A real-time fingerprint-based indoor positioning using deep learning and preceding states

Mohammad Nabati, Seyed Ali Ghorashi

PII: S0957-4174(22)01907-8
DOI: https://doi.org/10.1016/j.eswa.2022.118889
Reference: ESWA 118889

To appear in: *Expert Systems With Applications*

Received date : 10 May 2022
Revised date : 1 September 2022
Accepted date : 20 September 2022

A Real-Time Fingerprint-Based Indoor Positioning using Deep Learning and Preceding States

Mohammad Nabati[a], Seyed Ali Ghorashi[a,b,*]

[a] *Cognitive Telecommunication Research Group, Department of Telecommunications, Faculty of Electrical Engineering, Shahid Beheshti University, Tehran, Iran*
[b] *Department of Computer Science & Digital Technologies, School of Architecture, Computing and Engineering, University of East London, E16 2RD London, U.K.*

Abstract

In fingerprint-based positioning methods, the received signal strength (RSS) vectors from access points are measured at reference points and saved in a database. Then, this dataset is used for the training phase of a pattern recognition algorithm. Several noise types impact the signals in radio channels, and RSS values are corrupted correspondingly. These noises can be mitigated by averaging the RSS samples. In real-time applications, the users cannot wait to collect uncorrelated RSS samples to calculate their average in the online phase of the positioning process. In this paper, we propose a solution for this problem by leveraging the distribution of RSS samples in the offline phase and the preceding state of the user in the online phase.

In the first step, we propose a fast and accurate positioning algorithm using a deep neural network (DNN) to learn the distribution of available RSS samples instead of averaging them at the offline phase. Then, the similarity of an online RSS sample to the RPs' fingerprints is obtained to estimate the user's location. Next, the proposed DNN model is combined with a novel state-based positioning method to more accurately estimate the user's location. Extensive experiments on both benchmark and our collected datasets in two different scenarios (single RSS sample and many RSS samples for each user in the online phase) verify

*Corresponding author
Email addresses: mo.nabati@mail.sbu.ac.ir (Mohammad Nabati), s.a.ghorashi@uel.ac.uk (Seyed Ali Ghorashi)

the superiority of the proposed algorithm compared with traditional regression algorithms such as deep neural network regression, Gaussian process regression, random forest, and weighted KNN.

Keywords: Fingerprint-based positioning, Wi-Fi, smartphone, machine learning, deep learning.

1. Introduction

Real-time positioning with smartphones is an emerging technology in location-based services. Global positioning system by employing the satellite signals around the Earth plays a decisive role in real-time positioning for human requirements (Khalajmehrabadi et al., 2017a). However, these signals cannot penetrate into indoor environments. Also, they suffer from non-line-of-sight (NLOS) errors and are not practical for indoor or harsh outdoor areas (Khalajmehrabadi et al., 2017a; Nabati et al., 2020). Satellite-based positioning methods regularly extract the distance between some transmitters and a user receiver. Then, triangulation or trilateration methods are used to estimate the user's location (Wang et al., 2013; Thomas & Ros, 2005). The distance between a transmitter and receiver can be obtained by the angle of arrival, time of arrival, time difference of arrival, phase of arrival, or received signal strength (RSS) (Zafari et al., 2019). These approaches are also known as ranging-based positioning techniques (Ghari et al., 2019) that can be locally implemented for an indoor environment with some base stations (BSs). Among these methods, RSS can be easily captured, since it does not need a synchronous time clock between transmitters and the receiver. However, all of these ranging-based positioning techniques do not provide a promising accuracy in indoor areas due to the NLOS errors, even if they are implemented locally (Nabati et al., 2021).

The fingerprint-based positioning method is developed to deal with the ranging-based positioning drawback. In this method, some BSs are deployed in an indoor area and share radio signals, which can be in the form of Wi-Fi (Du et al., 2018), Bluetooth (Aranda et al., 2022), ZigBee (Zheng et al., 2017), and radio frequency identification (Ma et al., 2019). Among these communication technologies, Wi-Fi BS (known as Wi-Fi access point (AP)) is the prominent one due to its ubiquitous availability in indoor environments (Hernández et al., 2021). After deploying APs, two different phases are performed to establish a fingerprint-based positioning method: offline (or training) and online (or test) phases (Bai et al., 2021). First, RSS or channel state information (CSI) from several APs is acquired at some known locations called reference points (RPs) and stored in a database. Usually, a pattern recognition algorithm (PRA) is trained in the offline phase, which converts input signals' attributions (CSI and/or RSS) to the corresponding locations. The trained PRA estimates the users' locations in the online phase, using the received signals' attributions.

The CSI as a fingerprint is more stable than RSS (Wang et al., 2017). However, it cannot be captured with the typical AP modes and needs particular hardware and software tools on the transmitter and receiver sides (Zafari et al., 2019), whereas RSS can be easily captured even with an off-the-shelf smartphone. Thus, RSS can be launched for popular-practical and industrial applications. However, multi-path effects such as large-scale fading (known as shadowing) and small-scale fading profoundly impact the RSS (Zanella, 2016), and correspondingly the RSS-based positioning accuracy (Prasad et al., 2018).

Most of the existing works average out RSS samples in both the offline and online phases or assume that there are sufficient RSS samples[1] for users in the online phase. However, users obviously cannot wait to collect uncorrelated RSS samples in the online phase of a real-time positioning system using smartphones, because they do not provide high sampling rates for scanning of the Wi-Fi signals (Liu & Liu, 2018). Even without considering the restrictions, we lose the statistical distribution of RSS samples by getting the average of them for each RP.

[1]The term "sufficient RSS samples" refers to an acceptable number of samples received from APs at a fixed location, such that the cumulative moving average of RSS samples becomes stable after that sample. This concept is described in Fig. 4

Although sufficient RSS samples cannot be obtained in the online phase using off-the-shelf smartphones, the RSS samples of RPs in the offline phase can help to recognize the statistical behavior of RSS observations. In this paper, we propose a novel fingerprint-based positioning algorithm in which the RSS distribution of RPs is captured during the training process. Also, a state-based positioning method is utilized to consider the previous state information of users assuming that the user moves around the area in the online phase. We first use a deep neural network (DNN) to learn the distribution of RSS samples instead of directly converting RSS space into coordinate space. The output of the proposed DNN is the similarity of RSS samples to each RP. In other words, each RP in our proposed method is a class, and its RSS samples are class observations. In the online phase, the trained DNN estimates the similarity of users' RSS samples to each RP. Then, users' locations can be estimated by the weighted average of different RPs' locations, where weights are outputs of the trained DNN (similarities). The proposed method also is extended for situations in which there are sufficient RSS samples for users in the online phase. It improves the accuracy of positioning compared with its counterparts in two different datasets (an open-source dataset as well as a dataset that we collected ourselves), considering two scenarios (a single RSS sample and sufficient RSS samples in the online phase). Inspired by the fact that the user exists in the proximity of the previous states during its movement, we combine the proposed DNN-based positioning method with a novel state-based positioning algorithm to tackle the limitation of online phase positioning. In summary, the main contributions of this paper are as follows:

- We propose a DNN-based positioning algorithm that considers statistical behaviors of RSS samples at RPs in the offline phase, and a cost function is suggested for our proposed DNN model to optimize the layers' weights for giving a higher weight to the nearest RP as much as possible.
- A novel state-based positioning method is suggested, which uses the previous states of users to increase the location estimation accuracy.

- We provide a complexity analysis for our proposed positioning algorithm, and extensive experiments are conducted to show the robustness of the proposed method.

The rest of this paper is organized as follows: In section 2 related works are presented. In section 3 we present the preliminaries of the fingerprint-based positioning method, and the conventional DNN-based positioning model is described. In section 4 we explain the proposed DNN in which the distribution of RSS samples at each RP is learned and then combined with the proposed state-based positioning method. In section 5 the experimental results are presented to compare the proposed method with its counterparts. Finally, section 6 concludes this work.

In this paper, we use bold-face capital letters for matrices or arrays (e.g., $\mathbf{A}$), bold-face lower-case letters to indicate vectors (e.g., $\mathbf{a}$), and lower-case letters to show scalars (e.g., a). Also, a_{ij} shows the element at the i^{th} row and j^{th} column of matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$, and $a_{ij}(n)$ is used to indicate the element's location of a 3D array $\mathbf{A} \in \mathbb{R}^{I \times J \times N}$.

2. Related works

To solve the drawbacks of RSS-based positioning methods, many researchers have introduced novel techniques. Prasad et al. (2018) propose to use noise-free RSS training data generated by a path-loss model in the offline phase of the positioning process. They assume that there are enough RSS samples for users in the online phase such that RSS is not affected by small-scale fading, and a probabilistic-based method named Gaussian process regression is used to determine the users' locations. Yang et al. (2015) propose a probabilistic-based k-nearest neighbors (KNN) method using all samples in the offline phase, and also they assume that there are sufficient RSS samples at the online phase of their proposed algorithm. Afuosi & Zoghi (2020) propose an improved version of the KNN method utilizing the weighted average of RSS samples to reduce the online searching process for the nearest RPs. Getting the average from

RSS samples at the offline phase mitigates the effect of small-scale fading and reduces the calculations of KNN or kernel-based algorithms such as Gaussian process regression (Prasad et al., 2018; Homayounvala et al., 2019). Khalajmehrabadi et al. (2017b) propose a clustering-based approach to reduce the searching process of the online phase, using the Hamming distance between online measurements and the centroid of each cluster. Their proposed method needs adequate RSS samples in the online phase to define the sparse vectors for Hamming distance calculations. Fang et al. (2015) propose a histogram equalization-based positioning method in which the histogram of online RSS measurements is calculated with sufficient RSS samples.

Many DNN-based positioning approaches have been studied in recent years for different purposes of localization. (Nabati et al., 2020) proposed a deep learning model, called generative adversarial networks, to learn the distribution of limited data in a four-class problem. The learned DNN model is then used to generate synthetic data, which can be combined with real data to enhance overall localization accuracy. (Zheng et al., 2020) proposed a self-calibration deep learning framework using auto-encoders to reduce the effect of environmental changes on localization performance. (Dai et al., 2019) used the DNN for dividing the area into four subareas, and an improved KNN is used to determine the location. (Zhou et al., 2021) exploit the fingerprints' correlations to reconstruct them with a deep learning framework. (Lee et al., 2022) proposed a DNN-based framework to automatically reconstruct the fingerprinting dataset due to the environmental changes.

