0, m - D)]^2 \tag{1}$$

where $\mathcal{L}_c$ represents the contrastive loss; D is the Euclidean distance of the two feature vectors generated from the two feature extractors; and Y is the proximity ground truth of two CSI inputs, which is a binary value representing whether the two devices are close or not. The m is a configurable hyperparameter whose value is set to 2 in our training.

The combination of Siamese network and contrastive loss turn the training process into a process of finding the weights that guarantee the Euclidean distance is minimized for any CSI pairs that are measured from two mobile devices that are within six feet, *i.e.* the close contacts; and maximized for CSI pairs that are measured from two far away mobile devices. Specifically, for CSI pairs with ground truth $Y = 1$, *i.e.* two devices are close contact with each other, the loss equals to:

$$\mathcal{L}_c = D^2 \tag{2}$$

so the training goal becomes minimizing the Euclidean distance D of feature vectors of the CSI pair. On the other hand, for CSI pairs with ground truth $Y = 0$, *i.e.* two far away devices, the loss equals to:

$$\mathcal{L}_c = [\max(0, m - D)]^2 \tag{3}$$

where the training goal becomes maximizing the Euclidean distance D. Furthermore, we see from Eq. 3 that, the contrastive loss focuses on maximizing the feature space distance of CSI pairs from two far away devices but have Euclidean distances D smaller than the hyper-parameter m and ignore the CSI pairs whose distance D are already large enough.

Multi-task learning. We observe in our experiments that the model trained with contrastive loss suffers from overfitting. The candidate contact feature space is large so the Siamese network may select features that closely fit the dataset but have no connection with the physical location of mobile users or with the similarity between two user locations. Hence, we apply domain knowledge to guide the neural network model to extract generalizable features that are linked to the physical location of the mobile user.

The domain knowledge we have is the AoA and ToF of multipaths around the mobile user, which can be estimated using SpotFi [27] from CSI. Furthermore, we expect the feature extractor to be able to extract contact features representing AoA and ToF, since they are directly related to the physical locations of users. Therefore, we propose to teach the feature extractor to learn desirable contact features with the ToF and AoA we estimated using SpotFi.

To combine our domain knowledge with the training process of the feature extractor, we create two additional tasks: estimating AoA and *time difference of flight* (TDoF) using neural network[1], and propose to solve three tasks simultaneously using multi-taks learning [12]. We plot a widely used architecture of multi-task learning in Figure 7, in which a shared subnetwork learns the shared features among tasks and multiple task-specific

[1]Here we estimate TDoF between different multipaths because the absolution ToF is inaccurate without time synchronization, while the difference between ToF of multipaths is accurate and stable [57].

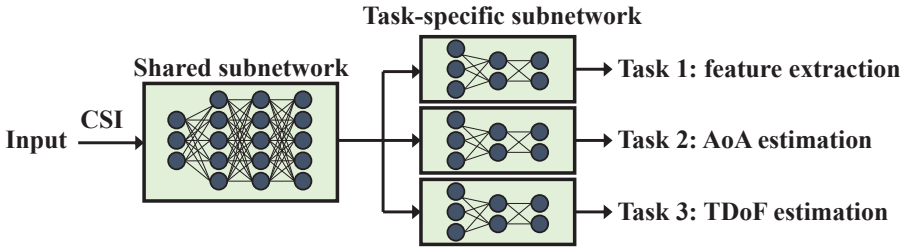


Fig. 7. The architecture of a multi-task learning network.

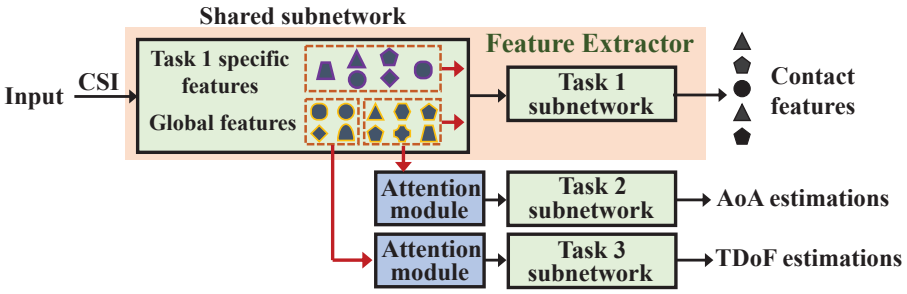


Fig. 8. The architecture of our training network designed according to multi-task attention network (MTAN).

subnetworks handles the uniqueness of its corresponding task. In our case, we train the shared subnetwork to learn features representing AoA and ToF, which is related to all three tasks: these features directly help AoA and ToF estimation tasks and also contribute to the feature extractor since they are related to the physical location of the user. Massive prior works that adopt multi-task learning in diverse applications have demonstrated that the shared subnetwork is able to learn the shared representing features (related to AoA and ToF), which in turn boosts the performance of each individual task.

To realize our idea, we use *multi-task attention network* (MTAN) [31] as a reference to design our end-to-end training network. We connect part of the neural network of our primary task, *i.e.*, contact feature extraction, to the subnetworks of the other two tasks, as shown in Figure 8, and train the shared network to learn both primary task specific features and global features that has impact across all tasks. The task two and three select a set of features in which they are interested, from the global feature pool using an *attention module*, and then feed the selected features into their task specific subnetworks. We use mean square error (MSE) of the prediction error as the loss for task two and task three, which is defined as:

$$\mathcal{L}_m = \frac{\sum_{i=1}^{n} \left(y_i - y_i^p\right)^2}{n}, \tag{4}$$

where y_i and y_i^p are estimated (using SpotFi) and predicted (using neural network) AoA or TDoF values respectively, and n is the number of estimations we have, *i.e.*, given that we estimate the parameters of L multipaths using SpotFi, we have $n = L$ for AoA estimation and $n = L - 1$ for TDoF estimation since we calculate the time difference across multipaths.

End-to-end network. We implement our end-to-end training network by combining the Siamese network in Figure 6 and the multi-task learning network in Figure 8. The final loss $\mathcal{L}_f$ of the whole network is the sum of

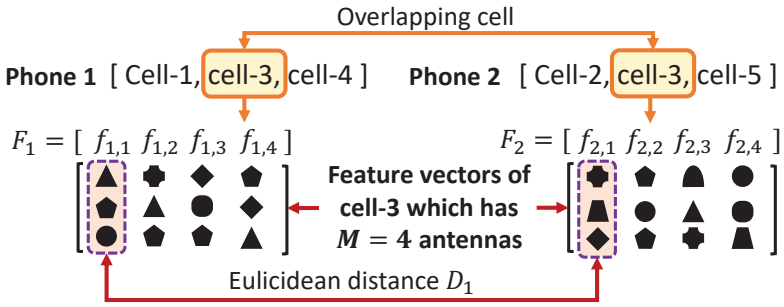


Fig. 9. Each smartphone reports the feature vectors of a group of neighbouring cells. Two smartphones may have one or multiple overlapping cells in their reported data.

the contrastive loss of our primary task and the MSE loss of AoA and TDoF estimation tasks we create:

$$\mathcal{L}_f = \mathcal{L}_c + \sum_{i=1}^{2} \beta_A \mathcal{L}^i_{m,A} + \sum_{i=1}^{2} \beta_T \mathcal{L}^i_{m,T} \tag{5}$$

where β_A and β_T is the weighting hyperparameter controlling the contribution of AoA and TDoF estimation tasks to the final loss, and we report their values in Section 6; the $\mathcal{L}^i_{m,A}$ and $\mathcal{L}^i_{m,T}$ are the MSE loss of the AoA and TDoF estimation tasks associated with the two parallel neural network in the Siamese network. By minimizing the loss $\mathcal{L}_f$, the neural network solves all tasks simultaneously.

4.2.2 Diverse Physical Layer Configurations. The configurations of the cellular physical layer determine the size of CSI matrices that are fed into the contact feature extractor. To be more specific, the size of the CSI matrix is represented as $M \times N \times S$, where M and N is the number of antennas in array of the base station and the mobile phone, respectively, and S is the number of subcarriers, which is determined by the channel bandwidth of the base station [2].

The physical layer configuration varies across base stations and mobile devices, so the size of CSI varies accordingly. A possible solution is to train one model to handle one corresponding input CSI size and distribute all the models to the mobile device, which, however, results in large training overhead since the number of possible CSI sizes is large. Specifically, in real implementation, the base station may have one, two or four antennas in its array, and communicate using a 5 MHz, 10 MHz and 20 MHz channel, and most of current smartphone have two antennas, resulting in nine possible bandwidth and array size combination. To reduce the training overhead, we only train for the combinations of one base station antenna ($M = 1$), two mobile phone antennas ($N = 2$) and all available bandwidth. We separate the CSI with $M > 1$ into M CSIs with antenna number $M = 1$, so we can reuse the single antenna model. As a result, for a cell with $M > 1$ antennas, the i-th smartphone reports a feature matrix:

$$F_i = [f_{i,1}, \ldots, f_{i,M}]^T, \tag{6}$$

where $f_{i,j}$ is the feature vectors extracted by the CNN based feature extractor using CSI of channel between j-th antenna of the base station to i-th user, as shown in Figure 9.