Most of the researchers have used Bayesian approaches such as Kalman filter, particle filter, and their modifications to employ previous states of the users in the online phase of positioning. Mahfouz et al. (2014) employed a machine learning (ML) framework to estimate the first position of the user, and then utilized Kalman filter with acceleration data to extract the previous state information. Zhu et al. (2018) utilized RSS values of radio-frequency-identification (RFID) as fingerprints of RPs in the offline phase. In the online phase, they used a modified version of Kalman filter with phase shift data extracted from

RFID to obtain the information of previous states. Xie et al. (2016) used magnetic field information as fingerprints and employed a particle filter, which uses additional information such as velocity acceleration and angular velocity obtained from inertial sensors. Recently, Silva et al. (2021) proposed a method that combines Wi-Fi signals with motion sensors to track the vehicles in indoor areas.

Most of the DNN approaches are applied to learn the distribution of RSS samples in subareas rather than learning the distribution of RSS samples at each RP in the offline phase. Sometimes, finding the subarea is the purpose of the localization. However, if the goal is to find the exact location, a pattern matching algorithm is utilized after finding the subarea. The proposed DNN finds the exact location in the online phase of positioning, which makes it different from the existing works in the literature. Also, all the aforementioned state-based positioning approaches employ extra information extracted from additional sensors, which are usually expensive and not available in any communication devices. The main difference between the proposed state-based method is that it does not need additional information and only uses preceding RSS samples and estimated locations in the online phase.

3. Preliminaries

In this section, we present the fundamental concept of fingerprint-based RSS positioning in the offline and online phases. Then, the conventional DNN-based regression (DNNR) positioning algorithm is descried. This is required as we will compare our proposed DNN later with this algorithm.

3.1. Fingerprint-Based Positioning

Map construction is the initial step of fingerprint-based positioning approaches, in which a reference position is considered for the origin of the environment as depicted in Fig. 1. Then, RSS values of APs are recorded at the determined coordinates known as RPs. Also, the locations of APs are not

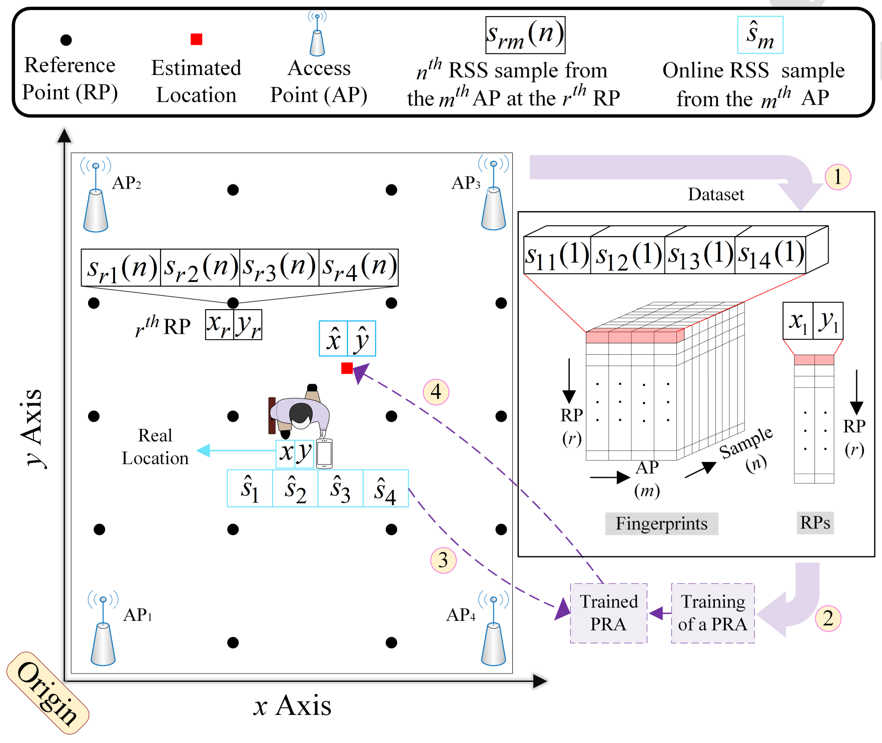


Figure 1: The entire process of fingerprint-based RSS positioning that can be performed in four steps denoted by 1, 2, 3, and 4. In 1, the RSS samples are collected at RPs from several APs. Then, a 3D RSS dataset and two locations' vectors are obtained, which are depicted in Eq. (1) and (2), respectively. In 2, the dataset is delivered to a PRA for training. In 3, the online RSS sample is conveyed to the input of the trained PRA. Finally, in 4, the trained PRA converts the received RSS sample to the location.

needed to be known in the environment, since we assume that APs are not moving and their RSS values are unique fingerprints at RPs. RSS values and RPs locations are used as PRAs' input and output, respectively. RSS is usually measured several times at a fixed point. The structure of the raw RSS dataset for all RPs can be shown by a 3D array as follows

$$\mathbf{S} = \begin{pmatrix} s_{11}(n) & s_{12}(n) & \cdots & s_{1M}(n) \\ s_{21}(n) & s_{22}(n) & \cdots & s_{2M}(n) \\ \vdots & \vdots & \ddots & \vdots \\ s_{R1}(n) & s_{R2}(n) & \cdots & s_{RM}(n) \end{pmatrix}_{R\times M\times N} \quad (1)$$
$$n = 1, 2, ..., N\,,$$

where $\mathbf{S} \in \mathbb{R}^{R\times M\times N}$ is a 3D array consists of all APs' samples at RPs, $s_{rm}(n)$ is the magnitude of the n^{th} received sample from the m^{th} AP at the r^{th} RP, R is the number of RPs, M is the number of APs, and N is the number of RSS samples at each RP. Also, RPs' locations can be represented as follows

$$\mathbf{x} = [x_1, x_2, \cdots, x_R]^T, \quad \mathbf{y} = [y_1, y_2, \cdots, y_R]^T. \quad (2)$$

Two types of noises are mixed with the raw RSS dataset in (1), namely small-scale and large-scale fading. The large-scale fading cannot be mitigated by averaging the samples over time, because it is space-dependent and approximately time-invariant (Zanella, 2016). Thus, the large-scale fading remains in the data. However, the effect of small-scale fading can be reduced by averaging the RSS samples over time. The average of RSS samples from the raw dataset in (1) can be shown by a 2D matrix as follows

$$\bar{\mathbf{S}} = \begin{pmatrix} \bar{s}_{11} & \bar{s}_{12} & \cdots & \bar{s}_{1M} \\ \bar{s}_{21} & \bar{s}_{22} & \cdots & \bar{s}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{s}_{R1} & \bar{s}_{R2} & \cdots & \bar{s}_{RM} \end{pmatrix}_{R\times M} , \quad (3)$$

where $\bar{s}_{rm}$ is the average of RSS samples for the m^{th} AP at the r^{th} RP. Most of the traditional positioning algorithms propose to use the average dataset instead of the raw RSS dataset to alleviate the impact of small-scale fading and to reduce the time of the searching process. Typically, a PRA learns how to convert RSS space into coordinate space. When the training phase is performed, the trained PRA can estimate users' locations. The schematic of all these processes is depicted in Fig. 1. As can be seen in Fig. 1, the RSS values of APs are measured at RPs (black dots). Then, these data are conveyed to the database, and the PRA performs an optimization process for fitting its hyperparameters (or its weights) to the collected data. In the online phase, a received RSS is fed to the PRA for location estimation. In section 3.2, we explain this process for conventional DNN-based positioning, which is used as a PRA.

3.2. Conventional DNN-based Positioning

Here, we explain the conventional DNN-based regression positioning algorithm, which learns how to map RSS vectors to the Cartesian coordinates. This section describes the basic model that typically is used to predict the user location. Reading this section helps the readers to understand the differences between the proposed DNN and the conventional one. The conventional DNN-based regression positioning consists of M input nodes for RSS samples and two output nodes for x and y coordinates. Since the DNN model wants to predict the locations based on APs' signals, the number of input nodes equals the number of APs M. In Fig. 2, the architecture of DNN-based regression positioning is illustrated, where f_l is the activation function of the l^{th} layer, $\mathbf{W}_l$ is the weight matrix for the l^{th} layer, and $\mathbf{v}_l$ is the input vector of the l^{th} layer. The layers' weights are optimized during the training phase. In the training phase of DNN regression, the goal is to minimize the mean squared error (MSE) between the train coordinates in (2) and predicted coordinates by DNN through the following objective function

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2R}\sum_{r=1}^{R}\left([f_L^x]_r - x_r\right)^2 + \left([f_L^y]_r - y_r\right)^2, \tag{4}$$

where x_r and y_r are actual coordinates of the r^{th} RP, $[f_L^x]_r$ and $[f_L^y]_r$ are predicted outputs for the r^{th} RP, and R is the number of RPs. The backpropagation algorithm is utilized for minimization of the objective function in (4), which is performed in two steps: forward propagation and backward propagation (Lee et al., 2016). After optimization, the trained DNN regression can directly convert the test RSS samples to Cartesian coordinates. Conventional

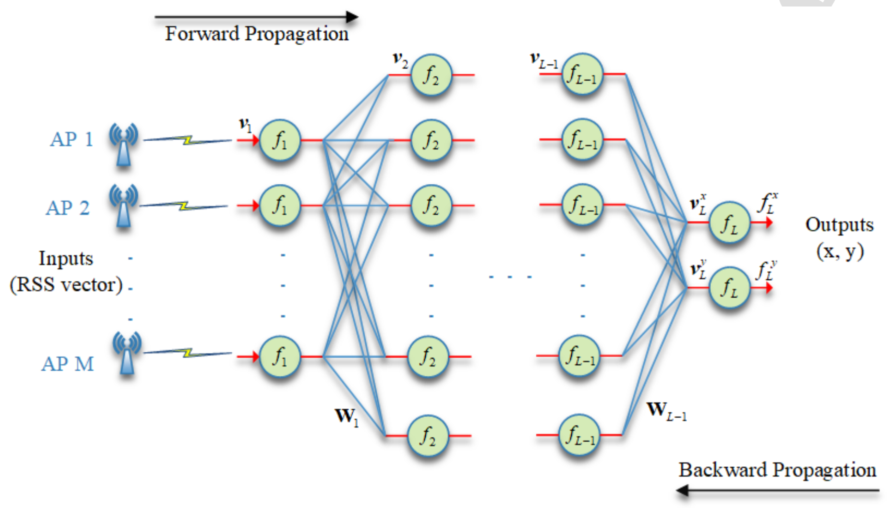


Figure 2: Conventional DNN-based regression positioning.

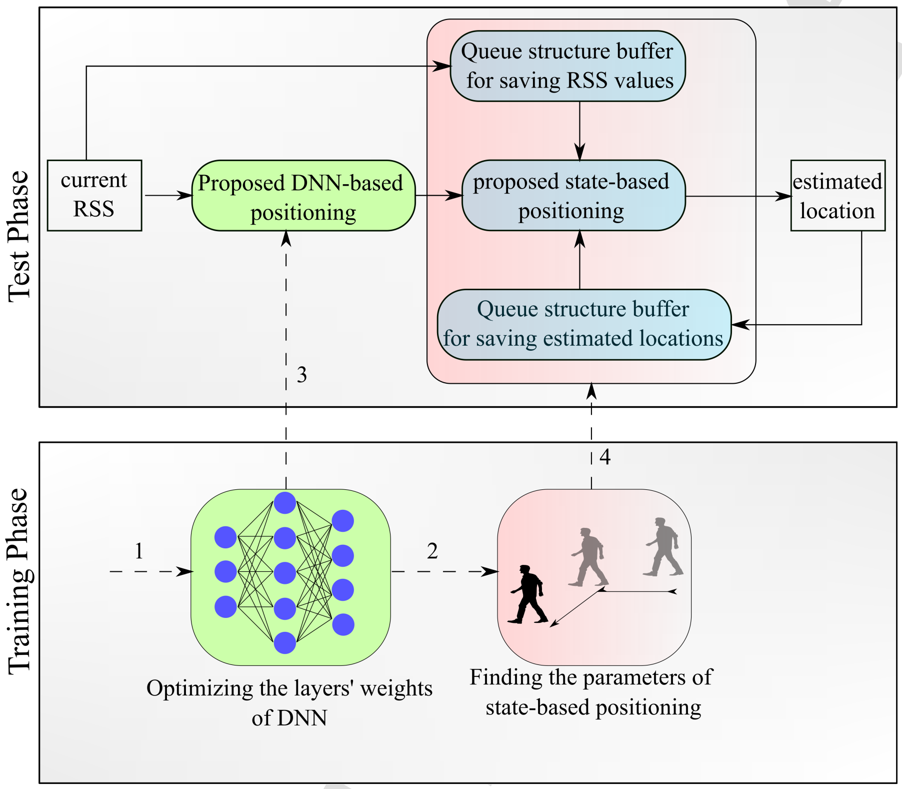
185 DNN regression does not consider the statistical behavior of RSS samples at each RP. In the next section, we explain our proposed method to consider all RSS samples of RPs during the training phase.

4. Proposed positioning algorithm

In this section, we first explain the proposed DNN-based positioning ap-
190 proach and then combine it with a novel state-based positioning method. The whole architecture of the proposed model is depicted in Fig. 3. The proposed DNN-based positioning method learns the distribution of available RSS samples at each RP in *offline* phase, and on the other side, the proposed state-based positioning method uses the *online* RSS samples stored from previous states of
195 the user. More details of Fig. 3 will be explained in the following.

4.1. Proposed DNN-based positioning method

Conventional DNN regression cannot be used for recognizing the statistical distribution of RSS samples at each RP. Also, users in real-time applications cannot wait to collect enough RSS samples in online phase with smartphones.



Test Phase
Training Phase
current RSS
Proposed DNN-based positioning
Queue structure buffer for saving RSS values
proposed state-based positioning
Queue structure buffer for saving estimated locations
estimated location
3
4
1
2
Optimizing the layers' weights of DNN
Finding the parameters of state-based positioning

Figure 3: The whole architecture of the proposed model. Steps of training the DNN, finding optimum values for state-based positioning, and then feeding them to whole architecture of proposed method are shown with numbers 1, 2, 3, and 4. Queue structure buffer is used for feeding the data based on first-in-first-out order.

In Fig. 4, 75 RSS samples of two different APs are illustrated as an example. These RSS samples have been recorded with a smartphone in 100 consecutive seconds at a fixed point. The cumulative moving average, which is the average of RSS values before the n^{th} sample, has also been illustrated in Fig. 4. As shown, RSS samples do no change before at least three consecutive observations (which in this example equals to 3.75 secs). Also, the average of RSS samples is not stable before 40 samples (which in this example equals to 50 secs). These observations show that users cannot wait to collect uncorrelated RSS samples in a real-time positioning application using smartphones. Depending on the

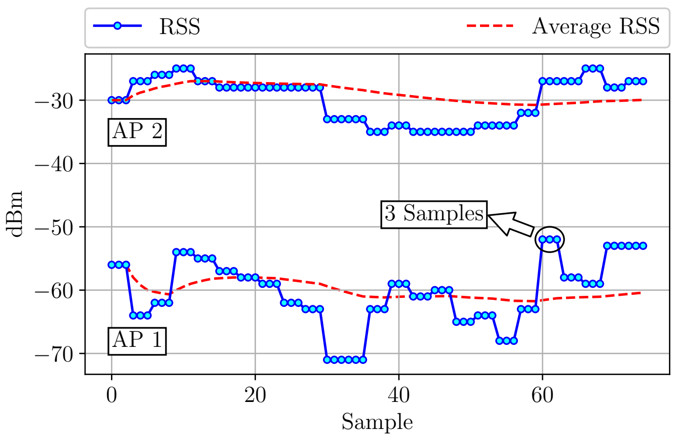


Figure 4: Typical fluctuations of 75 RSS samples of two different APs at one point for 100 consecutive seconds. "Average RSS" is the average of RSS samples before the n^{th} sample, which is called cumulative moving average.

smartphone model and Wi-Fi APs, the mentioned values can be different, which are related to the sample rate of Wi-Fi signal and stability of signals. However, the main problem remains in the positioning system: a high number of samples in online phase of positioning cannot be obtained. Although we cannot access the high number of RSS samples in online phase, the samples of RPs in offline phase can be used to recognize the RSS distributions, and this is the core idea behind the proposed DNN model. This model not only is feasible with only a single RSS sample, but also can be extended for every desired number of samples, assuming that the receiver supports high sample rates.

In our proposed DNN architecture, the input layer is the same as conventional DNN regression. However, the number of output nodes equals the number of RPs, as depicted in Fig. 5. The proposed method takes all RSS samples of RPs during the optimization process and learns the statistical behavior of RSS samples at each RP. In other words, each RP in our proposed method is a class, and the distribution of RSS samples at each RP is obtained during the optimization process. Also, it uses the 3D array in (1) which consists of all RSS samples at RPs instead of averaged RSS dataset in (3). The softmax activation function is used for the output layer to limit the output values between 0 and 1, where the sum of output values after passing from the softmax activation function equals 1. By using this activation function, the values of output nodes are interpreted as the probability of belonging to each RP as they sum to one. In other words, the proposed DNN is a multivariate probability density function, which gives a higher weight to similar samples and enables it to provide more information even with a single RSS sample in the online phase. The r^{th} node in the output layer is calculated as follows

$$\text{SoftMax} \rightarrow f_L^r = \frac{exp(\mathbf{v}_L^r)}{\sum\limits_{q=1}^{R} exp(\mathbf{v}_L^q)}, \tag{5}$$

where $\sum_{r=1}^{R} f_L^r = 1$. For the training phase of the proposed method, we cannot directly use the locations of RPs (x and y labels in (2)) as output labels for the classification problem. Instead, we can use one-hot encoded labels for RPs as follows

$$C = \{\underbrace{[1, 0, \cdots, 0]_1^T}_{\mathbf{c}^1}, \underbrace{[0, 1, \cdots, 0]_2^T}_{\mathbf{c}^2} \cdots \underbrace{[0, 0, \cdots, 1]_R^T}_{\mathbf{c}^R}\}. \tag{6}$$

where $\mathbf{c}^r$ is the label for samples of the r^{th} RP in (1). The proposed objective function in the optimization process is log-likelihood or cross-entropy which is defined as follows (Hastie et al., 2009)

$$\mathcal{L}_{\text{ENT}} = -\sum_{j=1}^{RN} \sum_{r=1}^{R} [\mathbf{c}^r]_j \log\left([f_L^r]_j\right), \tag{7}$$

where R is the number of the RPs (or output nodes), N is the number of samples at each RP, RN is the total number of samples in training set, $[\mathbf{c}^r]_j \in \{0, 1\}$ is the desired output of the j^{th} sample at the r^{th} node, and finally $[f_L^r]_j \in [0, 1]$ is the predicted output for the j^{th} sample at the r^{th} node. The cross-entropy loss function is not penalized for undesired output nodes where real output values are 0; however, it is logarithmically penalized for desired output nodes where the predicted value of the desired class is far away from 1. It means that the log

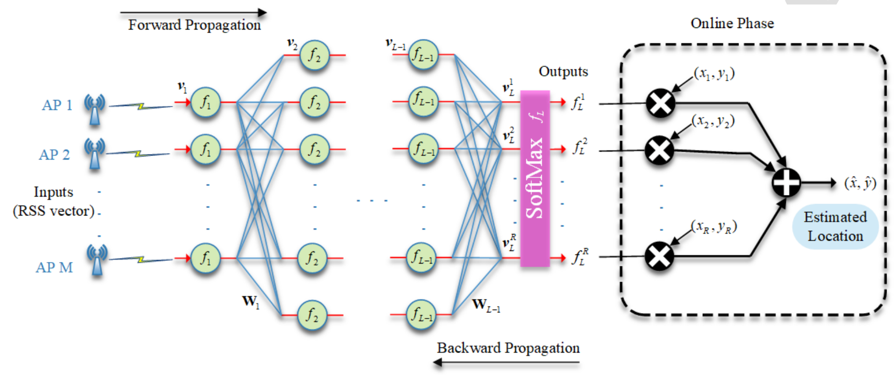


Figure 5: Proposed DNN architecture for fingerprint-based positioning.

function favors the hard selection of a single class, which is necessary to optimize the layers' weights in this way; because the obtained probability corresponding to the closest RP should be large as much as possible. Considering each RP as a class for the training of the proposed DNN is not easy, especially when RPs are close to each other, because the adjacent RPs usually have similar RSS values, and the *arg min*$\mathcal{L}_{\mathrm{ENT}}$ gives an overfit solution most of the time. Some tricks can be used to avoid this problem, such as using the regularization and/or dropout layers (Srivastava et al., 2014). First, we must ensure that the cost function stated in (7) converges. We use the theorem presented in (Zhen et al., 2018).

Theorem 1: If a cost function is bounded from below and monotonically decreased with a gradient-based algorithm, it converges.

In (7), if we prove that the term $-[\mathbf{c}^r]_j \log\left([f_L^r]_j\right)$ is bounded from below for every j and i, it means that $\mathcal{L}_{\mathrm{ENT}}$ is bounded from below, since the summation is performed over a limited number of samples. The term $[\mathbf{c}^r]_j$ equals to 0 or 1. If it is zero, the whole term $-[\mathbf{c}^r]_j \log\left([f_L^r]_j\right)$ equals to 0. Let's assume that the term $[\mathbf{c}^r]$ is 1. As we mentioned, the output of softmax activation function is between 0 and 1 (i.e., $0 < [f_L^r]_j < 1$). Therefore, $-\infty < \log\left([f_L^r]_j\right) < 0$ and we can write $0 < -[\mathbf{c}^r]_j \log\left([f_L^r]_j\right) < \infty$. Therefore, $\mathcal{L}_{\mathrm{ENT}}$ is bounded from below, and because it monotonically decreases with a gradient-based algorithm (e.g., Adam), it converges.