4.2.3 The CSI Time Misalignment. Two mobile devices in DRX mode may wake up at misaligned subframes and measure CSI at different time point, as shown in Figure 3. The widely support DRX cycle are 0.32 second, 0.64 second and 1.28 seconds, so the maximum misalignment between two CSI measurements is 0.64 second, within which the location of a mobile devices changes negligibly. We have CSI pairs with time misalignment ranging

[2] A mobile device measures the CSI of the entire frequency band of the channel (§2), regardless of the detailed bandwidth allocation.

from zero to a maximum of 0.64 second in our dataset. Our experimental results in Section §7 demonstrate that the trained model works effectively even in presence of diverse time misalignment.

4.3 Close Contact Identifier

When a positive case is reported, the server runs the close contact identifier to find all users that are potentially exposed to the COVID-19 positive user, which involves two steps. First, the identifier reduces the search space by finding users who has overlapping cells in their reported the feature vectors, as shown in Figure 9. Second, the identifier scans all users found in step one and identifies all possible close contacts that has been exposed to the positive case.

In the rest of this section, we introduce our algorithm to identify close contacts using reported feature vectors. Since each user reports feature vectors of a group of neighbouring base stations, the number of overlapping cells between two users may vary. We, therefore, begin with the introduction of close contact identification under the scenario of one overlapping cell and then generalize to multi-cell cases.

4.3.1 Single Overlapping Cell. Supposing two smart phones share one base station that has M antennas in its array, the close contact identifier first calculates the Euclidean distance D_j between the feature vector of $f_{1,j}$ and $f_{2,j}$, as shown in Figure 9, and then derive the average Euclidean distance between the feature vectors F_i of two users as:

$$D = \frac{1}{M}\sum_{1}^{M} D_j, \tag{7}$$

based on which, the identifier makes a preliminary identification P_{pre}:

$$P_{pre} = \begin{cases} -1, & \text{if } D \leq D_{threh} \\ 1, & \text{if } D > D_{threh}, \end{cases} \tag{8}$$

where D_{threh} is the prediction threshold. We set D_{threh} to 0.9 in our experiments. A preliminary identification of $P_{pre} = -1$ means the two devices, that report the two feature vectors, are within 6 feet with each other, and vice versa.

We note that this preliminary estimation is made on a single pair of CSI, which covers only one millisecond in time. Due to the fine granularity of downlink reference signal, we can have dense preliminary estimations within a short period, where the specific number of estimations depends on CSI sampling rates. Therefore, we propose to add another voting layer on top of the preliminary estimations, to mitigate the influence of sudden environmental changes or unpredictable interference, making the close contact identifier more robust. The voting result $\mathcal{P}_v$ over a series of feature vector pairs is defined as follows:

$$\mathcal{P}_v = \sum_{t=1}^{k} P_{pre,t} \tag{9}$$

where k is the number of feature vector pairs in a voting, $P_{pre,t}$ is the preliminary estimation on the t-th feature vector pair. If the resulting voting decision $\mathcal{P}_s \leq 0$, the final estimation is that these two devices has a close contact in the voting time span, and vice versa.

Sampling rate offset. In our current implementation, we perform one vote using all preliminary estimations obtained from feature vector pairs from a 18-second time window. A challenge we met when paring the feature vectors is that the number of feature vectors reported by two devices within the 18-second time window could be different because of the CSI sampling rate offset. To remove the sampling rate offset, we apply *nearest neighbour interpolation* to interpolate the feature vector sequence with lower sample rate, so we have the same number of features vectors inside two sequences spanning the same length of period.

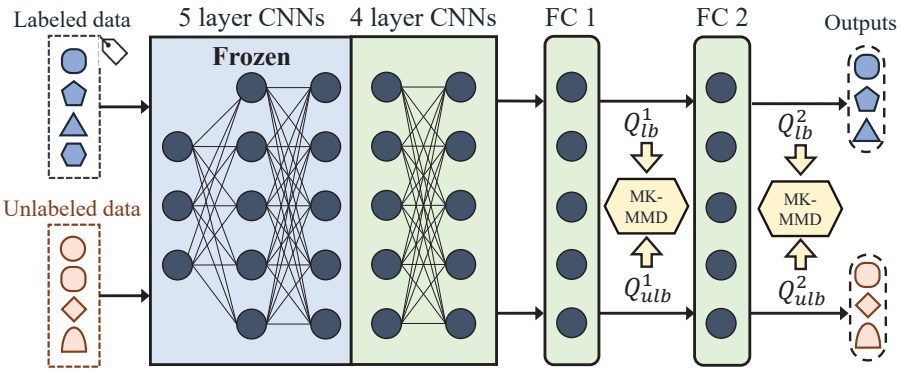


Fig. 10. The architecture of our DAN based network, which updates the model by taking both labeled and unlabeled data as input.

4.3.2 Multiple Overlapping Cells. We modify the voting scheme to handle multiple overlapping cells. Supposing two mobile users share n_c overlapping cells in their reported feature vectors, the voting results is given by:

$$\mathcal{P}_m = \sum_{t=1}^{k} \left(\sum_{i=1}^{n_c} w_i P_{t,i} \right), \tag{10}$$

where $P_{t,i}$ is the preliminary estimation results obtained from t-th feature vector of the i-th base station, and the weight w_i is used to adjust the impact of i-th base station on the final voting results. Different base station have different bandwidth and transmitting antennas, so feature vectors originated from different bandwidths contain different amount information. Higher weight should be given to base station with larger bandwidth and more antennas. In our experiments, we set w_i to 1 when the i-th base station has 20 MHz bandwidth and four antennas in its array, and then decrease the value of w_i proportional to the bandwidth and array size when the i-th base station adopts other configurations.

4.4 Continuous Model Updating

As the environment is highly dynamic, we need to continuously update our CNN based feature extractor after deployment. Completely retraining the feature extractor requires frequently recollecting fresh training data with ground truth and is thus unsustainable.

Instead of retraining, we propose to update our existing model using *deep adaptation networks* (DAN) [32]. Theoretically, the deep features gradually transition from general to specific along the neural network [32], *i.e.*, features extracted by the first several hidden layers are general and the features get more task or data specific when the layer gets deeper. Consequently, we apply DAN to freeze the first five hidden layers to extract general features and update the weight of the rest of the network to adapt to new environment, as shown in Figure 10.

We know from Figure 10 that updating the network demands two inputs: the labeled and the unlabeled training data. The label in our dataset is the proximity between two users, *i.e.*, close contact or not, so we can only collect labeled data by controlling the distance between users. When training the original network, we have collected huge amount of labeled data and we will reuse them here, so that Caper only requires unlabeled data in operation, once it is bootstrapped by the labeled data that we have already collected. The unlabeled data should be collected from the new environment in which the updated network should work. Collecting unlabeled data imposes no restrictions on the user behavior so we update the network with the data reported from all normal users from the

new environment, requiring no dedicated data collection process. That also means that the data reported by one user is not only used for close contact identification but also used for updating our network.

The data reported by a normal user is the features extracted by the original network, instead of the raw CSI that we require to update the network. Directly asking the users to report raw CSI would results privacy concerns. We observe that the original network and the updated network share the first five layer, so the intermediate results output by the first five layer is enough for both extracting feature vectors and updating the network. Therefore, we ask the user report such intermediate results instead of the final feature vector. By doing so, we protects the user privacy by hiding the raw CSI. We only select portion of our user (with their consent) to report intermediate results since such results are larger in data size comparing with feature vectors and thus incur larger communication overhead.

Updating the network. The labeled data we collected in the initial stage and unlabeled data we collected from the new environment results in different distributions in the features output by the fully connected layers at the end of the network. DAN computes the discrepancy of these two distributions using a *multi-kernel maximum mean discrepancies* (MK-MMD), and then add the discrepancy as a penalty at the loss function:

$$\mathcal{L}_{DAN} = \mathcal{L}_f + \lambda \cdot \sum_{\ell=l_1}^{l_2} M_{mk-mmd}\left(Q_{lb}^{\ell}, Q_{ulb}^{\ell}\right), \tag{11}$$

where λ is the penalty weight, it controls how much the distribution discrepancy penalty influences the whole loss, the l_1 and l_2 are the starting and final index of fully connected layers. Q_{lb}^{ℓ} and Q_{ulb}^{ℓ} represent the feature distribution at ℓ-th layer when feeding labeled and unlabeled data, respectively. The function M_{mk-mmd} computes the discrepancy between two distributions, which is defined in [32]. By adding the discrepancy penalty term, our model minimizes the contrastive loss, the AoA and TDoF estimation tasks prediction error, and the distribution discrepancy simultaneously during training. Our experimental results in Section 7.5 show that CAPER achieves an average accuracy of 91.38% in unseen environments by using the DAN for network updating.