After the training phase, a straightforward way for location estimation is to consider the maximum value of output nodes as the predicted class, and the location of predicted RP can be considered as the final estimation. However, this strategy does not consider other RPs' similarities. The sum of multiplying the similarity of output nodes to the RPs' locations is a better choice because it considers all RPs' similarities as depicted in the online phase block of Fig. 5. Therefore, if a user receives an RSS vector $\hat{\mathbf{s}} = [\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_M]^T$, the location coordinates ($\hat{x}$, $\hat{y}$) can be estimated after passing from DNN units $\mathbf{p} = f_L(\mathbf{W}_{L-1}^T f_{L-1}(\mathbf{W}_{L-2}^T f_{L-2} \cdots \mathbf{W}_2^T f_2(\mathbf{W}_1^T f_1(\hat{\mathbf{s}})) \cdots))$ as follows

$$\hat{x} = \mathbf{x}^T \mathbf{p} \quad \text{and} \quad \hat{y} = \mathbf{y}^T \mathbf{p}, \tag{8}$$

where $\mathbf{p} = [p_1, p_2, \cdots, p_R]^T$ is the similarity (probability) vector obtained from DNN units in online phase. Eq. (8) is used when a single RSS sample is available in the online phase. Assuming that the receiver can provide a high sampling rate and we have access to enough samples in the online phase, a probability matrix $\mathbf{P}$ is obtained for each user that can be used to more accurately estimate the user's location. Considering $\hat{N}$ samples for users in the online phase, the location of a user can be calculated as follows

$$\hat{x} = \frac{\sum_{i=1}^{\hat{N}} \sum_{j=1}^{R} x_j p_{ij}}{\sum_{i=1}^{\hat{N}} \sum_{j=1}^{R} p_{ij}} = \frac{\sum_{i=1}^{\hat{N}} \mathbf{x}^T \mathbf{p}_i}{\hat{N}}, \quad \hat{y} = \frac{\sum_{i=1}^{\hat{N}} \sum_{j=1}^{R} y_j p_{ij}}{\sum_{i=1}^{\hat{N}} \sum_{j=1}^{R} p_{ij}} = \frac{\sum_{i=1}^{\hat{N}} \mathbf{y}^T \mathbf{p}_i}{\hat{N}} \tag{9}$$

where $\hat{x}$ and $\hat{y}$ are estimated locations, $\mathbf{x}$ and $\mathbf{y}$ are coordinates of RPs' locations, $\mathbf{p}_i$ is the i^{th} column of the probability matrix $\mathbf{P}$, and R is the number of RPs.

4.2. Proposed state-based positioning method

The proposed DNN-based positioning method, presented in section 4.1, learns the distribution of offline available samples to enhance the positioning accuracy. However, during the users' movement, their previous RSS samples and estimated locations in the online phase can be helpful to achieve high accuracy. Motivated by the fact that the user is placed in the proximity of its previous

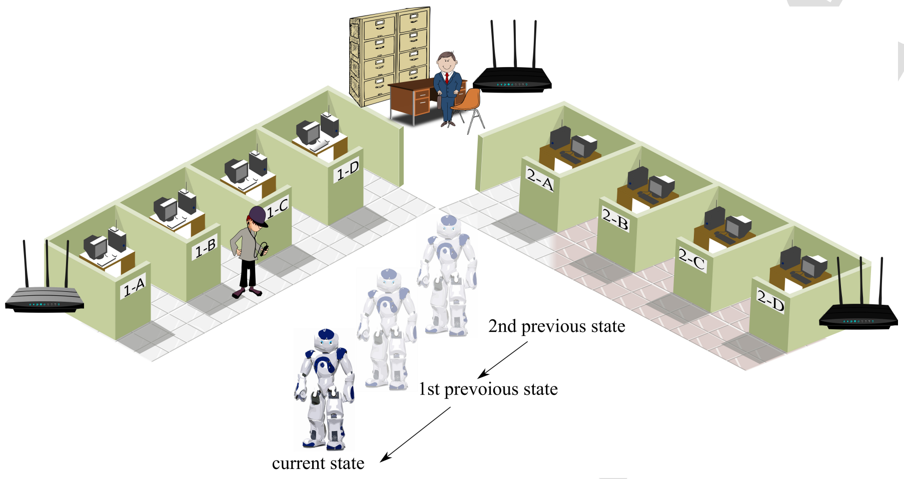


Figure 6: A schematic of an indoor office with a moving user.

states (depicted in Fig. 6), we propose a state-based positioning method to take the previous information into account. The proposed method needs previous locations obtained from the proposed DNN model and the previous RSS samples as depicted in Fig. 3. This information (including estimated locations and RSS samples) is stored in a queue-based structure buffer whose stored data are used based on the first-in-first-out order. We should set a maximum value to the buffer size. Setting up a high value to the buffer size can worsen the accuracy due to using irrelevant information far from the current location. For the first steps where the positioning system is established and the states do not reach the maximum buffer size, we only use the available states. If we define $\dot{x}$ to be the output of the proposed state-based positioning method, the estimated location at the i^{th} step can be obtained as follows

$$\dot{x}_i = \frac{\hat{x}_i + \sum_{j=1}^{\Omega} \alpha_j \beta_j \dot{x}_{i-j}}{1 + \sum_{j=1}^{\Omega} \alpha_j \beta_j}, \quad \dot{y}_i = \frac{\hat{y}_i + \sum_{j=1}^{\Omega} \alpha_j \beta_j \dot{y}_{i-j}}{1 + \sum_{j=1}^{\Omega} \alpha_j \beta_j}, \tag{10}$$

where α and β are adjuster weights, and Ω is the number of stored states in the buffer. α is defined based on the state order such that it gives a higher weight to the most recent state. β can be defined by kernel functions to give a higher weight to the most similar RSS of previous states. In other words, α and β model the importance of the recent states and their similarity, respectively. The weights are defined as follows

$$\alpha_j = \frac{\sqrt{\Omega - j}}{\Omega}, \quad \beta_j = \exp\left(-\frac{|\hat{\boldsymbol{s}}_i - \hat{\boldsymbol{s}}_j|}{\gamma^2}\right), \tag{11}$$

where γ is the width of the given kernel function that can be set up, experimentally. The proposed state-based positioning algorithm is practically convenient since it does not need to know whether the user is moved or not. If the user does not move in the environment, the current state is not changed; however, the previous estimated locations in (10) are average out based on calculated weights, meaning that averaging process is performed for a fixed location. It means that the proposed method can estimate the user location without needing the movement state. Please note that the DNN model is the core of the proposed state-based positioning method. Although we can choose another ML strategy as the core, it should be accurate enough to properly collaborate with the proposed state-based positioning model.

The drifting problem is a common concern for state-based positioning methods. It usually happens when the locations are only predicted by the previous states. For instance, pedestrian dead reckoning suffers from this problem (Chen et al., 2016). However, the proposed state-based positioning can handle this problem because of two reasons. The first reason is that the DNN method at each step predicts the location and has the most influence on the current predicted location. The second reason is that the adjuster weight β_j gives fewer weights to states that are far from the current position in the RSS space.

Complexity Analysis: Here, we only describe *online phase* multiplication complexity of the proposed method and compare it with traditional DNN regression, because the critical part of a real-time positioning algorithm is its online phase complexity. First, we explain the complexity of the conventional DNN regression, and then the complexity of the proposed method is presented. Note that the complexity of typical activation functions' calculations can be assumed to be the same as multiplication complexity because they have a finite number of multiplications in their definitions. Also, some activation functions depend on the exponential function that also can be expressed by a finite number of multiplications in the Taylor series. However, the complexity of the softmax activation function depends on all its inputs and will be discussed in more detail. We also do not discuss the state-based positioning complexity, since its complexity order is less than DNN-based positioning and can be ignored easily.

Complexity of conventional DNN-based regression positioning: In the conventional DNN regression, the number of input nodes equals M, the number of layers equals L, and there are two output nodes. Also, the number of nodes in layer l is assumed to be h_l where $h_1 = M$ and $h_L = 2$. There are $L-1$ layers where output nodes are calculated by activation functions and then multiplied by matrix weights. In the last layer, activation function calculation is only needed, which is typically linear for a regression problem. The complexity of activation function calculation generally depends on the activation function type. Therefore, the complexity order of activation function calculation in layer l is $\mathcal{O}(a(h_l))$ where $a(h_l)$ is the complexity of activation function. Also, matrix weight multiplication to the output of nodes in layer l causes $\mathcal{O}(h_l h_{l+1})$ calculation complexity. Consequently, the entire complexity order of layer l is $\mathcal{O}(a(h_l) + h_l h_{l+1})$. As explained above, the activation function calculation complexity is the same as multiplication complexity; therefore, the complexity calculation of the l^{th} layer can be simplified to $\mathcal{O}(h_l + h_l h_{l+1}) \approx \mathcal{O}((h_l + 1)h_{l+1}) \approx \mathcal{O}(h_l h_{l+1})$, and the whole complexity calculation of conventional DNN regression can be written as follows

$$\mathcal{O}(\sum_{i=1}^{L-1} h_i h_{i+1}). \tag{12}$$

As can be seen, the complexity of conventional DNN regression does not depend on the number of RPs, since in the output layer we have $h_L = 2 \ll h_i$ for $i < L$.

Complexity of proposed DNN algorithm: The number of input nodes in our proposed DNN architecture is the same as that of conventional DNN regression, and the number of output nodes equals the number of RPs. The complexity of layers till layer $L-2$ is the same as conventional DNN regression. For the $(L-1)^{th}$ layer, complexity of activation function is $a(h_{L-1})$, however, matrix weight multiplication in the $(L-1)^{th}$ layer depends on the number of RPs, therefore the complexity of the $(L-1)^{th}$ layer equals to $\mathcal{O}(a(h_{L-1}) + h_{L-1}R)$. In the latest layer, the output of the softmax activation function depends on the other outputs as shown in (5). We can assume that the complexity of *exp* function is the same as multiplication complexity because it can be calculated by a finite number of multiplications in the Taylor series. Therefore, the complexity of the softmax activation function is $\mathcal{O}(R^2)$ in the last layer. Also, the multiplication complexity of obtained probabilities to the coordinates is $\mathcal{O}(R)$, which can be ignored due to its less calculation cost than the other operations. Consequently, the entire complexity of the proposed method can be expressed as follows

$$\mathcal{O}(\sum_{i=1}^{L-2} h_i h_{i+1} + h_{L-1}R + R^2). \tag{13}$$

As can be seen in (13), the complexity of the proposed method depends on the number of RPs when the number of RPs is large, and this cost should be paid for enhancing the accuracy. Assuming that there are $\hat{N}$ samples for users in the online phase, the complexity would be $\mathcal{O}(\hat{N}(\sum_{i=1}^{L-2} h_i h_{i+1} + h_{L-1}R + R^2))$. The large-scale areas usually have a large number of RPs. They can be divided into different sub-areas to independently implement the proposed DNN for each sub-area via hierarchical-based positioning algorithms (Luo & Hsiao, 2019) to reduce this dependency on the number of RPs.

5. Experiments and Results

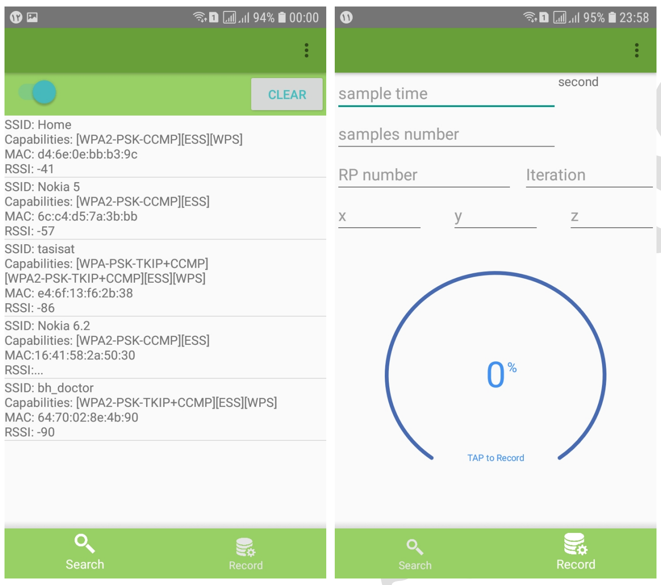
The proposed DNN-based positioning algorithm has been implemented using TensorFlow 2.8.0 library (in Python programming language)[2]. We conducted

[2]We have used an HP ProBook laptop with the following configurations. CPU: Intel(R) Core(TM) i7-10510U @ 2.3GHz, 4 Cores, 8 Logical Processors. GPU: NVIDIA GeForce MX250, 2.0 GB Shared GPU memory. RAM: 16GB, DDR4.

our experiments on two different datasets (Testbed1 and Testbed2). The first dataset has been collected at our university and the other one is a benchmark open access dataset. Testbed1 consists of 250 points and each point has 75 RSS samples. There are no separate TPs in this dataset and TPs are selected randomly. In the Testbed2 there are 345 RPs, and most of these RPs have 50 RSS samples. Also, there are 119 TPs and each TP has 10 RSS samples. We examine the proposed DNN-based positioning method on both datasets. The combination of proposed DNN and state-based positioning methods is only investigated on the benchmark dataset since the test samples of this dataset are separated from the training set and are near each other. The nearness of test samples can simulate the adjacency of the user's previous states during movement. We examine the training and test phases of algorithms on a laptop; however, we can train DNN on the server and transport the DNN's weights to the user device in the online phase to estimate the user location in a real-world scenario. Although DNNs are computationally extensive in the training phase, they are fast in the test phase and can practically be utilized on smartphones, as smartphone programming languages support the parallel computations (Zhang et al., 2019).

5.1. *Testbed 1*

We have developed a software tool for map construction in the offline phase. The screen view of the developed application is illustrated in Fig. 7. This application has been developed via Android Studio and can be installed on smartphones with the Android operating system. As illustrated in Fig. 7(a) on the search tab, the application finds all of the accessible APs and registers their unique media access control (MAC) addresses. This information is stored in a text file within the internal storage of the smartphone. As shown in Fig. 7(b) on the record tab, the application can be customized for the configuration and recording process. The ”sample time” demonstrates the record time for each sample, ”RP number” illustrates an ID for each RP, ”samples number” is the number of samples recorded by determined ”sample time”, ”Iteration”



SSID: Home
Capabilities: [WPA2-PSK-CCMP][ESS][WPS]
MAC: d4:6e:0e:bb:b3:9c
RSSI: -41

SSID: Nokia 5
Capabilities: [WPA2-PSK-CCMP][ESS]
MAC: 6c:c4:d5:7a:3b:bb
RSSI: -57

SSID: tasisat
Capabilities: [WPA-PSK-TKIP+CCMP]
[WPA2-PSK-TKIP+CCMP][ESS][WPS]
MAC: e4:6f:13:f6:2b:38
RSSI: -86

SSID: Nokia 6.2
Capabilities: [WPA2-PSK-CCMP][ESS]
MAC:16:41:58:2a:50:30
RSSI:...

SSID: bh_doctor
Capabilities: [WPA2-PSK-TKIP+CCMP][ESS][WPS]
MAC: 64:70:02:8e:4b:90
RSSI: -90

(a) Search Tab (b) Record Tab

Figure 7: Developed smartphone application for RSS data gathering.

demonstrates the iteration ID of measurement at specific RP during multiple days, and (x, y, z) are 3D coordinates of RPs (here we only use x and y). 325 We gathered the data using a Samsung Galaxy Grand-Prime smartphone and measured 75 RSS samples from 27 Wi-Fi APs for 100 consecutive seconds at each of 250 locations on the third floor of the Electrical Engineering Department of Shahid Beheshti University, which is depicted in Fig. 8.

We use Monte-Carlo cross-validation (Xu & Liang, 2001) to compare the proposed method with DNN regression (DNNR) (Félix et al., 2016), RF (Guo et al., 2018), GPR (Nabati et al., 2022; Prasad et al., 2018), and weighted KNN (Khalajmehrabadi et al., 2017a; Liu et al., 2017) in which the train and test samples are randomly selected for T times, and the mean average error (MAE) is reported to reduce the error bias which is defined as follows

$$\text{MAE} = \frac{1}{T\hat{R}} \sum_{t=1}^{T} \sum_{i=1}^{\hat{R}} \sqrt{(\hat{\mathbf{x}}_i^t - \mathbf{x}_i^t)^2 + (\hat{\mathbf{y}}_i^t - \mathbf{y}_i^t)^2}, \tag{14}$$

where $\hat{R}$ is the number of test samples, $\hat{\mathbf{x}}_i^t$ and $\hat{\mathbf{y}}_i^t$ are the real location of the i^{th}

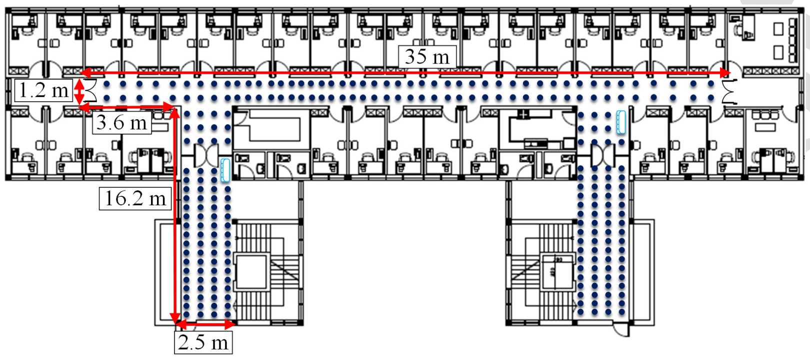


Figure 8: Floor plan of the indoor environment. The data have been collected at RPs (denoted by dots).

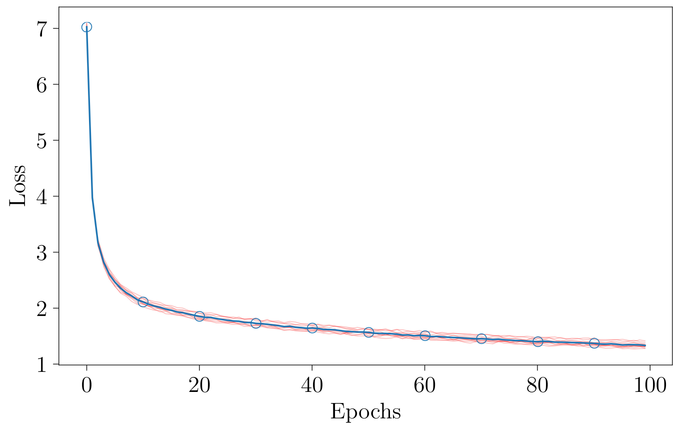
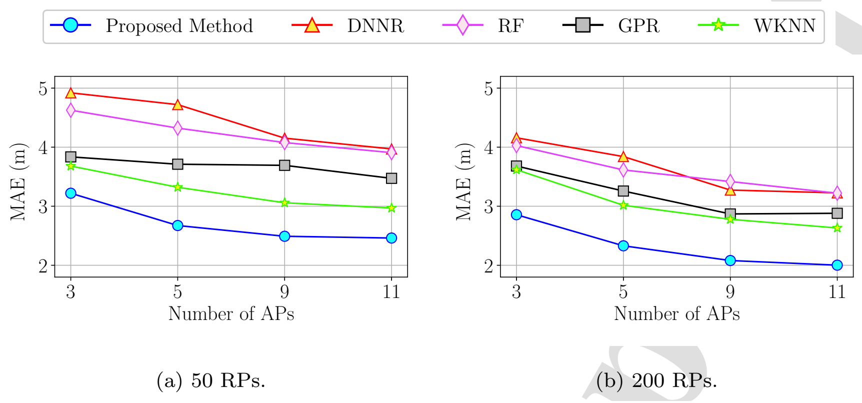


Figure 9: Training loss for the proposed DNN model with different epochs. The lines with pink color show the 10 runs results and the blue line with circle marker shows the average of these 10 runs.

user, and $\hat{\mathbf{x}}_i^t$ and $\hat{\mathbf{y}}_i^t$ are the estimated location of the i^{th} user in the t^{th} repeated time. In our experiments, we set the iteration parameter to $T = 10$. Also, the number of TPs $(\hat{R})$ equals $(250 - R) \times 75$ and $(250 - R)$ for the single available RSS sample and all RSS samples, respectively.

First, we investigate the convergence of the proposed DNN model. In the training phase, Adam optimizer is used with these parameters: $\alpha = 0.001, \beta_1 =$



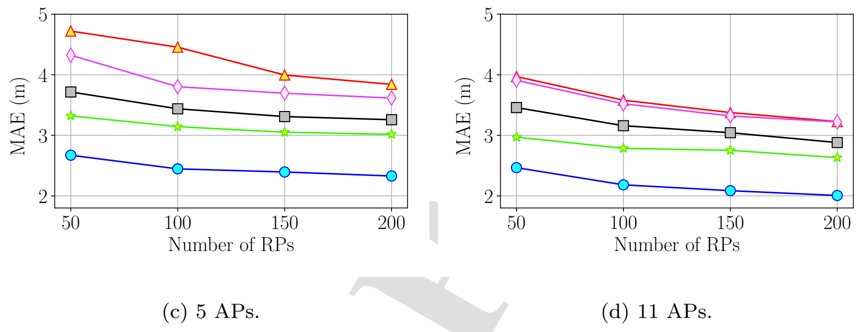


Figure 10: MAE of different methods with different experiment setups, considering a single RSS sample for each user in Testbed1.