4.5 Coverage Extension

Even though cellular networks has a pervasive coverage due to the ubiquitous cellular infrasturcture, there still exists challenging indoor environments, such as the basements or rooms with thick concrete walls, where the signal from cellular base stations are weak. To extend CAPER's coverage to such challenging environment, we propose to use Wi-Fi CSI as a complementary information source when the cellular CSI are not available or noisy. We reuse the feature extractor we trained for 20 MHz cells to extract features from CSIs measured from 20 MHz Wi-Fi channel. We note that, Wi-Fi has less subcarriers (50) in 20 MHz channel compared with cellular (1200), because its subcarrier is much wider than cellular's subcarrier [56]. To match input size required by CAPER's feature extractor, we perform interpolation [1] on WiFi CSI in frequency domain.

By default, CAPER relies on cellular CSI for contact tracing, if the *total effective bandwidth* $\mathcal{W}$ of all available base stations is larger than a threshold. We define the total effective bandwidth as:

$$\mathcal{W} = \sum_{i=1}^{n_{cell}} BW_i \times M_i, \tag{12}$$

where n_{cell} is the number of detected cells, and BW_i and M_i denote the bandwidth and number of antennas of the i th cell, respectively. When the effective bandwidth $\mathcal{W}$ falls below 30 MHz, CAPER starts to leverage Wi-Fi CSI for contact tracing. By utilizing the signal heterogenity, we demonstrate in Section 7.4 that CAPER achieve robust contact tracing performance across indoor and outdoor environments.

5 SECURITY AND PRIVACY CONSIDERATIONS

It is of great importance for a contact tracing system to preserve user privacy, since both the close contact information and location information are sensitive. In this section, we discuss the security and privacy considerations of CAPER.

Securing the Data at Server. To protect the user privacy, CAPER anonymizes the username (which can be used to identify a user), and cell information (which can be used to identify rough location of the user) by an authorized case investigators [52]. The data server only receives the anonymized user name and cell information. We note that the anonymization doesn't interfere with the process of contact tracing at the server side. The authorized case investigators de-anonymize the data to retrieve the user identity when and only it receives the information about close contacts of a positive case identified by the server. The data transfer between the client, the server and the authorized case investigators are encrypted using secure data transmission protocol, *e.g.* HTTPS.

Encrypting the Model. We adopt model encryption [50] and deploy encrypted model on client's mobile phone, so that the clients, including the malicious clients cannot access the model structure and weights. We also only store the encrypted model on the server, so the attacker can only get the encrypted model even after compromising the server.

5.1 Compromising the Server

In this section, we discuss how CAPER protect user's privacy even when the attacker compromises CAPER's server and obtain all the data we stored in the server. As shown in Figure 4, the server stores feature vectors reported by the users and the encrypted neural network model we trained.

Attack by CSI fingerprinting. One possible attack is via CSI fingerprinting. The attacker may collect the CSI at all possible locations, extract the feature vectors using the compromised encrypted feature extractor, and build the map between the physical location and the extracted feature vectors. According to the map, the attacker can then localize the user from the compromised feature vectors reported by the user. CAPER is able to protect users' privacy against such attack, since the CSI value is affected not only by the location of the user but also the surrounding environment, such as the walking human and the moving cars. The time-varying CSI make the fingerprinting based attack infeasible. CAPER is, however, robust to such changes, since it relies on the instantaneous CSI pairs measured at that specific time point when two user are close to each other to identify the close contact.

Attack by neural network model inversion. Another possible attack when attacker compromises CAPER's server is model inversion attacks against neural network, *i.e.,* inferring the inputs to the neural network model (CSI), from the corresponding model outputs (compromised feature vectors). There are basically two types of model inversion attack, *white-box model inversion* [17, 25, 33] and *black-box model inversion* [16, 25]. CAPER prevents the white-box model inversion by model encryption, since white-box model inversion requires knowing the structure and parameters of the model.

The black-box model inversion requires no prior knowledge of the structure or parameters of the neural network model. Existing work [38, 53, 62] has shown that differential private model is robust to the black-box model inversion attack. [18] proves that neural network models trained on wireless signals is differential private since the wireless signals data inherently contains additive Gaussian noise. Our feature extractor trained from wireless CSI data, which inherently contains additive Gaussian noise, preserves differential privacy, and thus is also robust to the black-box model inversion attack.

6 IMPLEMENTATION

Mobile client side. Accessing physical layer CSI data on smartphones requires customization of the cellular firmware. As a proof of concept, we implement the CRS decoding and CSI extraction parts on USRP X310 and B210 radios by modifying from srsLTE [19], which is an open-source LTE library. We use two laptops, each connecting with two USRPs, to emulate two mobile clients with two radio chains. Each mobile client extracts the feature vectors from the measured CSI, tags the feature vector with timestamps and subframe index, and at last uploads the feature vectors together with the tag to the server. The server aligns the feature vectors from multiple users according to their timestamp and subframe index at millisecond granularity.

Server side. We use PyTorch [41] to train our feature vector on a server, where the CPU is Intel i7-9700, and the GPU is Geforce RTX 2060. We set the initial learning rate to 0.01, and decrease the learning rate by a factor of 0.7 every 5 epochs to stabilize the training. We set the hyperparameters $m = 2$, $\beta_A = 0.1$, $\beta_T = 0.1$, and $\lambda = 0.2$ during the training. The server distributes the CNN based feature extractor to mobile clients, after finishing the training. To identify close contacts, the server sets the voting time span to 18s and the prediction threshold D_{threh} to 0.9. We update the model everyday, with fresh user reported data.

7 EVALUATION

In this section, we evaluate the performance of CAPER. We first introduce our evaluation methodology, and then give the end-to-end performance under diverse physical layer configurations. After the end-to-end evaluation, we compare CAPER with the widely used Bluetooth Low Energy (BLE) based approach [11, 20, 46] (Section 4.5). We then present CAPER's generalization ability across time and on different locations, followed by a micro-benchmark evaluating the accuracy gains arising from each component of CAPER. We also demonstrate the resource usage of CAPER on mobile phones at the end of this section.

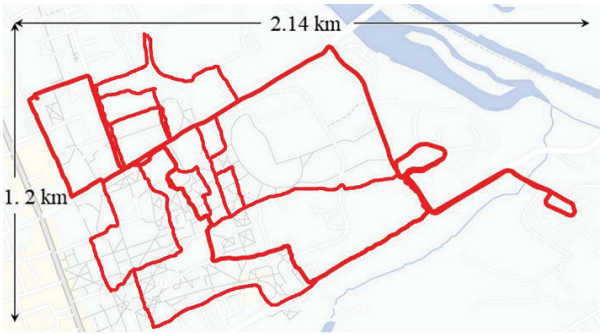


Fig. 11. The route of our outdoor data collection. In total, we cover a 2.6 km^2 area.

7.1 Methodology

Our system is intrinsically estimating proximity by comparing data pairs. A comprehensive and diverse dataset of CSI pairs subsume the data required to a real system. We collect CSI pairs by moving two mobile clients together, where each client comprises two USRPs. To get the ground truth of proximity between clients, we fix the distance between these two mobile clients to 0.25 meter, 0.75 meter, 1.5 meters, 2.25 meters, 3 meters, and larger than 3 meters. The first three distances are smaller than six feet and thus labeled as "close contact". The later three distances are labeled as "not close contact". We also include cases, where two radios are 1m away from each other but have a wall in between, into our dataset. Since in such cases, the two user don't have close contacts in the

context of Covid-19 contact tracing, we label them as "not close contact". During the data collection, we shuffle the combination of USRPs that each client is consisted of to emulate mobile clients with different RF hardware. The WiFi CSI are measured using USRPs at the same indoor locations of the cellular CSI experiments.

For indoor cases, we collect our data at 22 different locations of four buildings, covering both office environment and home environment, over a period of 16 days, covering both daytime when there are human moving around and idle late night. At each location, we move two mobile clients randomly inside an area that covers around 1,600 square feet. We note that the data collected from these indoor locations do not enjoy the line of sight with the cell towers. In total we collect data from 6 nearby cell towers owned by three mobile network operators, Verizon, AT&T and T-mobile, whose operating frequencies of the cells range from 1805MHz up to 2355MHz.