$0.9, \beta_2 = 0.999, \epsilon = 1e-7$ and the training loss (equation (7)) has been plotted in Fig. 9 for different epochs. We have run the proposed DNN model 10 times with different initialization seeds and random training data in our collected dataset. In all these 10 runs, the number of APs and RPs are set to 5 and 50, respectively. In Fig. 9, the lines with pink color show the 10 runs results and the blue line with circle marker shows the average of these 10 runs. As can be seen, by increasing the number of epochs, the loss value decreases, which shows the convergence of the proposed DNN model.

The performance of the proposed method under different experiment setups in terms of MAE is illustrated in Fig. 10 and 11 considering the single available RSS sample and 75 RSS samples for users, respectively. It should be noted that most of the detected APs in the Testbed1 are not reliable and belong to other

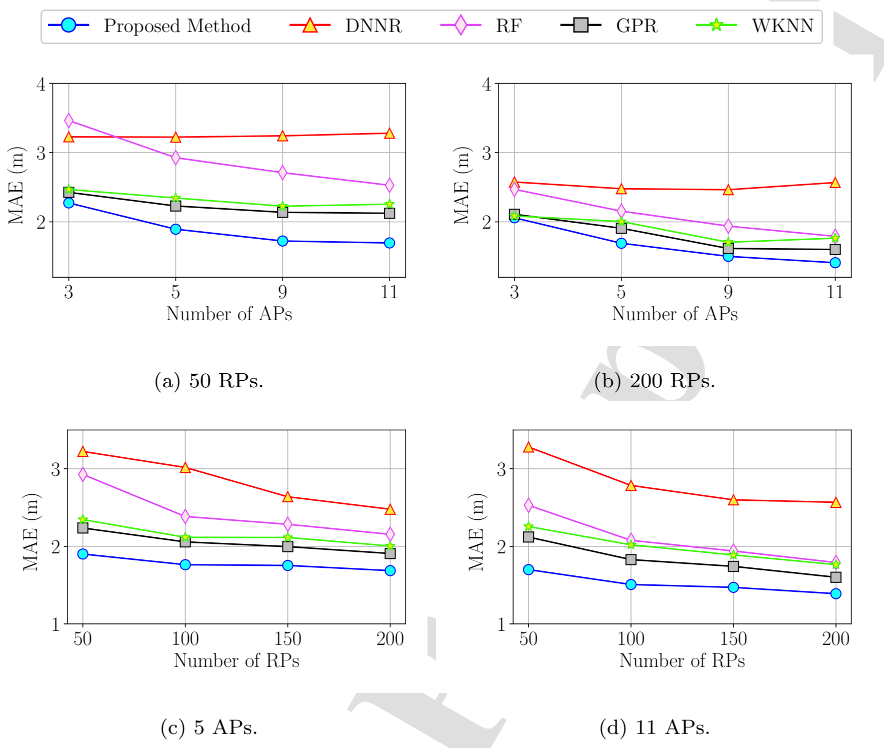
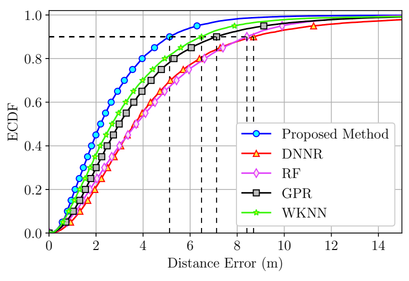
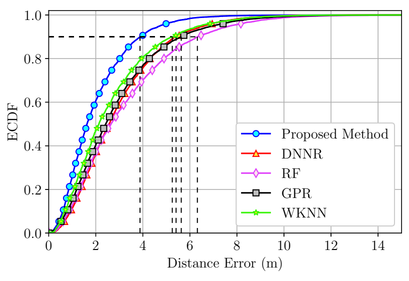


Figure 11: MAE of different methods with different experiment setups, considering sufficient RSS samples (75 RSS samples) for each user in Testbed1.

floors or buildings. Since unreliable APs do not provide useful information, they might have a negative impact on the localization accuracy (Eisa et al., 2013). Therefore, after the data collection phase, we carry out a preprocessing operation to choose the APs based on their reliability in simulations of the Testbed1. APs are selected based on a predefined proportion $\frac{k}{K} > \xi$, where k is the number of reliable samples of an AP ($\mathrm{RSS} > -70\mathrm{dBm}$), K is the number of total samples in Testbed1 (75×250), ξ is a threshold and the higher ξ chooses more reliable APs. We first calculate the value of $\frac{k}{K}$ for each AP in the dataset. Then, we choose a value for ξ to select the APs. When an AP has a large number of reliable samples in the environment, the value of k increases while K is a fixed value for all APs. If we set a large value for ξ, it means we want more reliable APs. Please also note that $0 \leq \frac{k}{K} < 1$ because the number of reliable



(a) 5 APs & 50 RPs.



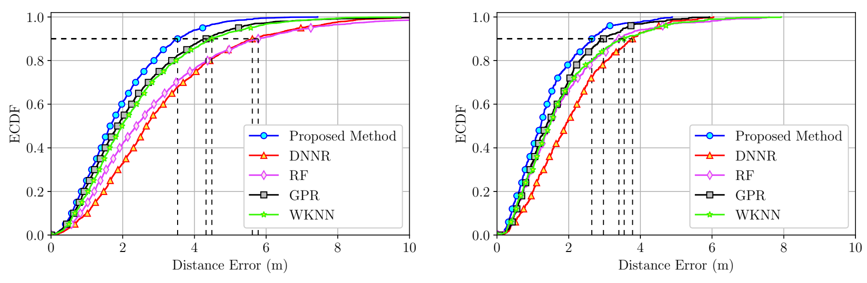
(b) 11 APs & 200 RPs.

Figure 12: Empirical cumulative distribution function (ECDF) of different methods, considering a single RSS sample for users in Testbed1.

samples of an APs (k) is always less than the total number of all APs' samples (K). The value of ξ is set to 0.6, 0.5, 0.4, and 0.3 for selection of 3, 5, 9, and 11 APs, respectively.

In Fig. 10(a) and Fig. 10(b), the number of RPs is fixed to 50 and 200, respectively, and the number of APs is changed. Also, in Fig. 10(c) and Fig. 10(d), the number of APs is fixed to 5 and 11, respectively, and the number of RPs is changed. The same explanations hold for Fig (11). As can be seen, the proposed method is the superior algorithm compared to others in all of these scenarios. Although the proposed method outperforms the accuracy of positioning in both scenarios, considering a single RSS sample and sufficient RSS samples, comparing Fig. 10 and Fig. 11 we can conclude that the proposed method is more effective for real-time applications when we do not have access to sufficient RSS samples in the online phase.

We have demonstrated the empirical cumulative distribution function (ECDF) of different algorithms in Fig. 12 and Fig. 13, considering a single RSS sample and sufficient RSS samples for users, respectively. In Fig. 12(a) the number of APs and RPs are limited to 5 APs and 50 RPs, respectively, and in Fig. 12(b) we use the full set of APs and RPs (11 APs and 200 RPs). The same explanations hold for Fig. 13. The extracted information from Fig. 12 and Fig. 13 such as MAE, accuracy at 75th percentile (Potortì et al., 2022), and accuracy



(a) 5 APs & 50 RPs. (b) 11 APs & 200 RPs.

Figure 13: Empirical cumulative distribution function (ECDF) of different methods, considering sufficient RSS samples (75 RSS samples) for users in Testbed1.

at 90th percentile is summarized in Table 1. As can be seen, the accuracy of the proposed method is the best among the popular algorithms and it is applicable for a real-time scenario where we do not have access to sufficient RSS samples in the online phase. For a fingerprint-based positioning system, it is important to have a good performance in non-ideal scenarios when we do not have a large number of RPs and RSS samples in the online phase due to the time-consuming process of data collection and sample rate of smartphones. As can be seen in Fig. 10 and 11, the error gap between the proposed method in non-ideal scenarios and others is more than in ideal scenarios, especially comparing those with single RSS samples and the full set of RSS observations; therefore, PRAs become more similar in ideal scenarios (Nabati et al., 2021; Zhu et al., 2016).

Table 1: Comparison of different algorithms in terms of accuracy at 75th percentile, accuracy at 90th percentile, and MAE for different scenarios of Fig. 12 and Fig. 13 for Testbed1.

	single RSS sample for each user						75 RSS samples for each user					
	5 APs 50 RPs			11 APs 200 RPs			5 APs 50 RPs			11 APs 200 RPs		
	75 perc.	90 perc.	MAE	75 perc.	90 perc.	MAE	75 perc.	90 perc.	MAE	75 perc.	90 perc.	MAE
Proposed method	3.55	5.12	2.67	2.69	3.88	2.00	2.59	3.53	1.90	1.82	2.64	1.38
DNNR	5.70	8.69	4.53	3.95	5.41	2.94	4.04	5.61	3.05	2.74	3.78	2.07
RF	5.94	8.41	4.32	4.43	6.31	3.22	3.87	5.77	2.92	2.41	3.40	1.79
GPR	4.77	7.12	3.70	3.86	5.63	2.88	3.01	4.32	2.23	2.12	2.97	1.60
WKNN	4.40	6.48	3.32	3.55	5.25	2.63	3.14	4.48	2.34	2.32	3.54	1.76

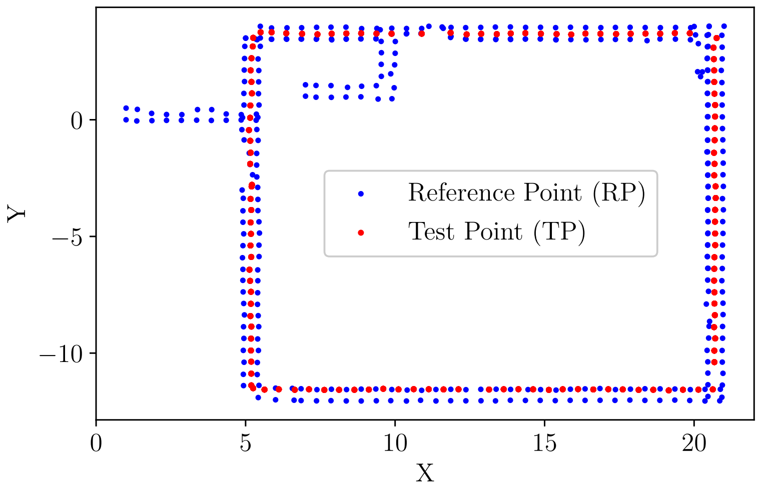
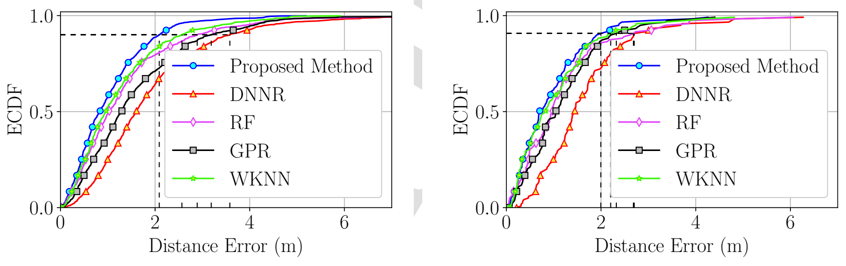


Figure 14: The locations of RPs and TPs in Testbed2.



(a) single RSS sample for each user.

(b) 10 RSS samples for each user.

Figure 15: Empirical cumulative distribution function (ECDF) of different methods in Testbed2, considering a single RSS sample for each user and sufficient RSS samples (100 RSS samples) for each user.

5.2. Testbed 2

Here we use a benchmark dataset [3] (Hoang et al., 2019) to show the robustness of our proposed algorithm. This dataset consists of 345 RPs, most of

[3]The dataset is available at: `https://ieee-dataport.org/open-access/wifi-rssi-indoor-localization`.

which have 50 RSS samples. There are 119 locations that are used as TPs, each of which has 10 RSS samples. We conduct the experiments, considering both situations in which we have a single RSS sample and sufficient RSS samples (10 RSS samples) for each user. For these two scenarios, the number of TPs equals $119 \times 10 = 1990$ and 119, respectively. Since the RPs and TPs have been separated in Testbed2, the comparisons are performed in terms of mean error (ME), which is defined as follows

$$\mathrm{ME} = \frac{1}{\hat{R}} \sum_{i=1}^{\hat{R}} \sqrt{(\hat{\mathbf{x}}_i - \mathbf{x}_i)^2 + (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2}, \tag{15}$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ are real TPs' locations, $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$ are estimated TPs' locations, and $\hat{R}$ is the number of TPs. The ECDF of algorithms has been depicted in Fig. (15) and their performances have been summarized in Table 2. As can be seen, the proposed method outperforms the accuracy of positioning compared with others in terms of ME, accuracy at 75^{th} percentile, and accuracy at 90^{th} percentile. The same conclusion of Testbed1 holds for Testbed2. The proposed method increases the accuracy of positioning in both cases when there is only a single RSS sample and also when there are sufficient RSS samples for users. In the next step, we examine the combination of the proposed state-based positioning method with the proposed DNN model. First, we should select an optimum value for the buffer size Ω and kernel width γ in (11). Since we have 119 TPs and there are 10 RSS samples at each TP, we randomly select an RSS sample at each TP to create a random RSS sample for the path. Once the path is created, we can use the algorithm with different values of Ω and γ and measure the ME, as depicted in Fig 16. The minimum ME occurs in $\Omega = 2$ and $\gamma = 11$.

After the process of choosing Ω and γ, we create another random trial path to test the algorithm. The ECDF of the model is depicted in Fig. 17 and compared with the pure DNN-based positioning method in two different scenarios (single RSS sample and all RSS samples). Also, Table 3 shows a summary performance of Fig. 17. Apart from achieving better accuracy in the state-based positioning method, we can see another important issue in Table 3; the

Table 2: Comparison of different algorithms in terms of accuracy at accuracy at 75th percentile, 90th percentile, and ME for two different scenarios of Fig. 15.

	single RSS sample for each user			10 RSS samples for each user		
	75 perc.	90 perc.	ME	75 perc.	90 perc.	ME
Proposed method	1.41	2.08	1.03	1.30	1.99	0.93
DNNR	2.39	3.54	1.88	2.02	2.69	1.60
RF	1.71	2.89	1.36	1.50	2.68	1.17
GPR	2.18	3.18	1.58	1.66	2.32	1.18
WKNN	1.65	2.56	1.23	1.53	2.20	1.02

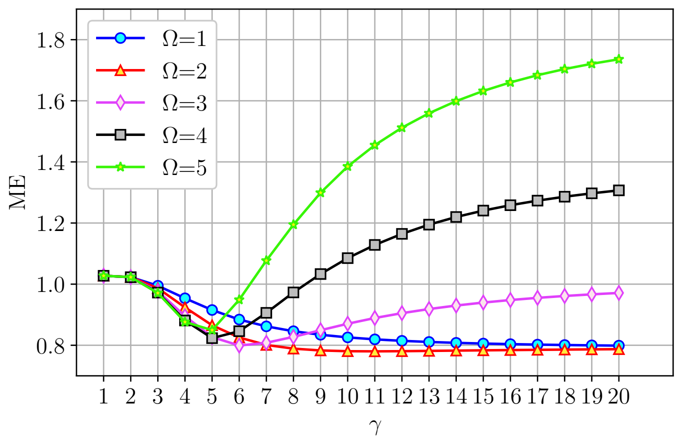


Figure 16: Mean error (ME) with different values of Ω and γ.

accuracy of the proposed method (combination of proposed DNN and proposed state-based positioning) using only a single RSS sample is very close to the accuracy of using all RSS samples. It means that with the combination of the proposed DNN-based positioning method and state-based positioning method we can handle the problem of a small sampling rate.

We measured the running time of the training and testing phases of the algorithms as stated in Table 4. For Testbed1, we used 5APs and 50 RPs for training, and for Testbed2, we used the full dataset. The training phase of

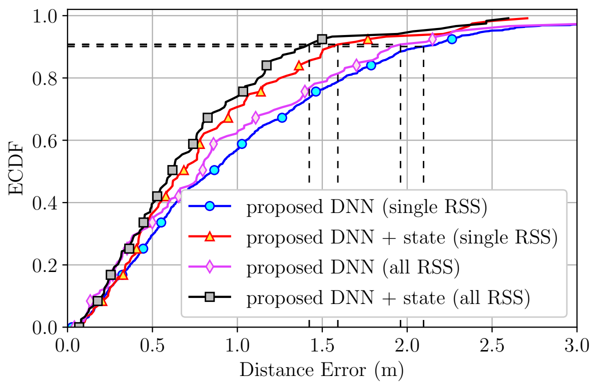
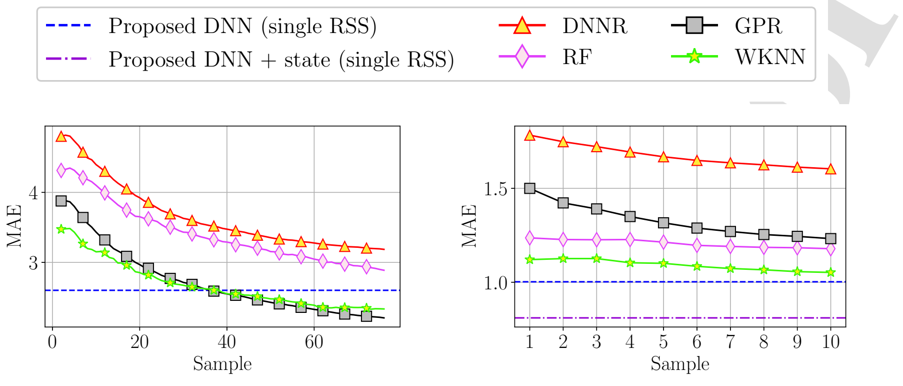


Figure 17: Empirical cumulative distribution function (ECDF) of the proposed method, in differenrt scenarios.

Table 3: Comparison of different scenarios of Fig. 17 in terms of accuracy at 75th percentile, accuracy at 90th percentile, and ME.

	single RSS sample for each user			10 RSS samples for each user		
	75 perc.	90 perc.	ME	75 perc.	90 perc.	ME
Proposed DNN	1.45	2.09	1.03	1.39	1.96	0.94
Proposed DNN + state	1.13	1.59	0.81	1.03	1.42	0.73

state-based positioning refers to finding the optimum values for buffer size Ω and kernel width γ. We used 10-fold cross-validation for finding the optimum value of neighbors in WKNN, and the training phase of WKNN refers to finding the optimum value of neighbors. In the test phase, we reported the estimation time for a single TP for all algorithms. The proposed method is in the range of microseconds to estimate the user location. At first glance, we may say that all algorithms are real-time. However, we should consider another important factor that was explained before. We cannot wait to collect uncorrelated RSS samples in the online phase. Other algorithms need to collect uncorrelated RSS samples to achieve an acceptable accuracy. We defined another column, named same accuracy time (SAT), to measure the performance of algorithms at the same accuracy. This column is filled with the information of Fig. 18. In this



(a) Testbed1. (b) Testbed2.

Figure 18: MAE of algorithms with different number of samples in two testbeds.

Table 4: Running time and same accuracy time (SAT) of different algorithms in two testbeds.

	Testbed1			Testbed2		
	train	test	SAT	train	test	SAT
proposed DNN (+state)	$15.37s$	$31.1\mu s$	$31.1\mu s$	$1.66(+0.75)s$	$87.5(+42.0)\mu s$	$129.5\mu s$
DNNR	$1.11s$	$30.0\mu s$	Inf	$2.04s$	$84.0\mu s$	Inf
RF	$0.097s$	$1.66\mu s$	Inf	$0.016s$	$1.54\mu s$	Inf
GPR	$0.146s$	$3.99\mu s$	$45s$	$1.69s$	$35.3\mu s$	Inf
WKNN	$0.21s$	$1.26\mu s$	$47.5s$	$0.27s$	$7.89\mu s$	Inf

figure, we depicted the MAE of our proposed method with only a single RSS sample, and the MAE of other algorithms is measured with a different number of samples. Once the MAE of an algorithm passes the proposed method by increasing the sample number, we measure the time. This strategy helps us to analyze the real-time performance of algorithms at the same accuracy. The SAT equals the estimation time plus the required time for collecting the RSS samples to reach the performance of proposed method. Since the test time is ignorable compared with collecting the RSS samples, we only report the required time to collect the samples. Also, we reported the test time for the proposed method in the SAT column; because we only used a single RSS sample for the proposed method.

In Testbed1, the GPR and WKNN pass the horizontal line in 36 and 38 samples, respectively. It means they need 45 and 47.5 seconds to reach the same accuracy of our proposed method with only a single RSS sample. In Testbed2, they even do not pass the horizontal lines of proposed DNN and state-based approaches. We marked the unpassed algorithms as Inf. Considering a real-time application and the fact that the user cannot wait to get uncorrelated RSS samples, the proposed method is more efficient than others because it gives the desired performance in a short-time period with only a single RSS sample. Therefore, we should consider both the nature of algorithms and the estimation time. A real-time system should be fast enough to provide the desired service. The proposed method provides an accurate localization performance for real-time applications.