As for outdoor experiments, we collect our data by moving our clients over a 2.6 km^2 area on campus, over a period of 6 days, as shown in Figure 11. These data are collected from 3 cell towers, whose operating frequencies of the cells range from 1940MHz up to 2355MHz. The distance from our clients to the cell towers ranges from 100 meters up to 800 meters. Since our data collection points distribute over a large area, our data includes diverse signal propagation scenarios, include both LoS and NLoS. We also observe diverse obstacles between the base sation and the client, including buildings, cars and trees. This outdoor dataset also covers data with different levels of crowd densities. We perform outdoor data collection at both busy hours where there are moving vehicles and human beings around, and late night where there are almost no other people around.

The cell towers we measured cover all antenna configurations, *i.e.*, one, two and four transmitting antennas in the array. As for bandwidth, we only observe 5MHz, 10MHz and 20MHz cells in practice, although the 3GPP specification also outlines 1.4MHz, 3MHz, 15MHz as supported bandwidth. Hence, in our experiments, we only show system results under the three observed bandwidths. At the client side, our mobile client has two antennas in its array. In total, we collect 6,212,008 CSI pairs. We train our feature extractor using the indoor data we collected from the first six days and outdoor data from the first four days, and then evaluate the system performance using indoor data collected from the later ten days and outdoor data collected from the later two days.

To reduce the data collection overhead, we emulate the users with different DRX cycles and CSI sampling time misalignments from densely sampled CSI. Specifically, we first collect the CSI at a high frequency, *i.e.*, every 10 subframes, and then downsample the CSI to create user traces that have varying DRX cycles, *i.e.*, 0.32 second, 0.64 second and 1.28 seconds, and CSI sampling time misalignments, *i.e.*, from 20 milliseconds to 640 milliseconds with a 20 milliseconds interval.

7.2 Close Contacts Identification Accuracy

In this section, we evaluate CAPER's accuracy in identifying close contacts. We begin with the the end-to-end accuracy, including results in both outdoor and indoor scenario and both busy hour and idle hours, followed by the evaluation of the impact of various physical layer configurations.

Methodology. We use CAPER's feature extractor to extract a pair of feature vectors from each collected CSI pair, and run CAPER's close contact identifier by taking the extracted feature vector pair as input to determine whether the CSI pair is collected from two devices that have close contacts with each other. We compare the identification results with the ground truth to calculate the accuracy. Except the overall accuracy, we also calculate the true positive (TP), *i.e.*, the correct close contact discovery rate, the true negative (TN), *i.e.*, the correct far contact identification rate, the false negative (FN), *i.e.*, the close contact missing rate, and the false positive (FP), *i.e.*, the false alarm rate.

7.2.1 Identification Accuracy. We run CAPER on all testing CSI pairs and calculate the accuracy. We give the confusion matrix of CAPER's identification accuracy in Table 1. from which we see that the overall accuracy of CAPER, is $TP + TN = 93.39\%$. In addition, in the context of COVID-19 contact tracing, missing close contacts is a

Table 1. The confusion matrix result of CAPER

Ground truth		Close	Far
Predictions	Close	TP = 49.12%	FP = 5.40%
	Far	FN = 1.21%	TN = 44.27%

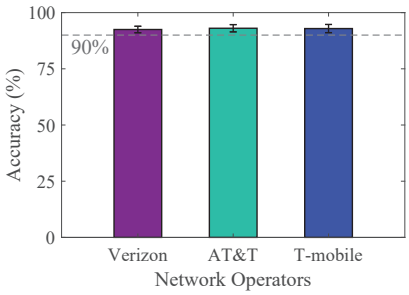


Fig. 12. Overall accuracy of CAPER working with data collected from different network service providers.

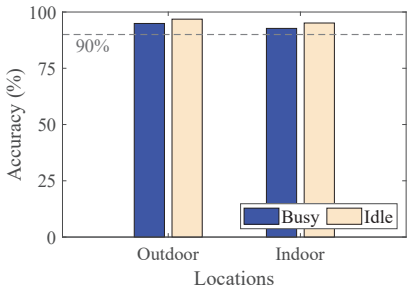


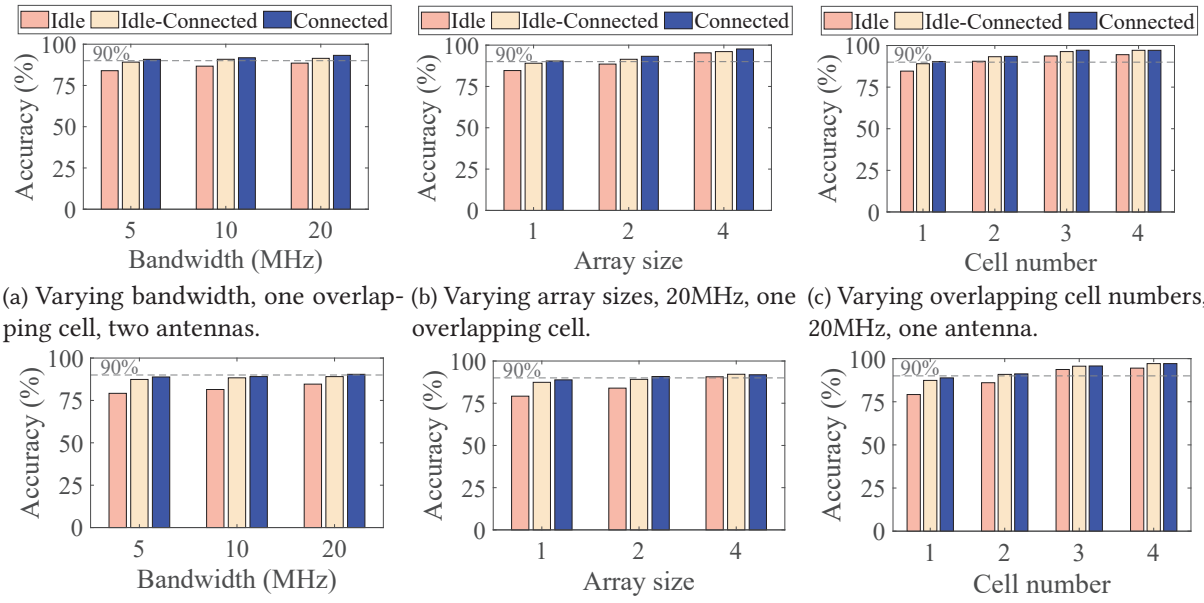
Fig. 13. Overall accuracy of CAPER on data collected from outdoor or indoor locations and in busy or idle hours.

much more severe problem than giving false alarms. From Table 1, we see that even though CAPER gives false alarms to 5.4% of CSI pairs, it only misses 1.21% of close contacts in the dataset.

We provide CAPER's identification accuracy with CSI pairs measured from cell towers of different network operators in Figure 12, where the error bar represents the standard deviation of accuracy of different cells towers from the same operator. We observe that CAPER achieves a high average accuracy working with data collected from different network operators. Furthermore, the small deviation indicates that CAPER is stable across different cells from three network operators.

To show how CAPER's performance varies depending on different locations and time, we give the overall accuracy of CAPER on data collected from both outdoor and indoor locations and in busy or idle hours in Figure 13, where the legend idle means the accuracy on data collected at late night. We note that the cell towers we measured in the outdoor experiment are all with 20MHz bandwidth. Hence, to perform a fair comparison between outdoor and indoor performance, in Figure 13 we only show the indoor performance with CSI pairs collected from 20MHz cell towers. We do a detailed discussion on the impact of diverse cell tower physical layer configurations in Section 7.2.2. We draw the conclusion from this figure that CAPER achieves a higher accuracy in outdoor locations, which is due to the better channel quality in outdoors. We also observe that CAPER performs better in late night than in busy hour. We attribute this higher accuracy to less interference from other LTE clients at late night. The highest achieved accuracy is 96.83% at outdoor locations and at late night.

7.2.2 Impact of Diverse Physical Layer Configurations. In this section, we evaluate the impact of diverse physical layer configurations on the end-to-end accuracy. The configurations we investigate include channel bandwidth, array size of the cell tower, the number of overlapping cells and the DRX cycle. We note that, two mobile devices may be configured with different DRX cycle when measuring the CSI. For demonstration purpose, we examine the combination of minimum and maximum DRX cycle, in this section. Specifically, when the device is in RRC connected mode, the DRX cycle can be treated as zero. When the device is in RRC idle mode, the maximum DRX



(a) Varying bandwidth, one overlapping cell, two antennas.
(b) Varying array sizes, 20MHz, one overlapping cell.
(c) Varying overlapping cell numbers, 20MHz, one antenna.
(d) Varying bandwidth, one overlapping cell, one antenna.
(e) Varying array sizes, 5MHz, one overlapping cell.
(f) Varying overlapping cell numbers, 5MHz, one antenna.

Fig. 14. CAPER's performance under diverse physical layer configurations. We examine the impact of bandwidth in **(a)** and **(d)**; the impact of array size in **(b)** and **(e)**; and the impact of the number of overlapping cells in **(c)** and **(f)**.

cycle is 1.28 seconds. In total, we have three combinations with two mobile devices, *i.e.*, idle-idle, idle-connected and connected-connected.

Impact of bandwidth. To evaluate the impact of bandwidth on the identification accuracy, we fix the number of overlapping cells to one and the antenna number in the array of the overlapping cell to one and two, and plot the accuracy of close contact identification in Figure 14(d) and 14(a). We can observe from these two figures that CAPER achieves higher accuracy when the bandwidth becomes larger and the highest achieved accuracy is 93.24% with 20 MHz bandwidth and two antennas. Even in the worst case, where two devices are in idle mode, *i.e.*, each CSI sample are 1.28 seconds from each other, and share one overlapping cell that has 5 MHz bandwidth and one antenna in its array, CAPER can still achieve an accuracy of 79.19%.

Impact of array size. To evaluate the impact of array size of the cell tower on the identification accuracy, we fix the number of overlapping cells to one and the bandwidth to 5 MHz and 20 MHz, and plot the accuracy of close contact identification in Figure 14(e) and 14(b). We see that increasing the array size significantly improve the accuracy. When the two devices are in connected mode with a configuration of 20MHz, one overlapping cell, four antennas, CAPER can achieve a accuracy of 97.67% on our testing dataset.

Impact of overlapping cells. To evaluate the impact of the number of overlapping cells on the identification accuracy, we fix the bandwidth to 5 MHz and 20 MHz, and array size to one antenna, and plot the accuracy of close contact identification in Figure 14(f) and 14(c). We note that, since each mobile user in our implementation has only two radio chains (two USRPs due to limited available hardware), so we emulate the four cell cases by

concatenating two traces. We observe higher accuracy when two devices have more overlapping cells. Specifically, the highest accuracy is 97.12% when two device are in connected mode and share four cells.

Impact of DRX cycle. From all six figures, we could see that smaller DRX cycles significantly improve the accuracy when the bandwidth, array size and the number of overlapping cells is small. For example, the accuracy improves from 79.19% to 88.84% when two users, which share a cell with 5 MHz and one antenna, changes from idle mode to connected mode. The accuracy improvement becomes marginal with increasing bandwidth, array size and number of overlapping cells, and thus more information from CSI.

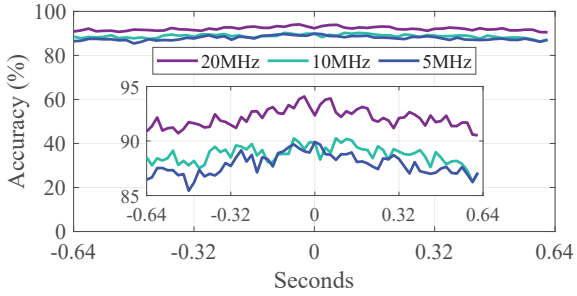


Fig. 15. CAPER's close contact identification accuracy with data that has varying time misalignment.

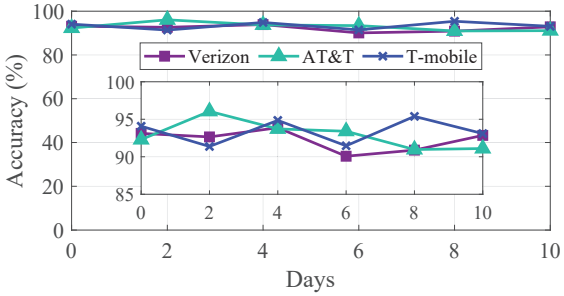


Fig. 16. CAPER's close contact identification accuracy across time, with data collected from four service providers.

7.2.3 Impact of Time Misalignment. As we mention in Section 4.2.3 that two devices in idle mode may wake up and measure CSI at misaligned subframes. We plot the accuracy with CSI pairs sampled with a time misalignment from -0.64 seconds up to 0.64 seconds, in Figure 15. In this figure, the testing samples contain configurations of one to four transmitting antennas and one to four overlapping cells. From this figure we see that the accuracy is high for three different bandwidths. When we zoom in the curves in the inner figure, we can see a slight drop of the accuracy when time misalignment increases. When the time misalignment is the maximum 0.64 seconds, CAPER can still achieve an accuracy of 90.71% with 20MHz bandwidth, and an accuracy of 87.80% and 86.78% with 10MHz and 5MHz bandwidth.

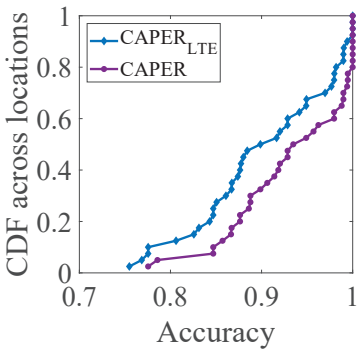


Fig. 17. The distribution of accuracy of $\text{CAPER}_{\text{LTE}}$ and CAPER across 40 locations in the basement floor.

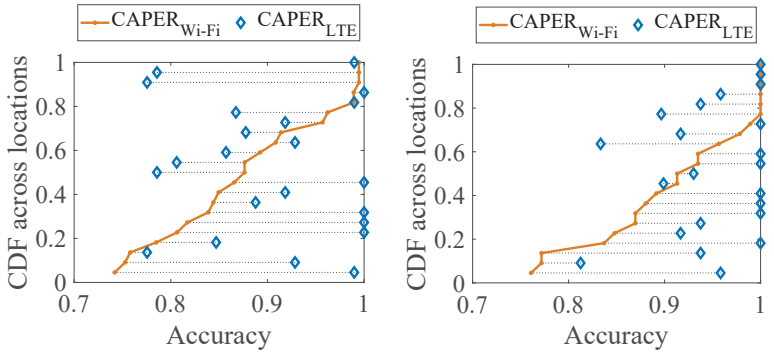


Fig. 18. The distribution of accuracy of $\text{CAPER}_{\text{Wi-Fi}}$ and $\text{CAPER}_{\text{LTE}}$ across 25 indoor locations.

7.3 Coverage Extension Using Wi-Fi

In this section, we verify the coverage extension of leveraging Wi-Fi as the complementary data source. We conduct experiments in the basement floor of a campus building, where cellular signal becomes weak but the campus Wi-Fi still covers the whole floor. We perform close contact identification at 40 locations using CAPER and record the accuracy. We then repeat the experiments but disable the functionality of leveraging Wi-Fi CSI as complementary data source when cellular signal is too weak. We denote the version of CAPER that merely uses cellular signal as $\text{CAPER}_{\text{LTE}}$. We plot the CDF of the accuracy across locations in Figure 17, from which we could see that the average accuracy of CAPER and $\text{CAPER}_{\text{LTE}}$ are 93.47% and 90.49%, respectively. In the challenging basement floor, the signal of many cellular base stations, especially those with high central frequency, cannot penetrate the wall, resulting in less available high quality cells for $\text{CAPER}_{\text{LTE}}$. CAPER outperforms in such challenging environment by leveraging the strong Wi-Fi signal.

To further evaluate the performance CAPER using different signals, we implement another version of CAPER that only use Wi-Fi CSI as input, which we denote as $\text{CAPER}_{\text{Wi-Fi}}$. We pick 22 indoor locations that simultaneously have two Wi-Fi AP (20MHz channel) and also two 20 MHz cellular cells, and test the performance of both $\text{CAPER}_{\text{Wi-Fi}}$ and $\text{CAPER}_{\text{LTE}}$ at each location. We plot the CDF of accuracy across locations in Figure 18. Specifically, in Figure 18(a), we only use one AP and one cell for contact tracing, while in Figure 18(b) we use both APs and both cells. We see from Figure 18(a) and Figure 18(b) that $\text{CAPER}_{\text{Wi-Fi}}$ has an overall accuracy of 89.76%, which is comparable to $\text{CAPER}_{\text{LTE}}$'s accuracy 90.75%. Even though LTE has a well-organized frame structure, which leads to uniformly distributed and finer-grained LTE CSI data than Wi-Fi CSI, there are still some locations where $\text{CAPER}_{\text{Wi-Fi}}$ performs better than $\text{CAPER}_{\text{LTE}}$. We attribute this to the weaker signal strength of LTE in some indoor scenarios. Such phenomenon is alleviated with more observed cellular cells, just as shown in Figure 18(b).

7.4 Comparison with BLE Based Approach.

In this section, we compare CAPER's performance with BLE based approach. Many BLE based approaches [11, 20, 46] have been proposed by the industry and the research community. We compare with CovidSafe since its source code is open-sourced[10].

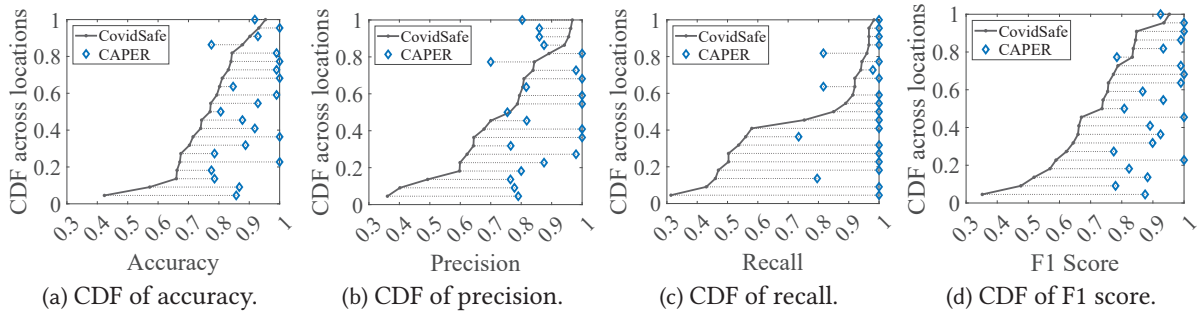
7.4.1 Methodology. BLE based approach requires two BLE devices to transmit signal between each other and identifies close contact by checking the RSSI of received signal— RSSI higher than a threshold indicating close contact. We therefore let two cellphones, one iPhone X and one Xiaomi MI 9, transmit signal at different distances, varying from 0.25 meter to larger than 3 meters, and then record the RSSI. For each distance, we repeat the experiment multiple times at 360 different nearby locations. To provide a head-to-head comparison between CAPER and CovidSafe, during our experiments, we attach these two cellphones to two USRPs and simultaneously collect the cellular CSI. We denote the data collection at six different distances, each repeating at 360 different locations, as *one group of experiments*. We conduct 42 groups of experiments at 22 indoor locations and 20 outdoor locations.

When comparing with the BLE based approach, it is important to have a comprehensive understand of the results. Hence, for each group of experiments, we evaluate the overall accuracy, precision, recall and F1 score of the contact tracing, where precision and recall capture the impact of false positives and false negatives, respectively. Specifically, we calculate the precision as: $Precision = \frac{TP}{TP+FP}$ and recall as: $Recall = \frac{TP}{TP+FN}$. Based on the calculated precision and recall, we calculate the F1 score as:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{13}$$

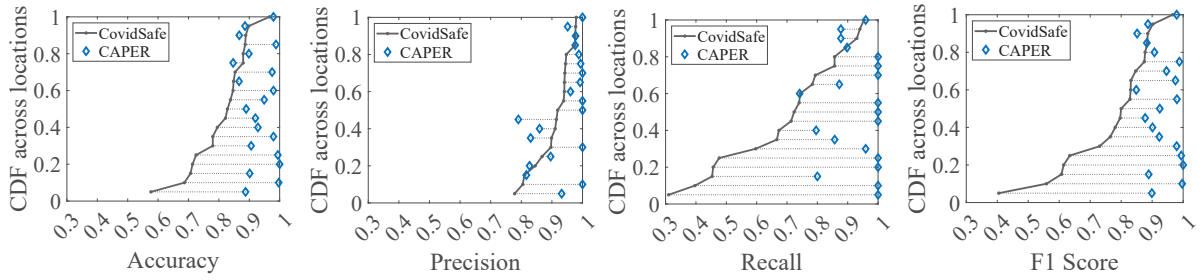
7.4.2 Results. We plot the CDF of CovidSafe's accuracy, precision, recall and F1 score across 22 indoor and 20 outdoor locations in Figure 19 and 20, respectively. Each point on the CDF curve represents the results of one group of experiments at a specific location. The blue markers depicts the corresponding results achieved by CAPER at the same location. We see from Figure 19 and 20 that CAPER outperforms CovidSafe at 37 out of 42 locations we tested, demonstrating CAPER's robustness across diverse environments. The average accuracy of CovidSafe is only 78.43%. CAPER, however, achieves an accuracy of 91.84%, a 13.41% improvement over CovidSafe.

By comparing Figure 19(b) and 19(c), we can see that both CAPER and BLE approach achieve a higher accuracy at outdoor locations than at indoor locations, due to less multipath effects and less interference in outdoor scenarios. Precision and recall are affected by false alarm rate and close contact missing rate, respectively. We notice that CAPER has a pretty high recall value (nearly 1) at most locations. A high recall value indicates that CAPER attaches more importance to lowering missing close contacts than lowering false alarms, which aligns with the the goal of COVID-19 contact tracing. The BLE approach, on the other hand, has a low average recall value of 0.70. The precision of BLE approach is 0.81, which is lower than CAPER's precision 0.90. F1 score is often used when the FN and FP are crucial, while accuracy is used when the TP and TN are more important. In Figure 19(d), CAPER has a higher F1 score than BLE at most of the locations, which means CAPER has less incorrectly classified cases than BLE approach.



(a) CDF of accuracy. (b) CDF of precision. (c) CDF of recall. (d) CDF of F1 score.

Fig. 19. The distribution of accuracy (a), precision (b), recall (c), and F1 score (d), of CovidSafe, CAPER-WiFi, and CAPER-LTE, across 22 indoor locations.



(a) CDF of accuracy. (b) CDF of precision. (c) CDF of recall. (d) CDF of F1 score.

Fig. 20. The distribution of accuracy (a), precision (b), recall (c), and F1 score (d), of CovidSafe, and CAPER-LTE, across 20 outdoor locations.

7.5 Generalizability

We evaluate the performance of CAPER's when adapting to new environments using the DAN in this subsection. We first present the evaluation of CAPER's performance at unseen locations. We then investigate whether CAPER is able to handle environment dynamics.

7.5.1 Generalizability to Unseen Locations. We divide the data we collected from indoor locations into two groups: one group includes data from 12 randomly selected locations and another group includes data from the rest ten locations. We train our feature extractor using the data collected from the 12 locations in the first group and test the close contact identification accuracy with the data from the rest ten locations, which are grouped into four categories, corridor, lobby, classroom, and Lab.

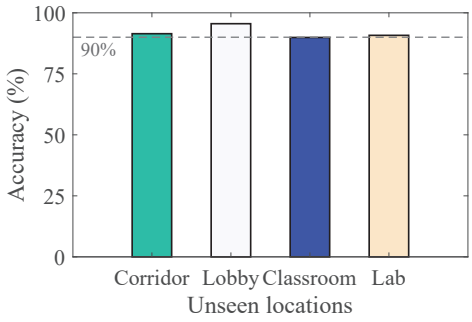


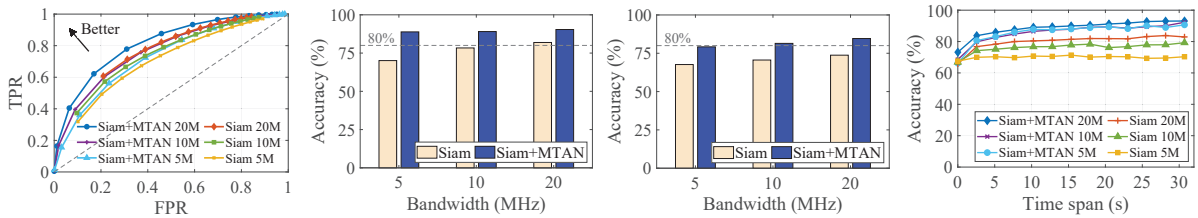
Fig. 21. Overall accuracy of CAPER working with data collected from four unseen locations.

We plot the close contact identification accuracy in Figure 21. We conclude that CAPER generalizes well on data collected from unseen locations, with a close contact identification accuracy ranging from 89.90% to 95.54% over the four kinds of unseen locations. We also note that the accuracy variance is attributed to the physical layer configurations of the associated base stations at those four locations. For example, in one classroom, we observe three base stations, one with 5 MHz and two with 10 MHz channel bandwidth, while in one lobby, we detect another three base stations, one with 20 MHz and two with 10 MHz channel bandwidth.

7.5.2 Adaption to Environmental Dynamics. We train our model using data we collected in the first four days of a 14-day data set. We then continuously update the trained feature extractor using the algorithm we introduce in Section 4.4 to adapt our model in the later ten days. We evaluate CAPER's adaptation capability over time by testing our adapted feature extractor on unseen data from the later ten days and plot the close contact identification accuracy in Figure 16. In this figure, the data point at day zero shows the close contact identification accuracy on data samples collected from the first six days, and then the data points at day two to day 10 shows the adapted model's accuracy on the following ten days. From this figure we observe that, CAPER maintains a high accuracy over the 10-day period by applying our continuous model updating algorithm. The mean accuracy over ten days is 92.79% and the accuracy at the tenth day is 92.35% .

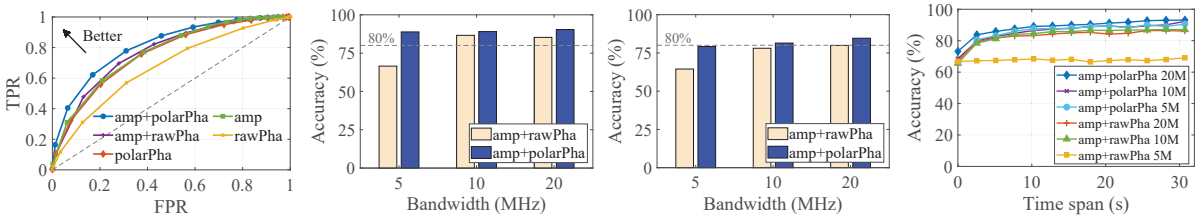
7.6 Micro-benchmark

In this section, we quantify the impact of different system components on the accuracy of close contact identification, specifically the impact of having a MTAN module, and the impact of our proposed CSI pre-processing technique.



(a) ROC curves. (b) Accuracy (connected). (c) Accuracy (idle). (d) Varying time spans.

Fig. 22. Micro-benchmarks comparing Siamese+MTAN with only Siamese. The ROC curve of two models trained using Siamese plus MTAN, and purely Siamese is plotted in **(a)**. The identification accuracy on CSI measured from devices in idle and busy model are given in **(b)** and **(c)**. The accuracy with varying time span is in **(d)**.



(a) ROC curves. (b) Accuracy (connected). (c) Accuracy (idle). (d) Varying time spans.

Fig. 23. Micro-benchmarks comparing different pre-processed CSI. The ROC curve of models is plotted in **(a)**. The identification accuracy for idle and busy devices are given in **(b)** and **(c)**. The accuracy with varying time span is in **(d)**.

7.6.1 Impact of Siamese Network and MTAN. We demonstrate that our MTAN based training network solves the overfitting problem introduced by training using purely Siamese network. We train multiple mode by varying the threshold D_{threh} in Eqn. 8, and plot the ROC curve [55] in Figure 22(a), where the x-axis is false positive rate (FPR), and y-axis is true positive rate (TPR):

$$FPR = \frac{FP}{FP + TN}, TPR = \frac{TP}{TP + FN}.$$

A curve resides in the upper left corner means the model having a strong capability of identifying close contact, and the dashed diagonal line means random guess. From Figure 22(a), we see that the models trained using MTAN based architecture performs better, since these models are able to extract features that are related to the physical locations of the mobile devices and thus has stronger capability of identifying close contacts.

To further quantify the improvement, we test the identification accuracy of two models trained using Siamese and MTAN network on CSI pairs with different cell tower configurations, specifically the data are measured from cells that have one antenna in its array but have varying channel bandwidth, including 5MHz, 10MHz and 20MHz. We plot the close contact accuracy obtained from CSI pairs measured when the two devices are in connected and idle mode, in Figure 22(b) and 22(c), respectively. We see that MTAN based network improves the accuracy by 18.75% and 11.54%, when the cell have 5 MHz bandwidth and the user is in connected mode and idle mode, respectively. Such an improvement decreases with increasing bandwidth, since wider bandwidth provides richer information for classification which boosts the base line accuracy. We also plot the accuracy with varying voting time span in Figure 22(d), from which we could clearly see an accuracy improvement when the model is trained with both MTAN and Siamese network.

7.6.2 Impact of the CSI Pre-processing. We investigate the contribution of our CSI pre-processing to the final accuracy. Specifically, we train five models with five differently pre-processed CSI, including the CSI amplitude plus polar phase, amplitude plus raw phase, purely amplitude, purely polar phase, and purely raw phase. We test the accuracy of these models and plot the ROC curve in Figure 23(a), from which we could see that the model trained using CSI amplitude plus our polar phase resides at the top left corner, meaning the best performance. We also quantify the improvement in Figure 23(b), 23(c) and 23(d), from which we see a clear accuracy improvement from the polar phase we input to the neural network.

Table 2. System resources usage of CAPER on mobile phones.

	CPU	Run time	energy	Memory	Storage	Model size
OnePlus 8	20.47%	12.1ms	26.54mAh	200.0MB	65MB	2.75MB
Xiaomi 9	17.92%	14.4ms	34.02mAh	238.5MB	59MB	2.75MB

7.7 Resources Usage

In this section, we investigate the computational overhead of our neural network based feature extractor and verifies its feasibility when deployed on commercial mobile phones. We implement CAPER's neural network based feature extractor on mobile phone using Pytorch Mobile [42][3].

We evaluate the system overhead of CAPER's feature extractor by measuring the CPU usage, memory usage, power consumption and required storage of two mobile phones, *e.g.*, one OnePlus 8 released in 2020 and one Xiaomi 9 released in 2019, when running the extractor. Specifically, we measure the CPU and memory usage using Android Debug Bridge [3], and the energy consumption using the Accubattery [14]. We plot the results in Table 2. We see that CAPER requires 200.0 MB and 238.5MB RAM which is only 1.7% and 4.5% of the total RAM of OnePlus 8 and Xiaomi 9, respectively. When running, CAPER occupies 20.47% and 17.92% of CPU on OnePlus 8 and Xiaomi 9, respectively. It finishes the feature extraction within 12.1 microseconds and 14.4ms for 95% of the cases on these two phones. We could also see that the final model size is only 2.75MB. The total required storage of our app is around 60MB for both phones. Table 2 also gives the average energy consumption. We run our CAPER for ten minutes on the tested cellphones, and the average consumption on the two phones are 26.54mAh and 34.02mAh, respectively. For OnePlus 8, its 4300mAh battery can support CAPER for 27 hours if only CAPER drains its energy.

8 CONCLUSION

We propose a cellular-assisted, deep learning based contact tracing system for containing the spread of COVID-19. We leverage a deep learning based feature extractor to map CSI into one point inside a high-dimensional feature space, which preserves user privacy and achieves high accuracy in identifying close contacts. CSI data used in CAPER provide much richer information than RSSI, including location information, and surrounding environment, making it a better input for proximity estimation. Experimental results prove our argument, where CAPER achieves an overall accuracy of 93.39% in identifying close contacts, which is 14.96% higher than the accuracy of the BLE based approach.

[3]Currently, we are able to deploy the feature extractor on mobile phones, but we are not able to extract the CSI from the cellular modem in real-time, since accessing the physical-layer CSI requires modification of the firmware of cellular modem, which is proprietary to the manufacturer, like Qualcomm.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback that has improved the quality of this paper significantly, and Zhao Song and Yangsibo Huang for their insightful discussions. This work is supported by an award from the Office of the Princeton University Dean for Research. This material is based upon work supported by NSF grant CNS-2223556, and the RAPID program of NSF under Grant No. CNS-2027647. We gratefully acknowledge a gift from the Amateur Radio Digital Communications Foundation.

REFERENCES

[1] 3GPP TS 36.141. 2018. Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) Conformance Testing. [3gpp.org].
[2] Nadeem Ahmed, Regio A Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K Jha. 2020. A Survey of COVID-19 Contact Tracing Apps. *IEEE Access* 8 (2020), 134577–134601.
[3] Android Team 2021. Android Debug Bridge. https://developer.android.com/studio/command-line/adb.
[4] Roshan Ayyalasomayajula, Aditya Arun, Chenfeng Wu, Sanatan Sharma, Abhishek Rajkumar Sethi, Deepak Vasisht, and Dinesh Bharadia. 2020. Deep learning based wireless localization for indoor navigation. In *ACM MobiCom*.
[5] Roshan Ayyalasomayajula, Deepak Vasisht, and Dinesh Bharadia. 2018. BLoc: CSI-Based Accurate Localization for BLE Tags. In *ACM CoNEXT*.
[6] Christopher M Bishop et al. 1995. *Neural networks for pattern recognition*. Oxford University Press.
[7] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In *NeurIIPS*.
[8] CDC. 2019. Contact tracing slows the spread of COVID-19. [cdc.gov].
[9] CDC. 2019. Digital Contact Tracing Tools. [cdc.gov].
[10] Justin Chan et al. 2020. COVIDSafe APP. [github.com].
[11] Justin Chan, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob Sunshine, et al. 2020. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544* (2020).
[12] Michael Crawshaw. 2020. Multi-Task Learning with Deep Neural Networks: A Survey. arXiv:2009.09796
[13] J. A. del Peral-Rosado, R. Raulefs, J. A. López-Salcedo, and G. Seco-Granados. 2018. Survey of Cellular Mobile Radio Localization Methods: From 1G to 5G. *IEEE Communications Surveys Tutorials* 20, 2 (2018), 1124–1148.
[14] Digibites 2019. AccuBattery. [accubatteryapp.com].
[15] Mikhail Dmitrienko, Abhishek Singh, Patrick Erichsen, and Ramesh Raskar. 2020. Proximity Inference with Wifi-Colocation during the COVID-19 Pandemic. *arXiv preprint arXiv:2009.12699* (2020).
[16] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *CVPR*. 4829–4837.
[17] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An {End-to-End} Case Study of Personalized Warfarin Dosing. In *USENIX Security*. 17–32.
[18] Akshay Gadre, Fan Yi, Anthony Rowe, Bob Iannucci, and Swarun Kumar. 2020. Quick (and dirty) aggregate queries on low-power wans. In *IPSN*. 277–288.
[19] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. 2016. srsLTE: an open-source platform for LTE evolution and experimentation. In *ACM WiNTECH*.
[20] Google and Apple. 2020. Exposure Notification Bluetooth Specification. [blog.google].
[21] GPS. 2021. GPS localization accuracy outdoor. [gps.gov].
[22] Unsoo Ha, Junshan Leng, Alaa Khaddaj, and Fadel Adib. 2020. Food and Liquid Sensing in Practical Environments using RFIDs. In *USENIX NSDI*.
[23] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *IEEE CVPR*.
[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE CVPR*.
[25] Zecheng He, Tianwei Zhang, and Ruby B Lee. 2019. Model inversion attacks against collaborative inference. In *ACSAC*. 148–162.
[26] R. Jia, M. Jin, Z. Chen, and C. J. Spanos. 2015. SoundLoc: Accurate room-level indoor localization using acoustic signatures. In *IEEE CASE*.
[27] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. Spotfi: Decimeter level localization using wifi. In *ACM SIGCOMM*.
[28] Patrick Lazik, Niranjini Rajagopal, Oliver Shih, Bruno Sinopoli, and Anthony Rowe. 2015. ALPS: A Bluetooth and Ultrasound Platform for Mapping and Localization. In *ACM SenSys*.
[29] Jinfeng Li and Xinyi Guo. 2020. COVID-19 Contact-tracing Apps: A Survey on the Global Deployment and Challenges. *arXiv preprint arXiv:2005.03599* (2020).

[30] Liqun Li, Pan Hu, Chunyi Peng, Guobin Shen, and Feng Zhao. 2014. Epsilon: A Visible Light Based Positioning System. In *USENIX NSDI*.
[31] Shikun Liu, Edward Johns, and Andrew J Davison. 2019. End-to-end multi-task learning with attention. In *IEEE CVPR*.
[32] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *ICML*.
[33] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *CVPR*. 5188–5196.
[34] Wenguang Mao, Mei Wang, Wei Sun, Lili Qiu, Swadhin Pradhan, and Yi-Chao Chen. 2019. RNN-Based Room Scale Hand Motion Tracking. In *ACM MobiCom*.
[35] R. Momose, T. Nitta, M. Yanagisawa, and N. Togawa. 2017. An accurate indoor positioning algorithm using particle filter based on the proximity of bluetooth beacons. In *IEEE GCCE*.
[36] A. Montanari, S. Nawaz, C. Mascolo, and K. Sailer. 2017. A Study of Bluetooth Low Energy performance for human proximity detection in the workplace. In *IEEE PerCom*.
[37] Tomoya Nakatani, Takuya Maekawa, Masumi Shirakawa, and Takahiro Hara. 2018. Estimating the physical distance between two locations with wi-fi received signal strength information using obstacle-aware approach. *ACM UbiComp* (2018).
[38] Cheolhee Park, Dowon Hong, and Changho Seo. 2019. An attack-based evaluation method for differentially private learning against model inversion attack. *IEEE Access* 7 (2019), 124988–124999.
[39] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan. 2019. Design of a Hybrid RF Fingerprint Extraction and Device Classification Scheme. *IEEE IoT-J* (2019), 349–360.
[40] Timothy J Pierson, Travis Peters, Ronald Peterson, and David Kotz. 2019. Proximity detection with single-antenna IoT devices. In *ACM MobiCom*.
[41] PyTorch Team 2016. PyTorch. [pytorch.org].
[42] PyTorch Team 2016. PyTorch Mobile. [pytorch.org].
[43] Ramesh Raskar, Greg Nadeau, John Werner, Rachel Barbar, Ashley Mehra, Gabriel Harp, Markus Leopoldseder, Bryan Wilson, Derrick Flakoll, Praneeth Vepakomma, et al. 2020. COVID-19 Contact-Tracing Mobile Apps: Evaluation and Assessment for Decision Makers. *arXiv preprint arXiv:2006.05812* (2020).
[44] Ramesh Raskar, Abhishek Singh, Sam Zimmerman, and S Kanaparti. 2020. Adding Location and Global context to the Google/Apple Exposure Notification Bluetooth API. *arXiv preprint arXiv:2007.02317* (2020).
[45] Leonie Reichert, Samuel Brack, and Björn Scheuermann. 2020. Privacy-Preserving Contact Tracing of COVID-19 Patients. *IACR Cryptol. ePrint Arch.* (2020), 375.
[46] Ronald L Rivest, Jon Callas, Ran Canetti, Kevin Esvelt, Daniel Kahn Gillmor, Yael Tauman Kalai, Anna Lysyanskaya, Adam Norige, Ramesh Raskar, Adi Shamir, et al. 2020. The PACT protocol specification. *Private Automated Contact Tracing Team, MIT, Cambridge, MA, USA, Tech. Rep. 0.1* (2020).
[47] Piotr Sapiezynski, Arkadiusz Stopczynski, David Kofoed Wind, Jure Leskovec, and Sune Lehmann. 2017. Inferring person-to-person proximity using wifi signals. *ACM UbiComp* (2017).
[48] Warren S. Sarle. 2002. Neural-nets FAQ. [faqs.org].
[49] Sheng Shen, Daguan Chen, Yu-Lin Wei, Zhijian Yang, and Romit Roy Choudhury. 2020. Voice Localization Using Nearby Wall Reflections. In *ACM MobiCom*.
[50] Zhichuang Sun, Ruimin Sun, Long Lu, and Alan Mislove. 2021. Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps. In *USENIX Security*. 1955–1972.
[51] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. 2020. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293* (2020).
[52] Amee Trivedi, Camellia Zakaria, Rajesh Balan, Ann Becker, George Corey, and Prashant Shenoy. 2021. Wifitrace: Network-based contact tracing for infectious diseases using passive wifi sensing. *ACM UbiComp* 5, 1 (2021), 1–26.
[53] Yue Wang, Cheng Si, and Xintao Wu. 2015. Regression model fitting under differential privacy and model inversion attack. In *IJCAI*.
[54] WHO. 2019. Coronavirus disease (COVID-19): Contact tracing. [who.int].
[55] Wikipedia. 2019. Receiver operating characteristic. [wikipedia.org].
[56] Yaxiong Xie, Zhenjiang Li, and Mo Li. 2018. Precise power delay profiling with commodity Wi-Fi. In *ACM MobiCom*.
[57] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2019. mD-Track: Leveraging multi-dimensionality for passive indoor Wi-Fi tracking. In *ACM MobiCom*.
[58] Yaxiong Xie, Yanbo Zhang, Jansen Christian Liando, and Mo Li. 2018. SWAN: Stitched Wi-Fi ANtennas. In *ACM MobiCom*.
[59] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: A fine-grained indoor location system. In *USENIX NSDI*.
[60] Fan Yi, Yaxiong Xie, and Kyle Jamieson. 2021. Cellular-assisted COVID-19 contact tracing. In *ACM HealthDL*. 1–6.
[61] Chi Zhang and Xinyu Zhang. 2017. Pulsar: Towards Ubiquitous Visible Light Localization. In *ACM MobiCom*.
[62] Jingwen Zhao, Yunfang Chen, and Wei Zhang. 2019. Differential privacy preservation in deep learning: Challenges, opportunities and solutions. *IEEE Access* 7 (2019), 48901–48911.

[63] Mingmin Zhao, Yonglong Tian, Hang Zhao, Mohammad Abu Alsheikh, Tianhong Li, Rumen Hristov, Zachary Kabelac, Dina Katabi, and Antonio Torralba. 2018. RF-based 3D skeletons. In *ACM SIGCOMM*.
[64] Qingchuan Zhao, Haohuang Wen, Zhiqiang Lin, Dong Xuan, and Ness Shroff. 2020. On the accuracy of measured proximity of bluetooth-based contact tracing apps. In *Springer SECURECOMM*.
[65] Shilin Zhu and Xinyu Zhang. 2017. Enabling High-Precision Visible Light Localization in Today's Buildings. In *ACM MobiSys*.