6. Conclusion

We proposed a novel fingerprint-based indoor positioning algorithm, which improves the accuracy compared with others, considering both cases of a single RSS sample and sufficient RSS samples for users. Evaluation of a model by considering a single RSS sample for each user is necessary as smartphones usually cannot capture a high number of samples in a short-time period. However, most of the existing works did not consider the limitation of smartphones to provide a high sample rate for scanning the RSS values received from Wi-Fi APs. Our proposed method consists of a DNN combined with a state-based positioning method to take preceding states of the users into consideration. The reason why the proposed DNN model achieves high accuracy is that the conventional DNN-based positioning algorithm does not consider the statistical behavior of RSS samples at each RP, and two output nodes estimate the users' locations directly. However, in the proposed method, one node per RP is considered in the output layer, and pattern variation of RSS samples at each RP is obtained during the optimization process.

On the other hand, the state-based positioning method by taking the preceding states' information such as RSS values and estimated locations of the DNN model can empower the online phase limitations. The main difference between proposed state-based positioning and other Bayesian approaches such as the Kalman filter and particle filter is that it does not need additional information and hardware tool. In a typical scenario, we did not assume any restriction or limitation for the combination of the proposed positioning method. Besides, there is no need to add hardware or software tools, which makes it a general approach. The proposed method can be established with CSI data in future work, assuming that the mobile devices can capture the CSI data without additional hardware requirements in the future.

References

Afuosi, M. B., & Zoghi, M. R. (2020). Indoor positioning based on improved weighted knn for energy management in smart buildings. *Energy and Buildings*, *212*, 109754.

Aranda, F. J., Parralejo, F., Álvarez, F. J., & Paredes, J. A. (2022). Performance analysis of fingerprinting indoor positioning methods with ble. *Expert Systems with Applications*, (p. 117095).

Bai, S., Luo, Y., Yan, M., & Wan, Q. (2021). Distance metric learning for radio fingerprinting localization. *Expert Systems with Applications*, *163*, 113747.

Chen, Z., Zhu, Q., & Soh, Y. C. (2016). Smartphone inertial sensor-based indoor localization and tracking with ibeacon corrections. *IEEE Transactions on Industrial Informatics*, *12*, 1540–1549.

Dai, P., Yang, Y., Wang, M., & Yan, R. (2019). Combination of dnn and improved knn for indoor location fingerprinting. *Wireless Communications and Mobile Computing*, *2019*.

Du, X., Yang, K., & Zhou, D. (2018). Mapsense: Mitigating inconsistent WiFi signals using signal patterns and pathway map for indoor positioning. *IEEE Internet of Things Journal*, *5*, 4652–4662.

Eisa, S., Peixoto, J., Meneses, F., & Moreira, A. (2013). Removing useless aps and fingerprints from wifi indoor positioning radio maps. In *International Conference on Indoor Positioning and Indoor Navigation* (pp. 1–7).

Fang, S., Wang, C., & Tsao, Y. (2015). Compensating for orientation mismatch in robust Wi-Fi localization using histogram equalization. *IEEE Transactions on Vehicular Technology*, *64*, 5210–5220.

Félix, G., Siller, M., & Álvarez, E. N. (2016). A fingerprinting indoor localization algorithm based deep learning. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 1006–1011).

Ghari, P. M., Shahbazian, R., & Ghorashi, S. A. (2019). Maximum entropy-based semi-definite programming for wireless sensor network localization. *IEEE Internet of Things Journal*, *6*, 3480–3491.

Guo, X., Ansari, N., Li, L., & Li, H. (2018). Indoor localization by fusing a group of fingerprints based on random forests. *IEEE Internet of Things Journal*, *5*, 4686–4698.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. New York, NY, USA: Springer.

Hernández, N., Parra, I., Corrales, H., Izquierdo, R., Ballardini, A. L., Salinas, C., & García, I. (2021). Wifinet: Wifi-based indoor localisation using cnns. *Expert Systems with Applications*, *177*, 114906.

Hoang, M. T., Yuen, B., Dong, X., Lu, T., Westendorp, R., & Reddy, K. (2019). Recurrent neural networks for accurate rssi indoor localization. *IEEE Internet of Things Journal*, *6*, 10639–10651.

Homayounvala, E., Nabati, M., Shahbazian, R., Ghorashi, S. A., & Moghtadaiee, V. (2019). A novel smartphone application for indoor positioning of users based on machine learning. In *Adjunct Proceedings of the ACM International Joint Conference on UbiComp/ISW* (pp. 430–437).

Khalajmehrabadi, A., Gatsis, N., & Akopian, D. (2017a). Modern WLAN fingerprinting indoor positioning methods and deployment challenges. *IEEE Communications Surveys & Tutorials*, *19*, 1974–2002.

Khalajmehrabadi, A., Gatsis, N., Pack, D. J., & Akopian, D. (2017b). A joint indoor WLAN localization and outlier detection scheme using lasso and elastic-net optimization techniques. *IEEE Transactions on Mobile Computing*, *16*, 2079–2092.

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, *10*, 508.

Lee, S.-H., Kim, W.-Y., & Seo, D.-H. (2022). Automatic self-reconstruction model for radio map in wi-fi fingerprinting. *Expert Systems with Applications*, *192*, 116455.

Liu, H.-H., & Liu, C. (2018). Implementation of Wi-Fi signal sampling on an Android smartphone for indoor positioning systems. *Sensors*, *18*, 3.

Liu, R., Yuen, C., Do, T.-N., & Tan, U.-X. (2017). Fusing similarity-based sequence and dead reckoning for indoor positioning without training. *IEEE Sensors Journal*, *17*, 4197–4207.

Luo, R. C., & Hsiao, T. (2019). Indoor localization system based on hybrid Wi-Fi/BLE and hierarchical topological fingerprinting approach. *IEEE Transactions on Vehicular Technology*, *68*, 10791–10806.

Ma, Y., Tian, C., & Jiang, Y. (2019). A multitag cooperative localization algorithm based on weighted multidimensional scaling for passive UHF RFID. *IEEE Internet of Things Journal*, *6*, 6548–6555.

Mahfouz, S., Mourad-Chehade, F., Honeine, P., Farah, J., & Snoussi, H. (2014). Target tracking using machine learning and kalman filter in wireless sensor networks. *IEEE Sensors Journal*, *14*, 3715–3725.

Nabati, M., Ghorashi, S. A., & Shahbazian, R. (2021). Joint coordinate optimization in fingerprint-based indoor positioning. *IEEE Communications Letters*, *25*, 1192–1195.

Nabati, M., Ghorashi, S. A., & Shahbazian, R. (2022). Confidence interval estimation for fingerprint-based indoor localization. *Ad Hoc Networks*, *134*, 102877.

Nabati, M., Navidan, H., Shahbazian, R., Ghorashi, S. A., & Windridge, D. (2020). Using synthetic data to enhance the accuracy of fingerprint-based localization: A deep learning approach. *IEEE Sensors Letters*, *4*, 1–4.

Potortì, F., Torres-Sospedra, J., Quezada-Gaibor, D., Jiménez, A. R., Seco, F., Pérez-Navarro, A., Ortiz, M., Zhu, N., Renaudin, V., Ichikari, R. et al. (2022). Off-line evaluation of indoor positioning systems in different scenarios: the experiences from ipin 2020 competition. *IEEE Sensors Journal*, *22*, 5011–5054.

Prasad, K. N. R. S. V., Hossain, E., & Bhargava, V. K. (2018). Machine learning methods for RSS-based user positioning in distributed massive MIMO. *IEEE Transactions on Wireless Communications*, *17*, 8402–8417.

Silva, I., Pendão, C., Torres-Sospedra, J., & Moreira, A. (2021). Trackinfactory: A tight coupling particle filter for industrial vehicle tracking in indoor environments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (pp. 1–12).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*, 1929–1958.

Thomas, F., & Ros, L. (2005). Revisiting trilateration for robot localization. *IEEE Transactions on Robotics*, *21*, 93–101.

Wang, X., Gao, L., Mao, S., & Pandey, S. (2017). CSI-based fingerprinting for indoor localization: A deep learning approach. *IEEE Transactions on Vehicular Technology*, *66*, 763–776.

Wang, Y., Xu Yang, Yutian Zhao, Yue Liu, & Cuthbert, L. (2013). Bluetooth positioning using RSSI and triangulation methods. In *IEEE Consumer Communications and Networking Conference* (pp. 837–842).

Xie, H., Gu, T., Tao, X., Ye, H., & Lu, J. (2016). A reliability-augmented particle filter for magnetic fingerprinting based indoor localization on smartphone. *IEEE Transactions on Mobile Computing*, *15*, 1877–1892.

Xu, Q.-S., & Liang, Y.-Z. (2001). Monte carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, *56*, 1–11.

Yang, L., Chen, H., Cui, Q., Fu, X., & Zhang, Y. (2015). Probabilistic-knn: A novel algorithm for passive indoor-localization scenario. In *IEEE Vehicular Technology Conference* (pp. 1–5).

Zafari, F., Gkelias, A., & Leung, K. K. (2019). A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, *21*, 2568–2599.

Zanella, A. (2016). Best practice in RSS measurements and ranging. *IEEE Communications Surveys & Tutorials*, *18*, 2662–2686.

Zhang, C., Patras, P., & Haddadi, H. (2019). Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys Tutorials*, *21*, 2224–2287.

Zhen, X., Yu, M., He, X., & Li, S. (2018). Multi-target regression via robust low-rank learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 497–504.

Zheng, K., Wang, H., Li, H., Xiang, W., Lei, L., Qiao, J., & Shen, X. S. (2017). Energy-efficient localization and tracking of mobile devices in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, *66*, 2714–2726.

Zheng, L., Hu, B.-J., Qiu, J., & Cui, M. (2020). A deep-learning-based self-calibration time-reversal fingerprinting localization approach on wi-fi platform. *IEEE Internet of Things Journal*, *7*, 7072–7083.

Zhou, C., Liu, J., Sheng, M., Zheng, Y., & Li, J. (2021). Exploiting fingerprint correlation for fingerprint-based indoor localization: A deep learning based approach. *IEEE Transactions on Vehicular Technology*, *70*, 5762–5774.

Zhu, D., Zhao, B., & Wang, S. (2018). Mobile target indoor tracking based on multi-direction weight position kalman filter. *Computer Networks*, *141*, 115–127.

Zhu, X., Vondrick, C., Fowlkes, C. C., & Ramanan, D. (2016). Do we need more training data? *International Journal of Computer Vision*, *119*, 76–92.

ORCID:

Mohammad Nabati: **0000-0002-4847-9829**

Seyed Ali Ghorashi: **0000-0002-2910-9208**

Highlights

- Real-time positioning by considering the limitations of smartphones.
- Extracting the RSS distribution in the offline phase.
- Leveraging previous states of users in positioning.
- Showing the robustness by extensive experiments on benchmark dataset.

Mohammad Nabati: Conceptualization, Methodology, Writing Original draft, Investigation, Software, Visualization **Seyed Ali Ghorashi:** Supervision, Writing and Editing

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: