remote sensing

MDPI

Article

A Robust Wi-Fi Fingerprint Positioning Algorithm Using Stacked Denoising Autoencoder and Multi-Layer Perceptron

Rongrong Wang [1], Zhaohui Li [1], Haiyong Luo [2,3,*], Fang Zhao [1], Wenhua Shao [1] and Qu Wang [4]

1 Schoole of Software Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; wangrongrong@bupt.edu.cn (R.W.); Lizhaoh888@bupt.edu.cn (Z.L.); zfsse@bupt.edu.cn (F.Z.); shaowenhua@ict.ac.cn (W.S.)
2 Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
3 University of Chinese Academy of Sciences, Beijing 100049, China
4 School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; wangqu@ict.ac.cn
* Correspondence: yhluo@ict.ac.cn

Received: 30 April 2019; Accepted: 28 May 2019; Published: 30 May 2019

Abstract: With the increasing demand for location-based services, Wi-Fi-based indoor positioning technology has attracted much attention in recent years because of its ubiquitous deployment and low cost. Considering that Wi-Fi signals fluctuate greatly with time, extracting robust features of Wi-Fi signals is the key point to maintaining good positioning accuracy. To handle the dynamic fluctuation with time and sparsity of Wi-Fi signals, we propose an SDAE (Stacked Denoising Autoencoder)-based feature extraction method, which can obtain a robust and time-independent Wi-Fi fingerprint by learning the reconstruction distribution from a raw Wi-Fi signal and an artificial-noise-added Wi-Fi signal. We also leverage the strong representation ability of MLP (Multi-Layer Perceptron) to build a regression model, which maps the extracted features to the corresponding location. To fully evaluate the performance of our proposed algorithm, three datasets are applied, which represent three different scenarios, namely, spacious area with time interval, no time interval, and complex area with large time interval. The experimental results confirm the validity of our proposed SDAE-based feature extraction method, which can accurately reflect Wi-Fi signals in corresponding locations. Compared with other regression models, our proposed regression model can better map the extracted features to the target position. The average positioning error of our proposed algorithm is 4.24 m when there is a 52-day interval between training dataset and testing dataset. That confirms that the proposed algorithm outperforms other state-of-the-art positioning algorithms when there is a large time interval between training dataset and testing dataset.

Keywords: Indoor positioning; feature extraction; regression positioning; stacked denoising autoencoder; multi-layer perceptron

1. Introduction

Indoor positioning technology, such as Wi-Fi [1,2], magnetic [3,4], pedestrian dead reckoning [5,6] and visible light [7,8] technologies, have become increasingly more important in people's daily life, and positioning services have gradually become an indispensable mobile application. Most Wi-Fi based positioning technologies do not require deploying additional hardware because they only utilize Wi-Fi hotspots and existing wireless LANs (Wireless Local Area Networks) to obtain position estimation. At

present, due to the wide deployment and availability of Wi-Fi infrastructure, Wi-Fi fingerprint-based localization has become one of the most dominant indoor positioning techniques.

There are two main types of Wi-Fi-based indoor positioning technologies: RSSI (Received Signal Strength Indicator)-based ranging positioning algorithm [9–11], and fingerprint-based positioning algorithm [12–14]. The RSSI-based ranging positioning algorithm [11] usually adopts the received Wi-Fi signal to estimate the distance between the target (its location is unknown) and the access point (its location is known) using the wireless radio signal propagation model, and then estimates the target position using trilateration or multilateration methods. The fingerprint-based positioning algorithm [14] adopts the signal matching algorithm to estimate the user location. It first collects environmental Wi-Fi signals and constructs a Wi-Fi fingerprint database during the offline phase. During the online positioning phase, the fingerprint-based positioning algorithm compares the current Wi-Fi observation with the recorded fingerprint in the database to obtain the target position using the optimum matching criterion. Compared with the fingerprint-based positioning algorithm, the RSSI-based ranging positioning algorithm struggles to meet high positioning accuracy due to the complex multipath and dynamic characteristics of signal propagation in indoor environments [15,16].

Current Wi-Fi fingerprint-based indoor localization mainly adopts either deterministic or probabilistic techniques [17–20]. The deterministic Wi-Fi positioning methods employ different deterministic machine learning algorithms to estimate the target location based on the shortest distance (such as Euclidean distance) criterion, such as KNN (K-NearestNeighbor) [21–23], linear discriminant analysis [24], and SVM (Support Vector Machine) [25]. The probabilistic Wi-Fi positioning methods use the posterior probability calculated by probabilistic inference methods [26,27] to estimate the target location. By calculating the conditional posterior probability using Bayesian estimation method in a previous study [19], the target location was obtained based on the maximum posterior probability criterion.

Though much work has been done on the Wi-Fi fingerprint-based indoor localization using various traditional machine learning methods, the localization accuracy still does not meet the high-accuracy and robust requirement of indoor location-based services [28]. With the rapid development of deep learning technology, some researchers have attempted to use deep learning methods in Wi-Fi positioning. One important advantage of the deep-learning-based Wi-Fi positioning method is that it can automatically filter the raw Wi-Fi observation data, extract reliable features, and build internal representations from dynamic Wi-Fi signals without additional devices or human intervention [29]. To handle the sparseness and volatility problem of the Wi-Fi signal, Khatab [30], Xu [31], and Kim [32] tried using the AE (Autoencoder) method to learn the feature representation from the raw Wi-Fi signal. Kim [33] and Nowicki [34] utilized a SAE (Stacked Autoencoder) to extract features for buildings and floors identification, and obtained a relatively desirable positioning performance on the UJIIndoorLoc dataset (http://indoorlocplatform.uji.es/databases/get/1/). By transforming the original Wi-Fi signal into an image form, Wang [35], Mao [36], and Shao [37] introduced the convolutional neural network to indoor positioning. Wang [38] and Hsieh [39] attempted to construct a recurrent neural network for indoor positioning.

Because the above-mentioned positioning methods take the target positioning as a classification problem, the location estimation results are discrete and the positioning accuracy relies on the density of collected fingerprints. To improve the smoothness and robustness of positioning results, some regression positioning algorithms are applied, such as support vector regression [40,41] and Gaussian process regression [42].

Though the previously-described positioning algorithms can obtain good positioning accuracy when there is a short time interval between the time of collecting training data and the time of collecting testing data, the positioning accuracy of these algorithms declines dramatically when the collection time of training data and testing data is separated by a large time interval. To maintain good positioning performance, these algorithms often need to periodically collect new samples and train the positioning model, which is time-consuming and labor-intensive. To solve the dynamic fluctuation with time

of Wi-Fi signals, this paper proposes a robust Wi-Fi fingerprint positioning algorithm using a SDAE (Stacked Denoising Autoencoder) [43] and a MLP (Multi-Layer Perceptron) [44]. The SDAE is used to extract time-independent features, and the MLP is used to find well-behaved mapping functions by constructing a reasonable regression model. In our proposed algorithm, we train the SDAE and MLP using the training dataset in a certain period.

The main contributions of this paper are summarized as follows:

- An SDAE-based robust feature extraction method is proposed. To extract the time-independent and robust features of the raw Wi-Fi data by using the SDAE, we design a deep neural network structure with three hidden layers, and the input data of each hidden layer adds reasonable noise. We use the layer-by-layer greedy training method to train the SDAE model.
- An MLP-based regression positioning method is proposed. By taking advantage of the Universal Approximation Theorem [45] and the fast training speed of MLP, we build a MLP-based regression model with nine hidden layers to obtain a good mapping function.
- Our proposed algorithm is fully evaluated using three datasets, which represent three different classical scenarios. We also compare our proposed localization algorithm with other localization algorithms. Extensive experimental results demonstrate that our proposed algorithm obviously outperforms the comparative localization algorithms when the Wi-Fi data covers longer time intervals.

2. Materials and Methods

2.1. System Overview

Our proposed positioning algorithm is comprised of four main modules, i.e., the data collection module, preprocessing module, feature extraction module, and MLP-based regression positioning module. The overall structure of our proposed positioning algorithm is shown in Figure 1. The data collection module collects Wi-Fi data using mobile devices. The preprocessing module is responsible for normalizing the Wi-Fi data, and then constructing the fingerprint database. The feature extraction module extracts robust and time-independent features using the SDAE method. The MLP-based regression positioning module employs the extracted features to estimate position.

2.2. Data Preprocessing

The data preprocessing aims to produce reasonable input for the feature extraction module by handling the raw Wi-Fi data. There are three steps in this phase. Firstly, we compensate the missing observation for those locations without collecting Wi-Fi signals. Considering the common range of raw RSSI observation is −110 dBm to 0 dBm, we set the missing RSSI value of the Wi-Fi signal to −110 dBm. After handling all the missing Wi-Fi observations, the range of the whole RSSI data is (−110 dB, 0 dB). Then, we normalize the raw RSSI data to the range between 0 and 1 using Equation (1), and obtain normalized data. By adopting this normalization operation, the distribution of normalized Wi-Fi data is unbiased and low variance. Directly using the raw asymmetry Wi-Fi observation data may lead to the network model training failure.

$$rssi_i = \frac{rssi_i - \text{min\_rssi}}{\text{max\_}rssi - \text{min\_rssi}} \tag{1}$$

where $rssi_i$ represents the RSSI value of the i-th Wi-Fi, min_rssi represents the smallest RSSI value, and max_$rssi$ represents the largest RSSI value.

Finally, we construct a fingerprint database, as shown in Equation (2), using the normalized Wi-Fi data.

$$\left[\begin{array}{ccc} x_1^1 & \cdots & x_1^n \\ \vdots & \vdots & \vdots \\ x_i^1 & x_i^j & x_i^n \\ \vdots & \vdots & \vdots \\ x_m^1 & \cdots & x_m^n \end{array} \left(\begin{array}{cc} target_{x1} & target_{y1} \\ \vdots & \vdots \\ target_{xi} & target_{yi} \\ \vdots & \vdots \\ target_{xm} & target_{xm} \end{array} \right)\right] \tag{2}$$

where n represents the number of features, m represents the number of samples, x_i^j represents the j-th feature of the i-th sample, $(target_x, target_y)$ represents the coordinate of the target position, $\left(target_{xi}, target_{yi}\right)$ represents the coordinate of the i-th sample.

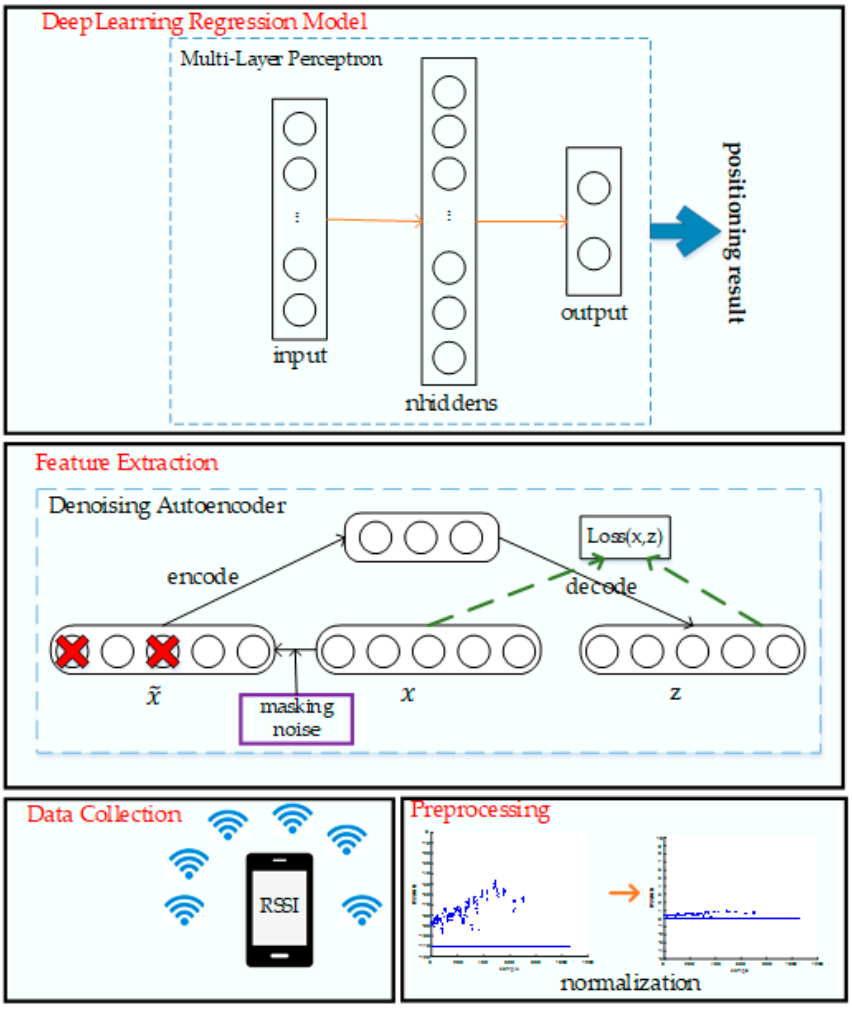


Figure 1. Overall structure of our proposed positioning algorithm.

2.3. Feature Extraction Based on the SDAE

The RSSI value of the Wi-Fi signal in the online stage may deviate from the initial fingerprint firstly collected in the offline stage. Figures 2 and 3 correspond to a teaching building and office building, respectively. As shown in Figures 2a and 3a, the two distributions of RSSI measurement sequences are different at different times in a specific location. The averaged RSSI value changes by about 9 dBm when there is a 10-day interval, as Figure 2b shows. The averaged RSSI value changes

by about 3.3 dBm when there is a 50-day interval, as shown in Figure 3b. A previous study [18] also reported that the Wi-Fi signal fluctuates over time

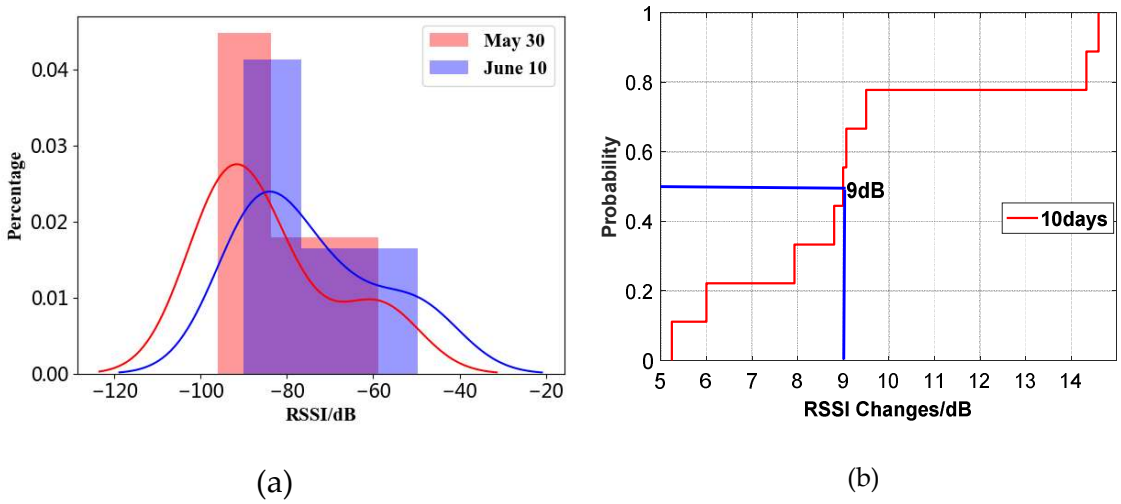


Figure 2. The dynamic fluctuation of Wi-Fi signal with time in a teaching building. (**a**) RSSI data distribution of different Wi-Fi signals at different times in a specific location. (**b**) RSSI changes of different Wi-Fi signals over a 10-day period in a specific location.

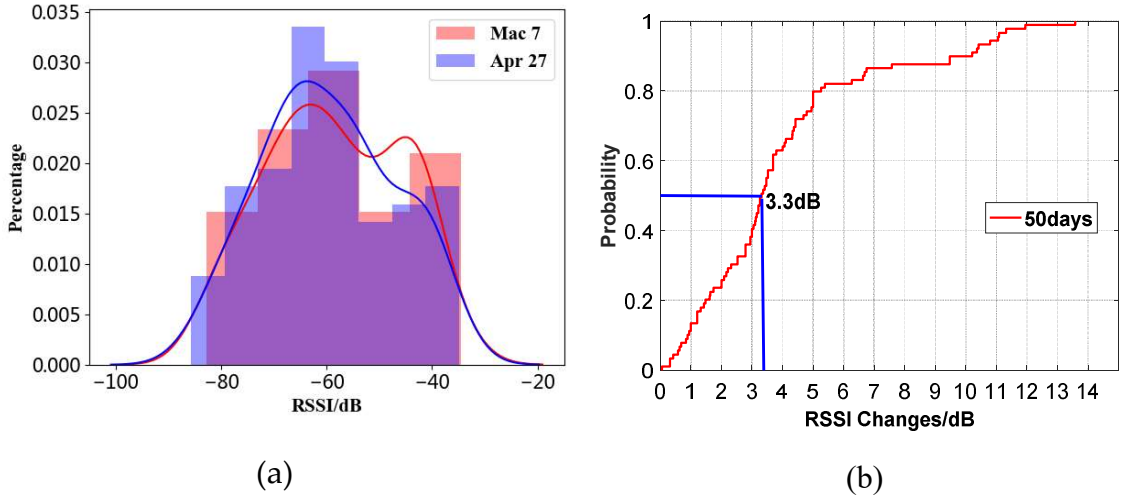


Figure 3. The dynamic fluctuation of Wi-Fi signal with time in an office building. (**a**) RSSI data distribution of different Wi-Fi signals at different times in a specific location. (**b**) RSSI changes of different Wi-Fi signals over a 50-day period in a specific location.

The Wi-Fi signal fluctuation with time results in a negative influence on the positioning performance. To enhance the accuracy and robustness of positioning, we employ the SDAE to extract the robust and time-independent features from the raw Wi-Fi signal observation.

We think that the fundamentals that the reason the SDAE-based Wi-Fi feature extraction can enhance the accuracy and robustness of positioning may lay below: (1) the SDAE adds noise to the original Wi-Fi data, enabling it to approach the distribution of the new Wi-Fi data, which has a large time interval from the original Wi-Fi data; (2) the SDAE reconstructs the original Wi-Fi data from the corrupted Wi-Fi data and extracts features, and the extracted features represent the essential distribution of the Wi-Fi signal. An experimental result shown in Figure 4 demonstrates that the correlation of RSSI samples of the original Wi-Fi data and the new-collected Wi-Fi data (there is a

10-day interval) at the same position is higher using the SDAE-based feature extraction than that without using the SDAE-based feature extraction.

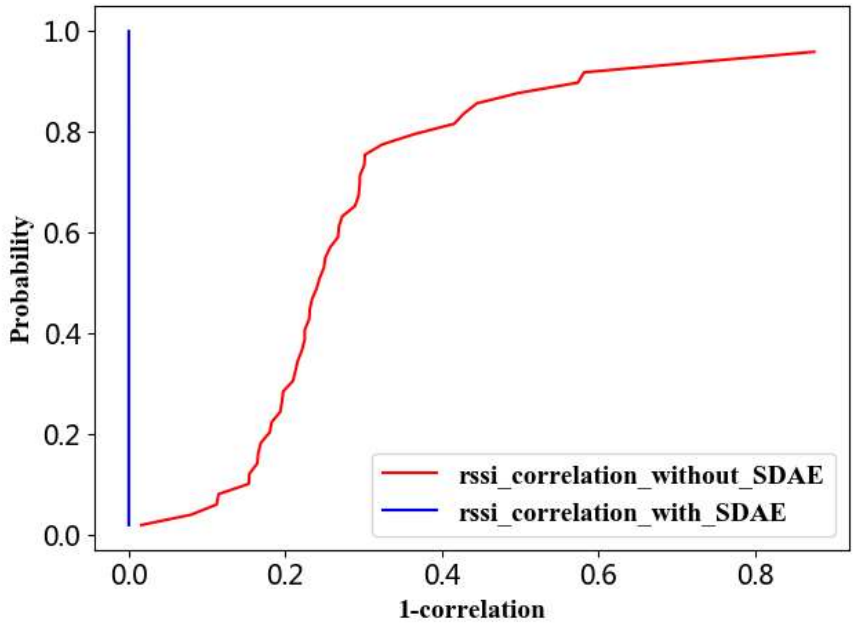


Figure 4. Correlation of RSSI samples of the original Wi-Fi data and the new-collected Wi-Fi data at the same position before SDAE and after SDAE.

The neural network structure and parameters of the proposed feature extraction method are shown in Figure 5. These consist of three stacked DAEs (Denoising Autoencoders) [46], and each DAE (Denoising Autoencoder) includes three parts, namely, the noise-added layer, encoder layer, and decoder layer. The noise-added layer obtains corrupted Wi-Fi data by adding masking noise to the original Wi-Fi data. In the first DAE, we set the noise parameter to 0.4, which randomly chooses neurons to drop out and removes them from the input layer temporarily. We produce a random seed for the disconnection operation, so the dropped neurons are definite. Noise parameters of the second DAE and the third DAE are 0.5 and 0.6, respectively. The encoder layer maps the corrupted Wi-Fi data into hidden representation, and the neurons of the encoder layer in each DAE are 256, 128, and 64, respectively. Finally, we extract 64 robust features using the SDAE. To increase nonlinearity, Relu [47] is employed in each encoder layer. The decoder layer maps the hidden representation back to a reconstruction of the original Wi-Fi data.

More specifically, the red rectangle part in Figure 5 is the second DAE in the SDAE. The dimension of the original input x is 256, and the Dropout layer Drop(·) obtains noise_x by adding masking noise to x. The Encoder layer Enc(·) obtains Enc_x by encoding noise_x. The Decoder layer Dec(·) obtains $\widetilde{x}$, whose dimension becomes 256 again. The DAE minimizes the reconstruction error L_{DAE}, as Equation (7) shows, by training the Encoder and Decoder.

$$\mathrm{x} = \left\{x^1, x^2, x^3, \ldots, x^{256}\right\} \tag{3}$$

$$noise\_x = \mathrm{Drop}(\mathrm{x}) \tag{4}$$

$$\mathrm{Enc\_x} = \mathrm{Enc}(\mathrm{noise\_x}) \tag{5}$$

$$\widetilde{x} = \mathrm{Dec}(\mathrm{Enc\_x}) \tag{6}$$

$$L_{DAE}(x, \widetilde{x}) = \frac{1}{2M}(x - \widetilde{x})^2 \tag{7}$$

where M represents the number of samples.

The SDAE is trained by the layer-by-layer greedy training method, in which the output of a DAE in the lower layer is used as the input of the DAE in the higher layer, and the SDAE completes the training task until all DAEs in the SDAE are trained. The Encoders and Decoders in the SDAE are trained to minimize the reconstruction error between the uncorrupted Wi-Fi data and reconstructed Wi-Fi data. The SDAE can extract more essential and robust features because it reconstructs the original Wi-Fi data even with the presence of high noise levels.

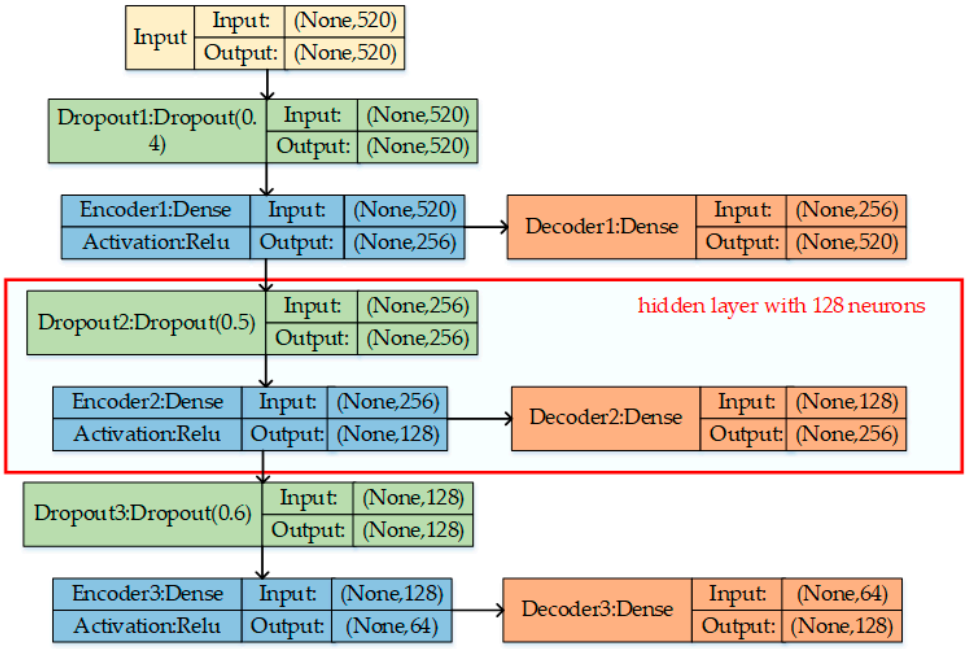


Figure 5. The SDAE network structure.

2.4. Regression Model Using MLP

To avoid dense fingerprint collection and maintain training features for each location grid, we employ a MLP-based regression model to estimate target location, which can improve smoothness and robustness of positioning results.

The MLP-based regression model is constructed by adopting the Universal Approximation Theorem [45], which possesses fast training speed. MLP defines a mapping function, as shown in Equation (8), and obtains the best function approximation by learning the value of the parameter θ.

$$y = f(x; \theta) \tag{8}$$

where y is target position, x is the Wi-Fi sample, and θ represent weight parameters in MLP.

The neural network structure and parameters of the proposed MLP-based regression model are shown in Figure 6. The neurons of the input layer in the network structure are set to 64, which are the features extracted by the SDAE algorithm. Our designed network structure includes nine hidden layers, and the neurons are set to 128, 256, 512, 256, 128, 64, 32, 16, and 8, respectively. To increase nonlinearity, we use the Tanh as the activation function for each hidden layer. We adopt the BatchNormalization [48] layer between the hidden layers because this layer makes the input of each hidden layer have the same distribution, which can speed up the convergence process. Furthermore, it plays a regularization role and can mitigate the overfitting problem. Since the target value of the mapping function in MLP is the position (x, y), the neurons in the output layer are set to 2. We exploit the sigmoid function as the activation function of the output layer because the target value in the dataset has been normalized and the sigmoid function maps the output of the last hidden layer to (0, 1). There are nine mixing layers (each mixing layer consists of a Dense layer and a BatchNormalization layer) in our proposed network structure, and the MLP network can approximate a function with arbitrary precision when the network

structure contains enough hidden neurons. The hidden layer weights are updated by minimizing the loss function $L_{regression}$ using the back-propagation algorithm, and the proposed network structure in this module is determined according to the loss function value of the validation dataset.

$$L_{regression} = \frac{1}{M}\sum_{i=1}^{M}(y_i - f(x_i))^2 \tag{9}$$

$$\text{sigmoid(x)} = \frac{1}{1+e^{-x}} \tag{10}$$

where M represents the number of samples, y_i represents the true position of the i-th sample, $f(x_i)$ represents the prediction position of the i-th sample.

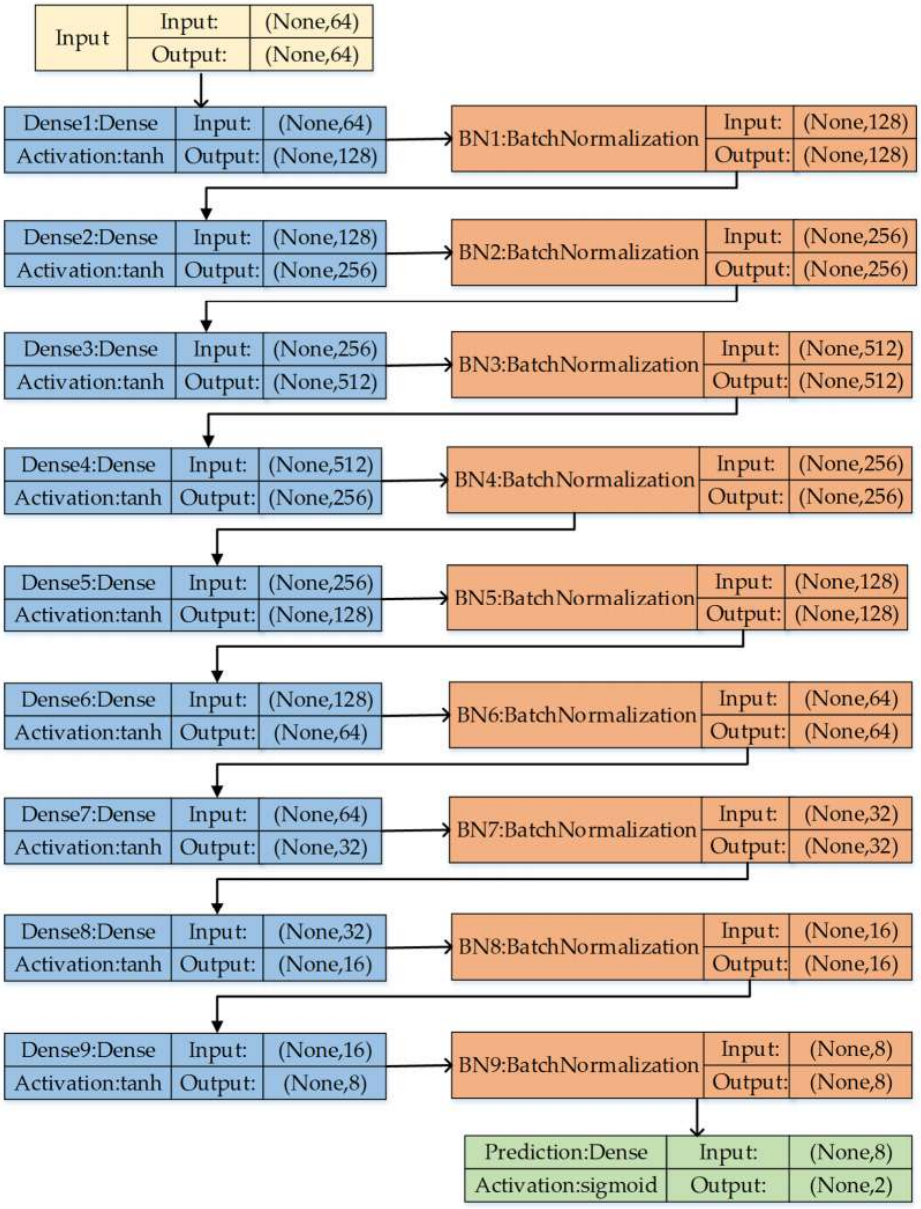


Figure 6. Regression model network structure based on MLP.

Our proposed algorithm is implemented using Keras. Table 1 summarizes the hyperparameter values in our proposed algorithm.

Table 1. Hyperparameter values in our proposed SDAE and MLP.

Parameter	SDAE	MLP
Batch size	30	100
Activation	Relu [47]	Tanh
Optimizer	Adam [49]	RMSprop [50]
Learning rate	0.1	0.0008
Epochs	30 (first DAE), 20 (second and third DAEs)	200
Loss function	MSE	MSE

Algorithm 1 (As Figure 7 shows) describes the general process of our proposed robust Wi-Fi fingerprint positioning algorithm using SDAE and MLP.

Algorithm 1. robust Wi-Fi fingerprint positioning algorithm using SDAE and MLP

```
Input: training dataset, validation dataset, testing dataset
Output: positioning result (x, y)
// Data preprocessing
Normalize the training dataset, validation dataset and test dataset according to Equation (1).
// Feature extraction
Build SDAE_model as shown in Figure 5
new_training_dataset = training dataset
new_validation_dataset = validation dataset
for sdae_layer in all hidden layers of SDAE_model do
    for each training epoch do
        new_training_dataset is used to train sdae_layer, and the training process is monitored by the new_validation_dataset.
    new_training_dataset = sdae_layer.predict(new_training_dataset)
    new_validation_dataset = sdae_layer.redict(new_validation_dataset)
//Model training
Build regression_model as shown in Figure 6
train_features = SDAE_model.predict(training dataset)
validation_features = SDAE_model.predict(validation dataset)
num_epoch = 0
for each training epoch do
    training features are used to train regression_model, and the training process is monitored by the validation features.
    if current_epoch_val_loss – previous_epoch_val_loss <= 1e-7 do
        num_epoch += 1
        if num_epoch == 10 do
            lr = lr * factor
// Model prediction
test_features = SDAE_model.predict(testing dataset)
test_positioning_result = regression_model.predict(test_features)
return test_positioning_result
```

Figure 7. The pseudo code of the robust Wi-Fi fingerprint positioning algorithm using SDAE and MLP.

2.5. Tree-Fusion-Based Regression Model

Recently, tree models (such as XGBoost and LightGBM) [51–54] have been widely used in various problems and have achieved good performances. Inspired by this idea, in this section, we construct and implement a tree-fusion-based regression model and use it as a localization comparison.

The overall framework of our proposed tree-fusion-based regression model is shown in Figure 8. Firstly, we train AdaBoost, RandomForest, and KernelRidge as three meta-learners, then utilize these meta-learners to obtain a new training dataset. Secondly, we utilize the new training dataset to train the secondary-learner GBDT (Gradient Boosting Decision Tree), and then utilize the secondary-learner to obtain the stacked model predictions in Figure 8. Thirdly, we utilize the original training dataset to train XGBoost and LightGBM, and the two single models are used to obtain the testing dataset outputs, respectively, which are the XGBoost model predictions and LightGBM model predictions in Figure 8. The weights of the stacking model, XGBoost, and LightGBM are configured to w3, w1, and w2, respectively. Finally, the final prediction in Figure 8 is obtained by weighted average (as Equation (11) shows).

$$\text{final\_prediction} = P_{stack}*\text{w3} + P_{xgb}*\text{w1} + P_{lgb}*\text{w2} \tag{11}$$

where P_{stack} is the stacked model predictions, P_{xgb} is the XGBoost model predictions, and P_{lgb} is the LightGBM model predictions.

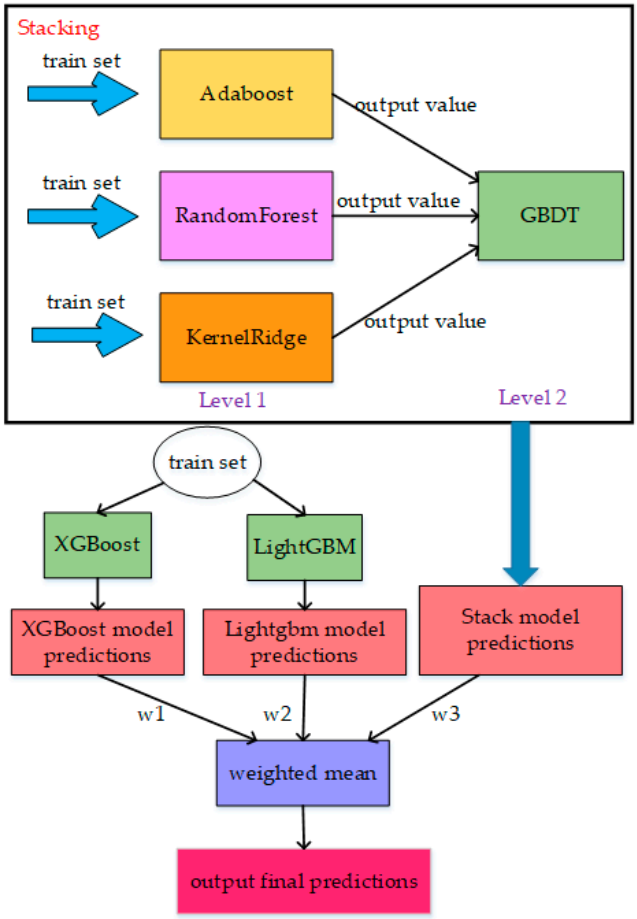


Figure 8. The overall framework of the tree-fusion-based regression model.

3. Experiments and Evaluation

In this section, we adopt three datasets to evaluate our proposed algorithm, and these datasets represent three different typical scenarios, namely spacious area (i.e., teaching building) with time

interval, complex area (office building) without time interval, and complex area (office building) with time interval. Several experiments are conducted to evaluate our proposed positioning algorithm. We also compare our proposed algorithm with other state-of-the-art localization algorithms.

3.1. Datasets

Besides the public UJIIndoorLoc dataset (named Dataset1 in this paper), we also collect Wi-Fi signals from 20 sampling points in our laboratory area. The distance between adjacent sampling points is 5 m or more. We collect Wi-Fi signals three times in these locations, which are recorded as the training dataset, validation dataset, and testing dataset, respectively. The sampling durations of each sampling point in the training dataset, validation dataset, and testing dataset are 12 s, 9 s, and 6 s, respectively. The collection interval of the training dataset and validation dataset is about 13 min. The collection interval of the validation dataset and testing dataset is 6 min. There is no time interval between the training dataset and testing dataset. This dataset is recorded as Dataset2. The sampling area is approximate 40×30 m^2. The positions of sampling points are shown in Figure 9.

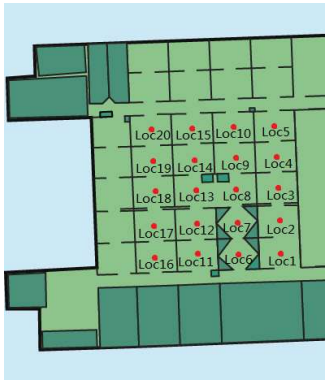


Figure 9. The sampling positions of Dataset2 in our laboratory.

We collect Wi-Fi signals from 57 sampling points in the laboratory area. The distance between sampling points is 5 m or more. Firstly, we collect Wi-Fi data twice in these locations, which are record as the training dataset and validation dataset, respectively. The sampling durations of each sampling point in the training dataset and validation dataset are 14 s and 9 s, respectively. For the testing dataset, we collect three sub-datasets, and record them the as sub-test1, sub-test2, and sub-test3, respectively. The sampling durations of these sub-datasets are 6 s. The collection intervals between these sub-datasets and the training dataset are one day, eleven days, and fifty-two days, respectively, and all sub-datasets are collected at 3 pm. The sampling area is approximate 40×60 m^2. This dataset is recorded as Dataset3. The position of sampling points is shown in Figure 10.

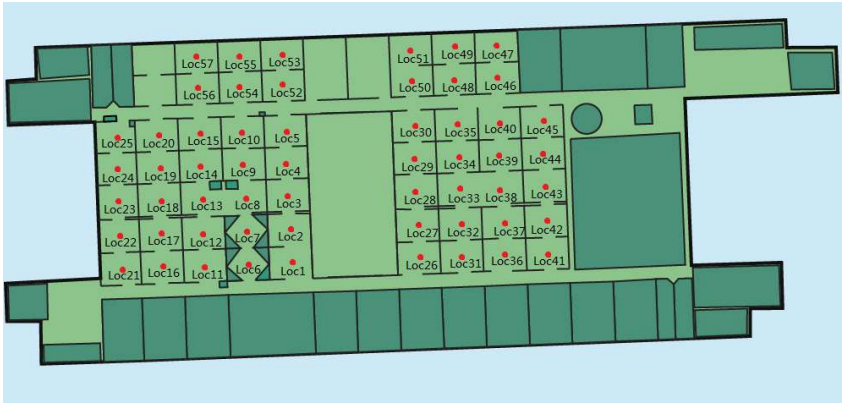


Figure 10. The sample position of the Dataset3 in our laboratory.

The sample format of these three datasets at a certain sampling point (x, y) is as shown in Equation (12).

$$s = \{rssi_0, rssi_1, \ldots, rssi_n, x, y\} \tag{12}$$

where n represents the number of Wi-Fi AP, and (x, y) represents the position of the sample s, $rssi_i$ represents RSSI value of the i-th Wi-Fi AP.

The environment of the experimental area is shown in Figure 11.

(a) (b)

Figure 11. The experimental environment: (**a**) corridor; (**b**) work-station.

3.2. Effect of SDAE Feature Extraction

To evaluate the effect of the SDAE-based feature extraction method for localization performance, we use the Dataset1 to train our proposed MLP-based regression model with feature extraction operation and without feature extraction operation. The positioning accuracy comparison with or without SDAE-based feature extraction operation is shown in Figure 12. It can be seen from Figure 12 that using the SDAE-based feature extraction method can obtain higher localization accuracy, which confirms that the SDAE can extract the robust and time-independent Wi-Fi fingerprint features from the original dynamic Wi-Fi dataset, and using the features obtained by the SDAE method can improve the positioning accuracy.

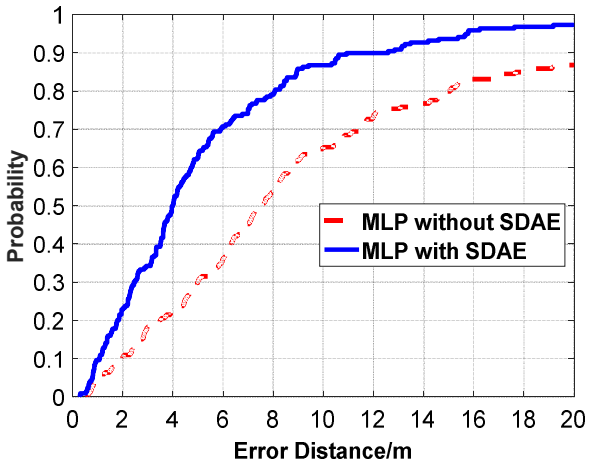


Figure 12. Comparison of CDF (Cumulative Distribution Function) positioning errors between the algorithm with feature extraction and the algorithm without feature extraction.

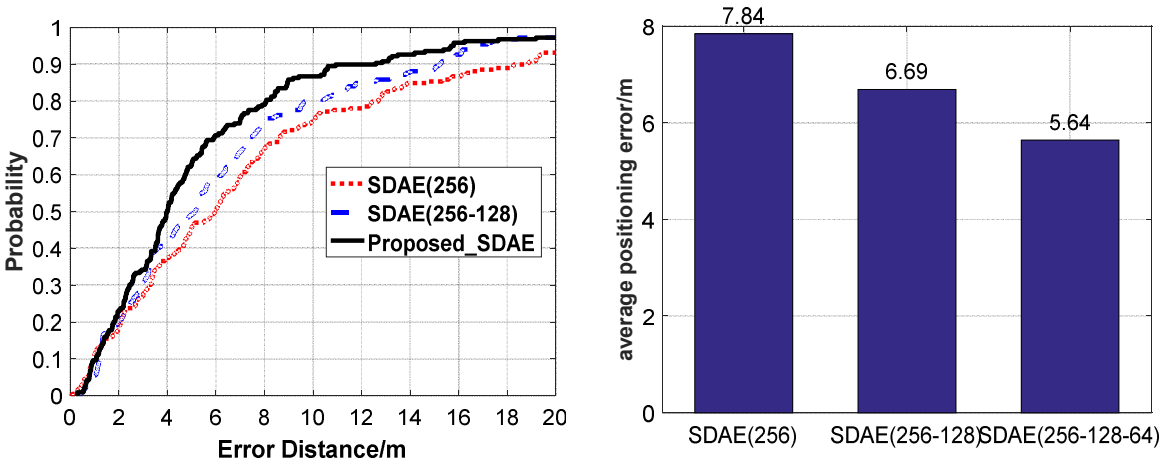
3.3. Positioning Performance of the Proposed Algorithm

3.3.1. Performance of Our Proposed Algorithm under Different Parameters

In this section, we evaluate the performance of our proposed algorithm on Dataset1 with different parameters.

- Influence of different numbers of hidden layers in SDAE network structure

In addition to the SDAE structure proposed in this paper (256-128-64), we also build two comparative SDAE network structures, i.e., one SDAE network structure only contains one hidden layer with 256 neurons, and the other SDAE network structure contains two hidden layers (256-128). The positioning accuracy is shown in Figure 13. We can see from Figure 13 that using more hidden layers can obtain higher positioning accuracy, which demonstrates that adopting more a complex SDAE network structure can better represent the robust and time-independent Wi-Fi fingerprint.



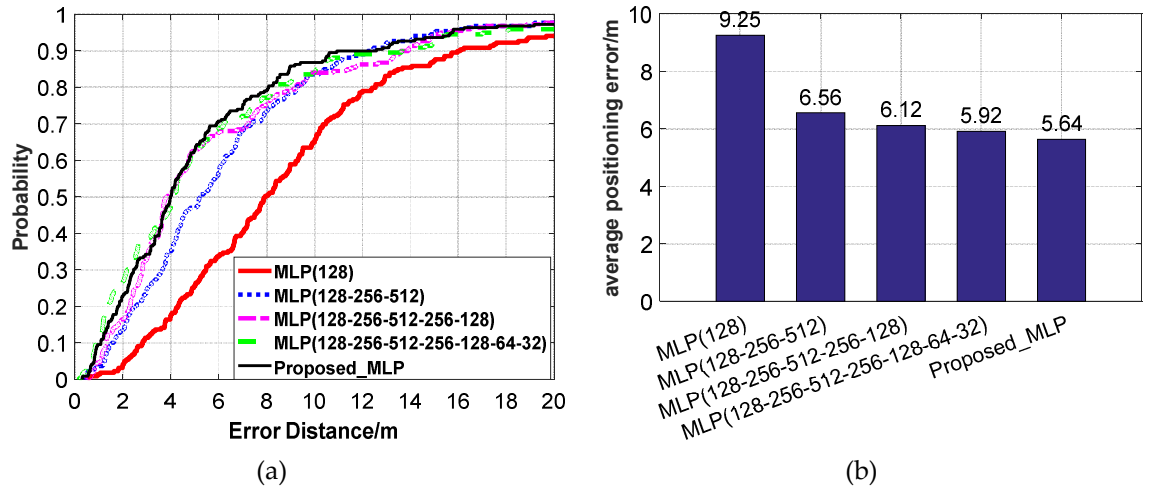
(a) (b)

Figure 13. The positioning accuracy using different SDAE network structures. (**a**) CDF positioning errors. (**b**) Average positioning errors.

- MLP regression model comparison in different parameters

(1) Performance using different MLP network structures

Figure 14 compares the positioning accuracy using different MLP network structures. According to the CDF positioning errors (shown in Figure 14a) and average positioning errors (shown in Figure 14b), the MLP-based regression model can obtain smaller localization error when adopting a more complicated network structure. Using our proposed MLP network structure (128-256-512-256-128-64-32-16-8-2) can obtain the best positioning accuracy.



(a) (b)

Figure 14. The positioning accuracy using different MLP network structures. (**a**) CDF positioning errors. (**b**) Average positioning errors.

(2) Performance using different activation functions

From Figure 15, we can obviously see that using the Tanh activation function outperforms the Linear and Relu activation functions. The MLP-based regression model obtains an average positioning error of 5.64 m when using the Tanh activation function, which is 28.1% less than using Linear activation function and 17.3% less than using Relu activation function, respectively.

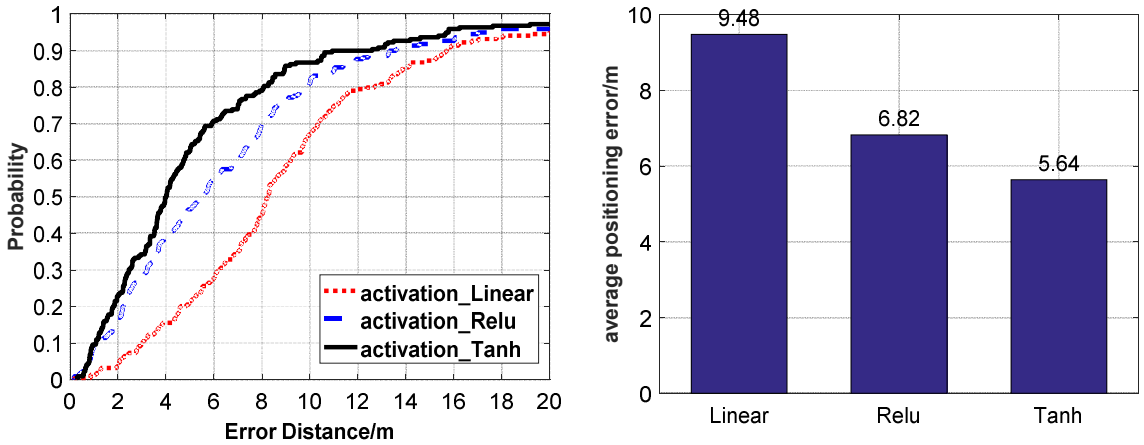


Figure 15. The positioning accuracy using different activation functions. (**a**) CDF positioning errors. (**b**) Average positioning errors.

(3) Performance using different MLP optimizers

In addition to RMSprop used in our proposed algorithm, we also try using other optimizers, such as Adamax, Adam, Nadam, Adadelta, Adagrad, and SGD. For each optimizer, we optimize the learning rate that corresponds to the best positioning performance. The positioning accuracy using these optimizers is shown in Figure 16. The MLP using RMSprop obtains the best positioning accuracy. As shown in Figure 16a, the localization error using the MLP-based regression model for 80% of the testing dataset falls between 8 m and 9 m, except the model using Nadam, in which the model using RMSprop obtains approximately 8 m. The MLP-based regression model using RMSprop obtains an average positioning error of 5.64 m, which is 26.6% less than using Nadam, which obtains the worst positioning performance and 8.3% less than using the SGD, which obtains the highest positioning accuracy, except using RMSprop, as shown in Figure 16b.

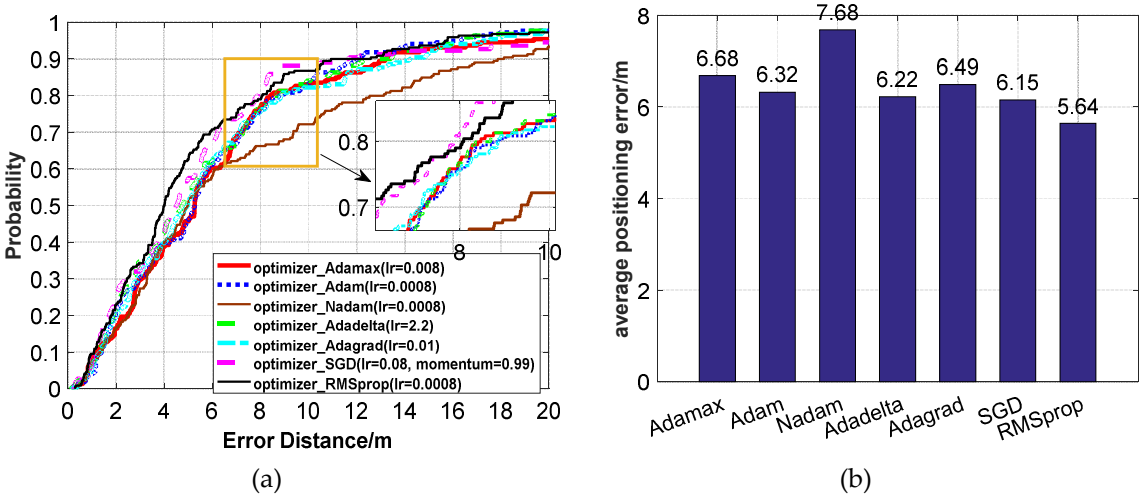


Figure 16. The positioning accuracy using different optimizers. (**a**) CDF positioning errors. (**b**) Average positioning errors.

(4) Performance using different MLP epoch

The loss function declining curve with epoch on the validation dataset is plotted in Figure 17. As can be seen from Figure 17, when epoch reaches 150, val_loss approximates the minimum value, but there is still slight vibration. When epoch reaches 200, val_loss becomes stable. Therefore, The epoch 200 is the appropriate epoch in our proposed algorithm.

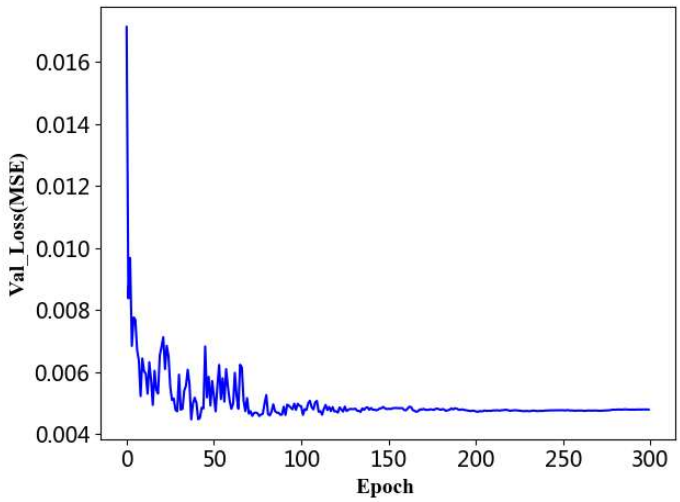


Figure 17. The loss function declining curve with epoch of the validation dataset.

(5) Performance using different learning rates in optimizer RMSprop

In addition to the learning rate (lr = 0.0008) used in this paper, we try employing 0.0004, 0.0006, 0.0007, 0.0009, 0.001, and 0.002. The positioning performance influence using different learning rates is shown in Figure 18 and the MLP-based regression model obtains the best positioning accuracy when the learning rate is set to 0.0008. According to Figure 18a, the localization error of the MLP-based regression model for 80% of the testing dataset falls between 8 m and 10 m when using RMSprop under different learning rates, in which the model (lr = 0.0008) obtains about 8 m. According to Figure 18b, the MLP-based regression model obtains the average positioning error of 5.64 m when the learning rate is set to 0.0008, which is 12.7% less than the model (lr = 0.002) that obtains the highest positioning error and 1.2% less than the model (lr = 0.0007) that obtains the highest positioning accuracy, except the model (lr = 0.0008).

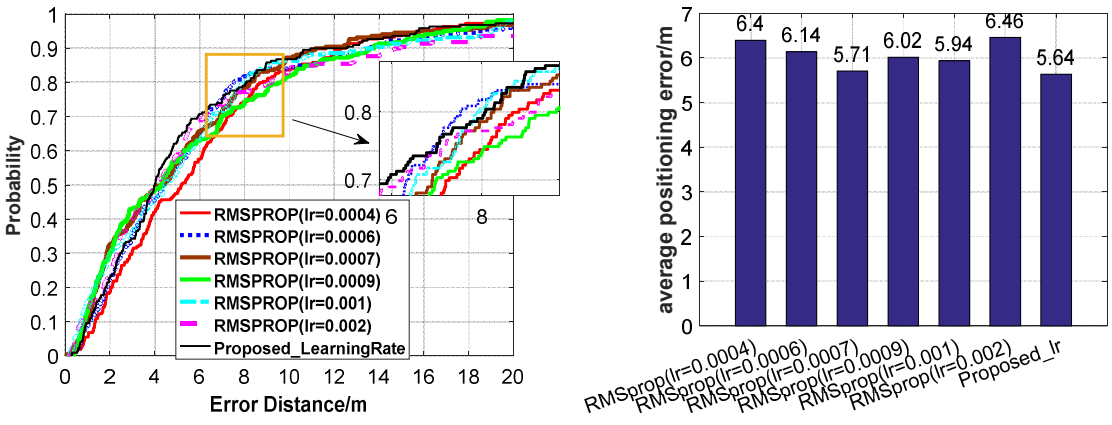


Figure 18. The positioning accuracy using different learning rates in optimizer (RMSprop). (**a**) CDF positioning errors. (**b**) Average positioning errors.

In the following experiments, we will use above-mentioned optimal parameters to further evaluate our proposed algorithm.

3.3.2. The Performance of Our Proposed Algorithm under Different Sample Densities

In this section, we evaluate the positioning performance of our proposed algorithm under different sample densities. We collect four datasets with different sampling densities. The distances between neighboring sampling points in the four datasets are 3 m, 5 m, 7 m, and 10 m, respectively. Each dataset contains training data, validation data and testing data. The positioning accuracy of our proposed algorithm on the four datasets is shown in Figure 19 and Table 2.

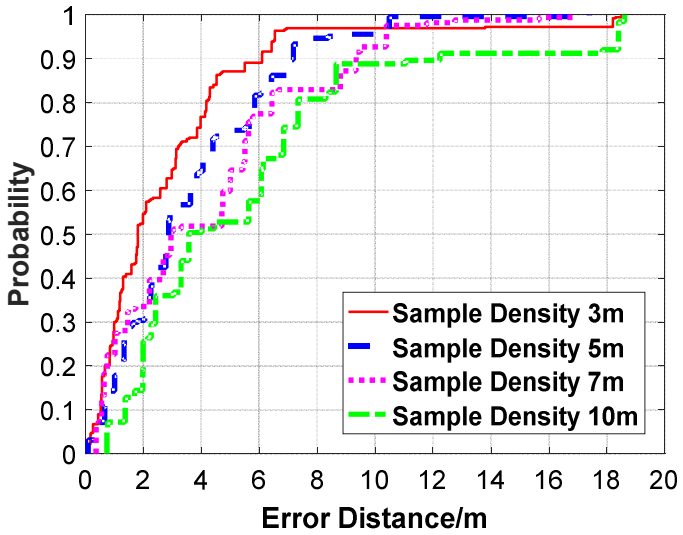


Figure 19. The CDF positioning error of our proposed algorithm under different sample densities.

Table 2. Average positioning error of our proposed algorithm under different sample densities.

Dataset	Neighboring sample_spacing (m)	Mean_error (m)
Sample Density 3 m	3	2.84
Sample Density 5 m	5	3.56
Sample Density 7 m	7	4.18
Sample Density 10 m	10	5.63

From Figure 19 and Table 2, we find that the positioning accuracy of our proposed algorithm decreases with the sampling density decrease. According to Table 2 the average positioning error is 2.84 m when the distance between sampling points is 3 m. The average positioning error increases to 5.63 m when the distance between sampling points is 10 m. In general, with the increase of neighboring sample distance, the localization errors of our proposed algorithm increase gradually.

3.3.3. The Positioning Performance on Three Datasets

The positioning performance of our proposed algorithm on Dataset1, Dataset2, and Dataset3 is shown in Figure 20 and Table 3.

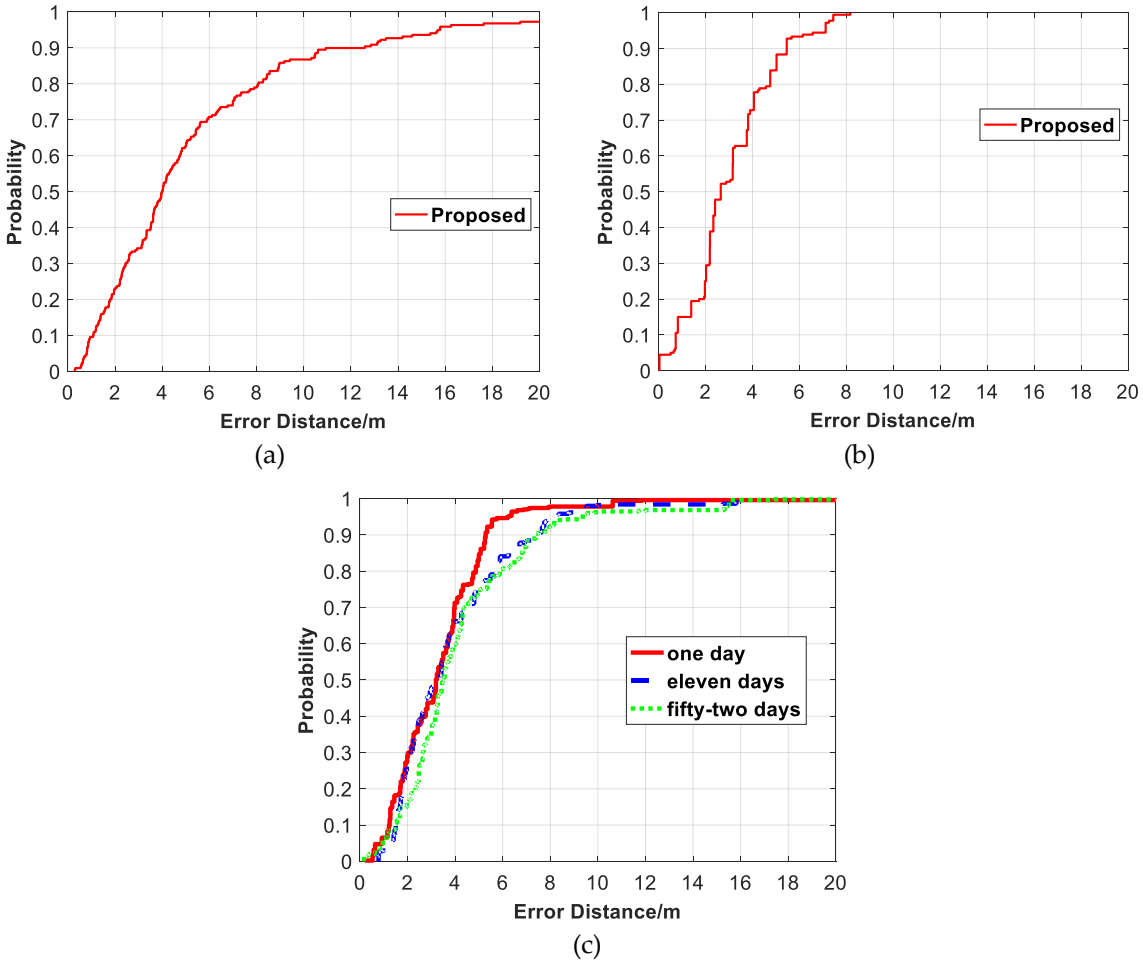


Figure 20. The CDF positioning error of our proposed algorithm on three datasets. (**a**) Dataset1. (**b**) Dataset2. (**c**) Dataset3.

Table 3. Average positioning error of the algorithm proposed in this paper on three datasets.

Dataset	Collection Interval (day)	Mean_error (m)
Dataset1	10	5.64
Dataset2	0	3.05
Dataset3	1	3.39
	11	3.85
	52	4.24

According to Figure 20, the proposed algorithm achieves about 8 m localization error for 80% of the testing dataset on Dataset1. The proposed algorithm achieves about 2.5 m and 5.2 m localization error for 50% and 90% of the testing dataset on Dataset2, respectively. On Dataset3, the proposed algorithm produces about 4.7 m localization error, 5.8 m localization error, and 6 m localization error for 80% of sub-test1, sub-test2, and sub-test3, respectively. From Table 3, we can find that using our proposed algorithm can obtain 5.64 m and 3.05 m of average positioning errors on Dataset1 and Dataset2, respectively. On Dataset3, the average positioning error of our presented method is 4.24 m when there is a 52-day collection interval between training dataset and testing dataset.

3.4. Performance Comparison with the Tree-Fusion-Based Regression Model

We compare the positioning accuracy of our proposed algorithm with the tree-fusion-based regression model described in Section 2.5 with the above-mentioned datasets. The fusion weight parameters of w1, w2, and w3 corresponding to the XGBoost, LightGBM, and Stacking model are set to 0.15, 0.15, and 0.7, respectively, based on our optimal experiments.

- Dataset1

We sort Dataset1 according to the data collection timestamp, and divide it into training dataset, validation dataset, and testing dataset. The collection time of the training dataset is 30 May 2013, and the collection time of the testing dataset is 10 June 2013. The positioning performance on Dataset1 is shown in Figure 21 and Table 4. Our proposed algorithm can obtain 5.64 m of average positioning error, as shown in Table 4, which is 10.8% less than the tree-fusion-based regression model. Our proposed algorithm and the tree-fusion-based regression model achieve 8 m localization error and 9 m localization error (shown in Figure 21) for 80% of the testing dataset, respectively, i.e., the localization accuracy using the tree-fusion-based regression model is 1 m worse than that using our proposed algorithm for 80% of the testing dataset.

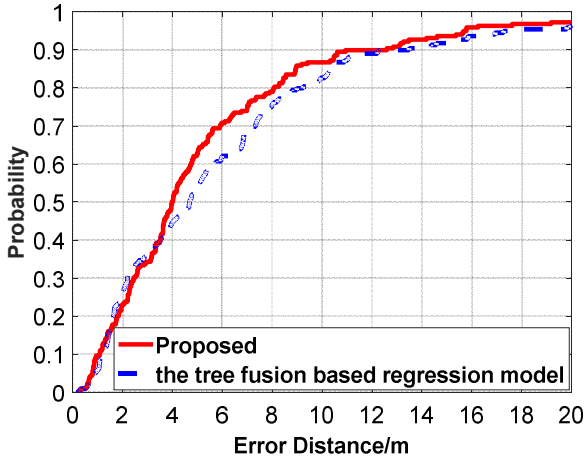


Figure 21. Comparison of CDF positioning errors between the proposed algorithm and the tree-fusion-based regression model.

Table 4. Average positioning errors of the proposed algorithm and the tree-fusion-based regression model.

Model	Mean_error (m)
tree-fusion-based regression model	6.32
Proposed model	5.64

- Dataset2

The positioning performance on Dataset2 using the proposed algorithm and the tree-fusion-based regression model is shown in Figure 22 and Table 5. Both the proposed algorithm and the tree-fusion-based regression model achieve about 4.8 m localization error for 80% of the testing dataset, as shown in Figure 22. Furthermore, the two algorithms produce about 3 m of average positioning error, as shown in Table 5. Considering that there is no time interval between training dataset and testing dataset in Dataset2, the fingerprint features corresponding to a specific location between the training dataset and the testing dataset are very similar, both the proposed algorithm and the tree-fusion-based regression model can accurately estimate target locations.

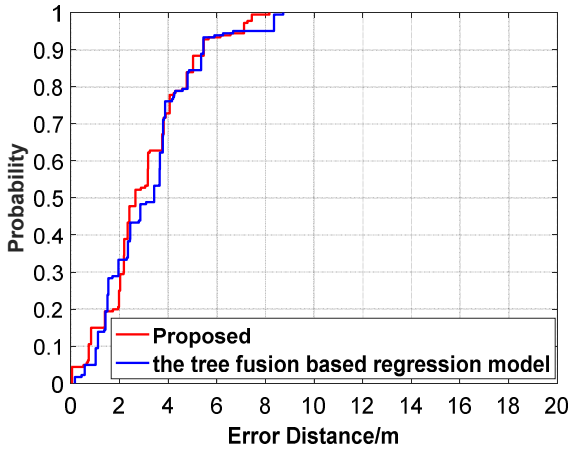


Figure 22. The positioning errors (CDF) between the proposed algorithm and the tree-fusion-based regression model.

Table 5. The average positioning error comparison between the proposed algorithm and the tree-fusion-based regression model.

Model	Mean_error (m)
tree-fusion-based regression model	3.22
Proposed model	3.05

- Dataset3

In Dataset3, positioning algorithms are performed on the testing dataset, which contains three sub-datasets (sub-test1, sub-test2, and sub-test3). The positioning performance is shown in Figure 23 and Table 6. As shown in Figure 23, the proposed algorithm and the tree-fusion-based regression model achieve 6 m and 7.2 m localization error for 80% of the sub-test3 (there is a 52-day long interval), respectively. As shown in Table 6, our proposed algorithm outperforms the tree-fusion-based regression model on all three sub-datasets. For instance, our proposed algorithm obtains 4.24 m of average positioning error on sub-test3, which is 14.5% less than the tree-fusion-based regression model. These experimental results confirm that our proposed algorithm can obtain better positioning performance than the tree-fusion-based regression model when there is a long time interval.

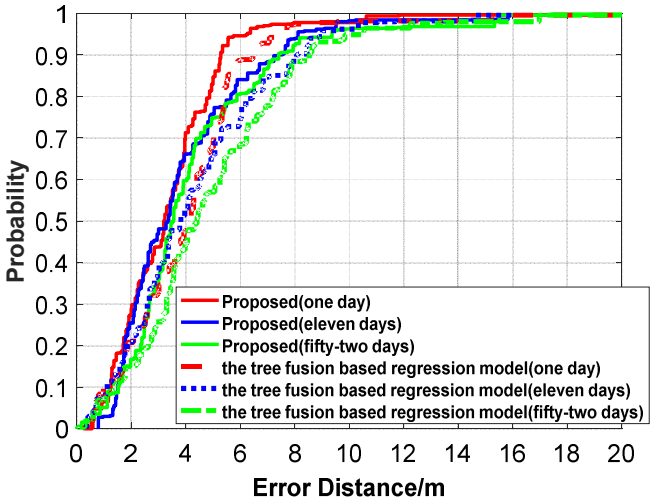


Figure 23. Comparison of CDF positioning errors between the proposed algorithm and the tree-fusion-based regression model.

Table 6. Comparison of average positioning errors between the proposed algorithm and the tree-fusion-based regression model.

Model	Collection Interval (day)	Mean_error (m)
Proposed model	1	3.39
tree-fusion-based regression model		4.19
Proposed model	11	3.85
tree-fusion-based regression model		4.54
Proposed model	52	4.24
tree-fusion-based regression model		4.97

The experimental results conducted on these three datasets demonstrate that when there is no time interval between training dataset and testing dataset, both the tree-fusion-based model and our proposed algorithm can achieve similar positioning performance. However, when there is a large time interval between training dataset and testing dataset, our proposed algorithm can obtain better positioning performance than the tree-fusion-based regression model. Therefore, the proposed algorithm is more robust when there is a large time interval between training dataset and testing dataset, which confirms that the MLP-based regression model can find good mapping between the Wi-Fi fingerprints and locations based on the strong representation of MLP.

3.5. Performance Comparison with Related Methods

We also compare the localization accuracy of our proposed algorithm with other state-of-the-art methods (Khatab [30], Xu [31]). Khatab introduced the AE to extract Wi-Fi features, and then used the ELM (Extreme Learning Machine) for indoor positioning. Xu also adopted the AE method for feature extraction, and then used the MLP for indoor positioning.

3.5.1. Performance Comparison with Khatab's Method

Considering that the dataset used in this paper is different from the dataset used in Khatab's paper, we do not use the same parameter values as those used in Khatab's paper. This paper reconstructs the same network structure of Khatab's paper. Then, we train this reconstructed model using our dataset. We conduct comparative experiments on Dataset1, Dataset2, and Dataset3. The experimental results are shown in Figure 24 and Table 7.

Table 7. The average positioning errors between our proposed algorithm and Khatab's method.

Dataset	Model	Collection Interval (day)	Mean_error (m)
Dataset1	Khatab [30]	10	8.14
	proposed	10	5.64
Dataset2	Khatab [30]	0	3.14
	proposed	0	3.05
Dataset3	Khatab [30]	1	4.08
		11	5.01
		52	5.60
	proposed	1	3.39
		11	3.85
		52	4.24

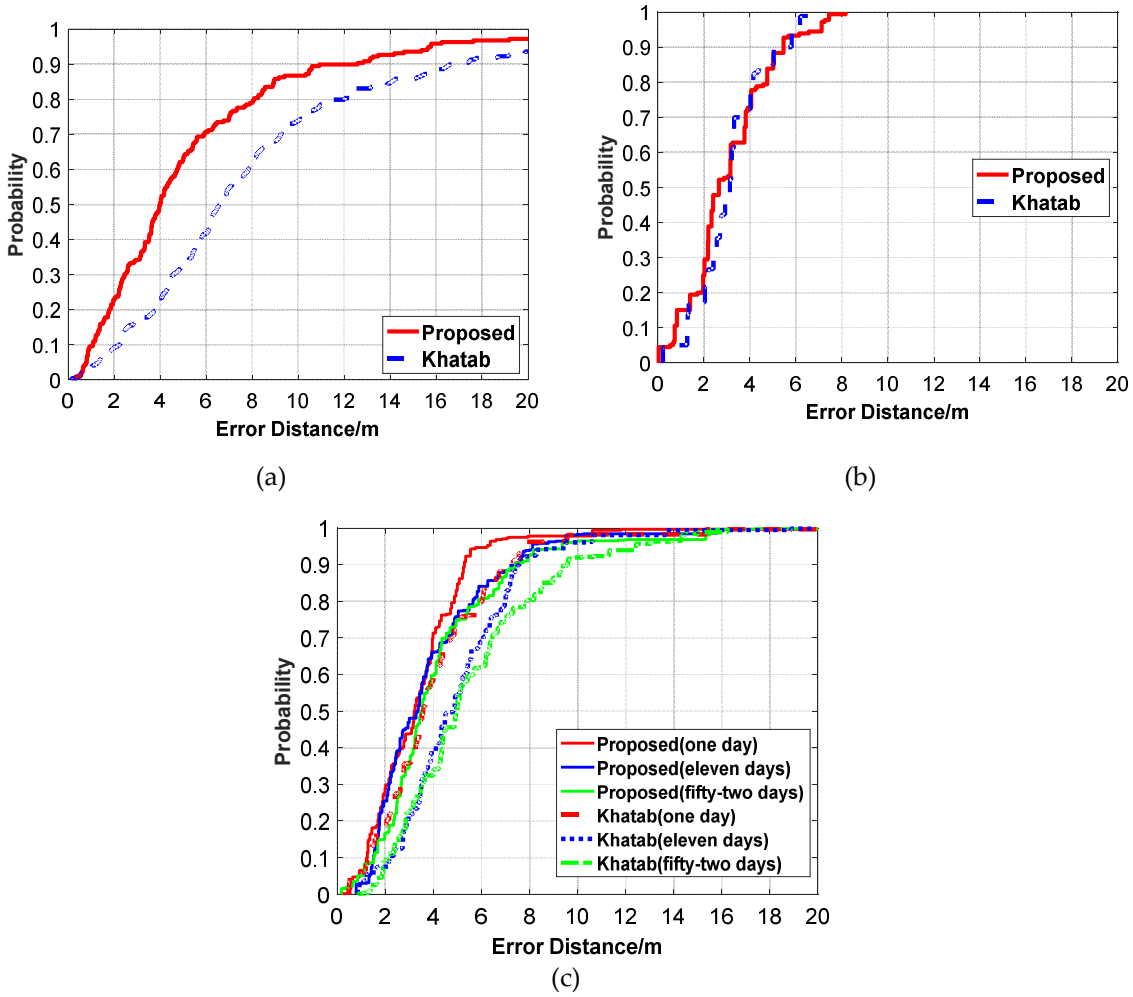


Figure 24. The positioning error comparison between our proposed algorithm and Khatab's method on three different datasets. (**a**) Dataset1. (**b**) Dataset2. (**c**) Dataset3.

From Table 7, we can find that the average positioning errors of our proposed algorithm and Khatab's method on the Dataset2 are similar (3.05 m and 3.14 m, respectively). This similar positioning performance is also illustrated in Figure 24b (both the proposed algorithm and Khatab achieve about 6 m localization error for 90% of the testing dataset on Dataset2). This demonstrates that when there is no time interval between training dataset and testing dataset, both algorithms can achieve good and similar positioning performance. However, from Figure 24a,c and Table 7, we can see that the positioning performance using our proposed algorithm is better on Dataset1 and Dataset3. Our proposed algorithm produces 5.64 m of average positioning error on Dataset1, which is 30.7% less than Khatab. Our proposed algorithm produces the average positioning error of 4.24 m on sub-test3 of Dataset3, which is 24.3% less than Khatab. These experimental results confirm that our proposed algorithm is more robust when there is a large time interval between the training dataset and testing dataset. The reason why our proposed algorithm can obtain better positioning accuracy when there is a large time interval between the training dataset and testing dataset is as follows—the output features extracted by AE may be a simple copy of the input layer, which does not extract more essential features from the Wi-Fi signal. Furthermore, the weights of hidden layers in ELM are no longer updated after determining their weights by solving the equation set. Differing from ELM, the MLP updates the weights of hidden layers until the loss function becomes steady. Therefore, the MLP can obtain a better mapping function.

3.5.2. Performance Comparison with Xu's Method

Similar to the 3.5.1, we also employ our three datasets to evaluate the performance of Xu's method. We also reconstruct the network structure of Xu's paper. The comparative experiments on Dataset1, Dataset2, and Dataset3 are conducted and the experimental results are shown in Figure 25 and Table 8.

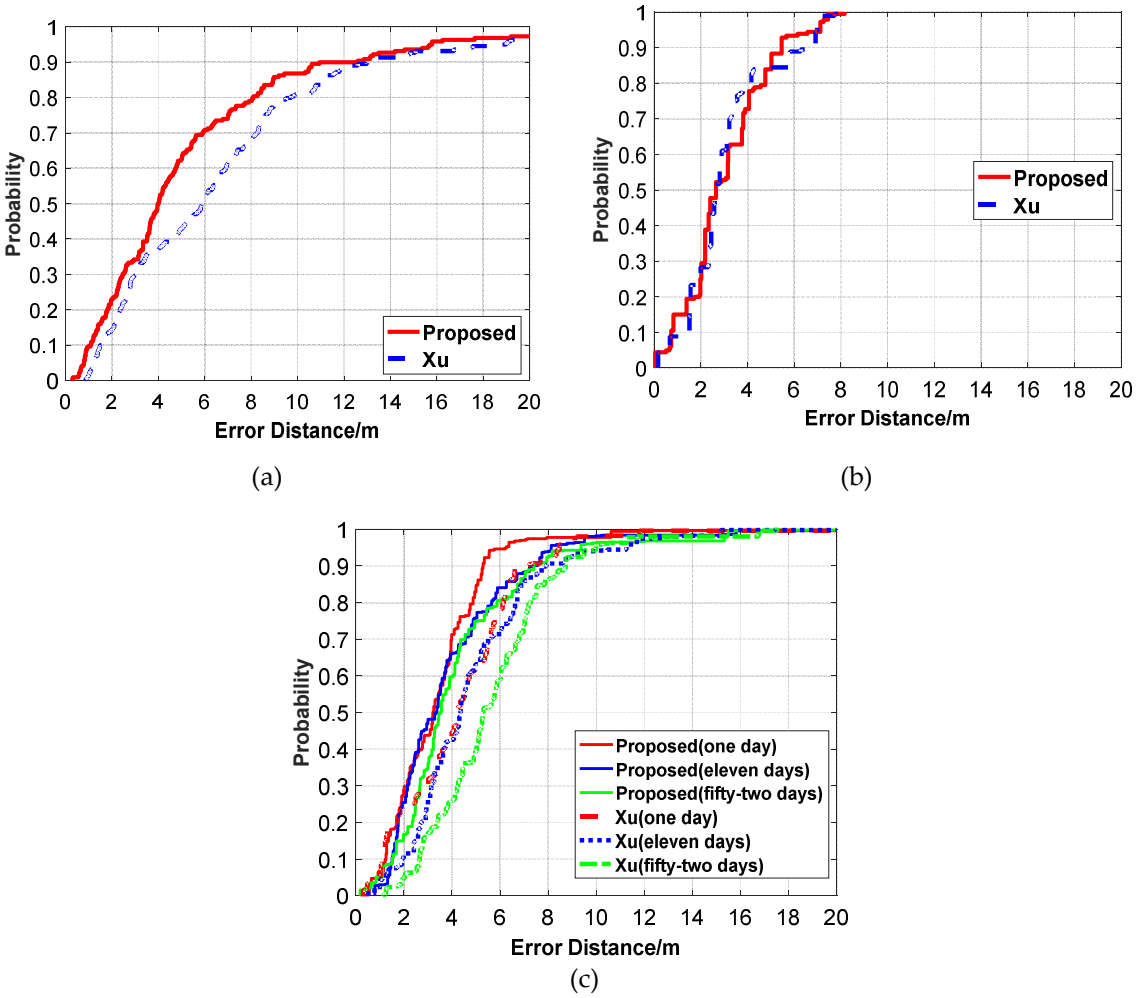


Figure 25. Comparison of CDF positioning errors between the proposed algorithm and Xu on different datasets. (**a**) Dataset1. (**b**) Dataset2. (**c**) Dataset3.

Table 8. Comparison of the average positioning errors between the proposed algorithm and Xu.

Dataset	Model	Collection Interval (Day)	Mean_error (m)
Dataset1	Xu [31]	10	7.00
	proposed	10	5.64
Dataset2	Xu [31]	0	3.07
	proposed	0	3.05
Dataset3	Xu [31]	1	4.30
		11	4.86
		52	5.67
	proposed	1	3.39
		11	3.85
		52	4.24

From Table 8 and Figure 25b, we can find that both our proposed method and Xu's method obtain similar localization performance on the Dataset2, without time intervals between the training dataset and testing dataset. Both the proposed algorithm and Xu achieve about 6 m localization error for 90% of the testing dataset on Dataset2, and produce about 3 m of average positioning error. However, when there is a large time interval between training dataset and testing dataset, the positioning performance using our proposed algorithm is obviously better on Dataset1 and Dataset3, as Figure 25a,c and Table 8 show. Our proposed algorithm produces 5.64 m of average positioning error on Dataset1, which is 19.4% less than Xu. Our proposed algorithm produces the average positioning error of 4.24 m on sub-test3 of Dataset3, which is 25.2% less than Xu. The reason for this difference between our proposed algorithm and Xu's method is that AE is also used for feature extraction in Xu's method. Similar to previous experiments, the features extracted by AE may be a simple copy of the input layer, which does not extract more essential features.

From the results of all experiments in Sections 3.5.1 and 3.5.2, we obtain the following main conclusions:

- When there is no time interval between training dataset and testing dataset, both Khatab's method and Xu's methods can achieve good positioning results.
- When there is a time interval between the training dataset and testing dataset, our proposed algorithm can obtain better positioning performance, which confirms that our proposed algorithm is more robust.

3.6. Calculation Complexity

To evaluate the calculation complexity of our proposed algorithm, we compare the total time (including total training time and prediction time for one sample on Dataset1) of our proposed algorithm with Khatab's method and Xu' method. All algorithms are run on a PC with Intel i5-6500 CPU and 8GB RAM. Table 9 lists the total time of different algorithms.

Table 9. Calculation time of different algorithms.

Model	Total Training Time (s)	Prediction Time for One Sample (ms)
proposed	20.84	181
Khatab	5.21	3
Xu	16.54	166

Table 9 indicates that it takes the longest time for our proposed algorithm. However, the total training time is only about 21 s and run on the offline stage, which does not influence online real-time positioning. Khatab's method adopts ELM to obtain positioning results during the positioning phase. The weights in the ELM are obtained by solving the equation set. However, the MLP method adopts the back-propagation algorithm to repeatedly adjust the weights, so the ELM algorithm obtains the shortest runtime in the offline stage. Khatab's method and the proposed algorithm in this paper adopt AE and DAE in the feature extraction phase, respectively, with the DAE having longer runtime than the AE. In the positioning phase, ELM in Khatab's method adopts a simpler network structure, and the MLP network structure proposed in this paper is more complicated. Therefore, Khatab's method obtains the shortest prediction time. Finally, although our algorithm has the longest prediction time for one sample, the latency at the millisecond level is negligible.

4. Conclusions

In this paper, we propose an indoor positioning algorithm combining SDAE and MLP, in which the SDAE performs feature extraction and the MLP performs regression positioning. To solve the Wi-Fi signal dynamic fluctuation with time, we adopt the SDAE-based robust feature extraction method, and then build a MLP-based regression model for indoor positioning. To evaluate our proposed algorithm,

we conduct experiments in three datasets which represent different scenarios. The experimental results indicate that the SDAE-based feature extraction method extracts robust and time-independent features, which represent the raw Wi-Fi data well, and the MLP-based regression model finds a good mapping function. Extensive experimental results demonstrate that the proposed algorithm and other algorithms can achieve similar and good positioning performance when there is a short time interval between the training dataset and testing dataset. However, the proposed algorithm obtains 4.24 m of average positioning error when there is a 52-day interval between training dataset and testing dataset, which is 24.3% less than Khatab's method, 25.2% less than Xu's method, and 14.7% less than the tree-fusion-based regression model, respectively. This confirms that our proposed algorithm is more robust than other algorithms when there is a large time interval between the training dataset and testing dataset.

In future work, we will continue to improve positioning accuracy, apply our proposed method in practical environments, and we will evaluate our proposed algorithm using more datasets. Also, we will consider extracting more essential features to eliminate the effect of time on the Wi-Fi signal. Furthermore, we will consider better mapping the extracted features to the target position by building other network structures.

Author Contributions: Conceptualization, R.W., H.L., and Z.F.; methodology, R.W.; software, R.W.; validation, H.L. and Z.L.; formal analysis, Q.W.; investigation, Z.L. and H.L.; resources, H.L.; data curation, Z.L., R.W., and Q.W.; writing—original draft preparation, R.W.; writing—review and editing, R.W., H.L., and Q.W.; visualization, R.W. and Z.L.; supervision, H.L. and W.S.; project administration, H.L.; funding acquisition, H.L. and F.Z.

Funding: This work was supported in part by the National Key Research and Development Program (2018YFB0505200), the National Natural Science Foundation of China (61872046,61374214), the "Blue Fire Plan" (Huizhou) Industry-University Joint Innovation Project of Ministry of Education (CXZJHZ201729), and the Open Project of the Beijing Key Laboratory of Mobile Computing and Pervasive Device.

Acknowledgments: We would like to thank the editors and the three anonymous reviewers for their valuable comments, which greatly improved the quality of this manuscript. Many thanks to the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology Chinese Academy of Sciences for the support in our research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wei, J.; Zhao, F.; Luo, H. SP-Loc: A crowdsourcing fingerprint based shop-level indoor localization algorithm integrating shop popularity without the indoor map. *Int. J. Distrib. Sens. Netw.* **2018**, *14*. [CrossRef]
2. Guo, X.; Shao, W.; Fang, Z.; Qu, W.; Li, D.; Luo, H. WiMag: Multimode Fusion Localization System based on Magnetic/WiFi/PDR. In Proceedings of the 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Alcala de Henares, Spain, 4–7 October 2016; pp. 1–8.
3. Luo, H.; Zhao, F.; Jiang, M.; Ma, H.; Zhang, Y. Constructing an indoor floor plan using crowdsourcing based on magnetic fingerprinting. *Sensors* **2017**, *17*. [CrossRef]
4. Shao, W.; Zhao, F.; Wang, C.; Luo, H.; Muhammad Zahid, T.; Wang, Q.; Li, D. Location Fingerprint Extraction for Magnetic Field Magnitude Based Indoor Positioning. *J. Sens.* **2016**, *2016*, 1–16. [CrossRef]
5. Xu, L.; Xiong, Z.; Liu, J.; Wang, Z.; Ding, Y. A Novel Pedestrian Dead Reckoning Algorithm for Multi-Mode Recognition Based on Smartphones. *Remote Sens.* **2019**, *11*, 294. [CrossRef]
6. Yang, F.; Xiong, J.; Liu, J.; Wang, C.; Li, Z.; Tong, P.; Chen, R. A Pairwise SSD Fingerprinting Method of Smartphone Indoor Localization for Enhanced Usability. *Remote Sens.* **2019**, *11*, 566. [CrossRef]
7. Wang, Q.; Luo, H.; Men, A.; Zhao, F.; Gao, X.; Wei, J.; Zhang, Y.; Huang, Y. Light positioning: A high-accuracy visible light indoor positioning system based on attitude identification and propagation model. *Int. J. Distrib. Sens. Netw.* **2018**, *14*. [CrossRef]
8. Wang, Q.; Luo, H.; Men, A.; Zhao, F.; Huang, Y. An infrastructure-free indoor localization algorithm for smartphones. *Sensors* **2018**, *18*. [CrossRef] [PubMed]
9. Sen, S.; Lee, J.; Kim, K.H.; Congdon, P. Avoiding multipath to revive inbuilding WiFi localization. In Proceedings of the 11th annual international conference on Mobile systems, applications, and services, Taipei, Taiwan, China, 25–28 June 2013; pp. 249–262.

10. Ciurana, M.; Barcelo-Arroyo, F.; Izquierdo, F. A ranging method with IEEE 802.11 data frames for indoor localization. In Proceedings of the 2007 IEEE Wireless Communications and Networking Conference(WCNC), Kowloon, China, 11–15 March 2007; pp. 2094–2098.
11. Chintalapudi, K.; Padmanabha, I.A.; Padmanabhan, V.N. Indoor localization without the pain. In Proceedings of the sixteenth annual international conference on Mobile computing and networking, New York, NY, USA, 20–24 September 2010; pp. 173–184.
12. Zhang, J.; Han, G.; Sun, N.; Shu, L. Path-loss-based fingerprint localization approach for location-based services in indoor environments. *IEEE Access* **2017**, *5*, 13756–13769. [CrossRef]
13. He, S.; Chan, S.H.G. Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 466–490. [CrossRef]
14. Ding, H.; Zheng, Z.; Zhang, Y. AP weighted multiple matching nearest neighbors approach for fingerprint-based indoor localization. In Proceedings of the 2016 Fourth International Conference on Ubiquitous Positioning, Indoor Navigation and Location Based Services (UPINLBS), Shanghai, China, 2–4 November 2016; pp. 218–222.
15. Zayets, A.; Steinbach, E. Robust WiFi-based indoor localization using multipath component analysis. In Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Sapporo, Japan, 18–21 September 2017; pp. 1–8.
16. Xiao, C.; Yang, D.; Chen, Z.; Tan, G. 3-D BLE Indoor Localization Based on Denoising Autoencoder. *IEEE Access* **2017**, *5*, 12751–12760. [CrossRef]
17. Barsocchi, P.; Crivello, A.; La Rosa, D.; Palumbo, F. A multisource and multivariate dataset for indoor localization methods based on WLAN and geo-magnetic field fingerprinting. In Proceedings of the 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Alcala de Henares, Spain, 4–7 October 2016; pp. 1–8.
18. Ficco, M.; Esposito, C.; Napolitano, A. Calibrating Indoor Positioning Systemswith Low Efforts[M]. *IEEE Trans. Mob. Comput.* **2014**, *13*, 737–751. [CrossRef]
19. Chai, X.; Yang, Q. Reducing the calibration effort for probabilistic indoor location estimation. *IEEE Trans. Mob. Comput.* **2007**, *6*, 649–662. [CrossRef]
20. Umair, M.Y.; Ramana, K.V.; Dongkai, Y. An enhanced K-Nearest Neighbor algorithm for indoor positioning systems in a WLAN. In Proceedings of the 2014 IEEE Computers, Communications and IT Applications Conference, Beijing, China, 20–22 October 2014; pp. 19–23.
21. Ge, X.; Qu, Z. Optimization WIFI indoor positioning KNN algorithm location-based fingerprint. In Proceedings of the 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 26–28 August 2016; pp. 135–137.
22. Lu, X.; Qiu, Y.; Yuan, W.; Yang, F. An improved dynamic prediction fingerprint localization algorithm based on KNN. In Proceedings of the 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), Harbin, China, 21–23 July 2016; pp. 289–292.
23. Dakkak, M.; Daachi, B.; Nakib, A.; Siarry, P. Multi-layer perceptron neural network and nearest neighbor approaches for indoor localization. In Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, 5–8 October 2014; pp. 1366–1373.
24. Altay, O.; Ulas, M. Location determination by processing signal strength of Wi-Fi routers in the indoor environment with linear discriminant classifier. In Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018; pp. 1–4.
25. Zhang, S.; Guo, J.; Luo, N.; Wang, L.; Wang, W.; Wen, K. Improving Wi-Fi Fingerprint Positioning with a Pose Recognition-Assisted SVM Algorithm. *Remote Sens.* **2019**, *11*, 652. [CrossRef]
26. Bekkali, A.; Sanson, H.; Matsumoto, M. RFID indoor positioning based on probabilistic RFID map and Kalman Filtering. In Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007), White Plains, NY, USA, 8–10 October 2007; p. 21.
27. Youssef, M.; Agrawala, A. Handling samples correlation in the horus system. In Proceedings of the IEEE INFOCOM 2004, Hong Kong, China, 7–11 March 2004; pp. 1023–1031.
28. Rizk, H.; Torki, M.; Youssef, M. CellinDeep: Robust and Accurate Cellular-based Indoor Localization via Deep Learning. *IEEE Sens. J.* **2018**, *19*, 2305–2312. [CrossRef]

29. Zhang, W.; Sengupta, R.; Fodero, J.; Li, X. DeepPositioning: Intelligent fusion of pervasive magnetic field and Wifi fingerprinting for smartphone indoor localization via Deep Learning. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 7–13.
30. Khatab, Z.E.; Hajihoseini, A.; Ghorashi, S.A. A Fingerprint Method for Indoor Localization Using Autoencoder Based Deep Extreme Learning Machine. *IEEE Sens. Lett.* **2017**, *2*, 1–4. [CrossRef]
31. Xu, C.; Jia, Z.; Chen, P.; Wang, B. CSI-based autoencoder classification for Wi-Fi indoor localization. In Proceedings of the 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, 28–30 May 2016; pp. 6523–6528.
32. Kim, K.S.; Wang, R.; Zhong, Z.; Tan, Z.; Song, H.; Cha, J.; Lee, S. Large-scale location-aware services in access: Hierarchical building/floor classification and location estimation using Wi-Fi fingerprinting based on deep neural networks. In Proceedings of the 2017 International Workshop on Fiber Optics in Access Network (FOAN), Munich, Germany, 6–8 November 2017; pp. 1–5.
33. Kim, K.S. Hybrid building/floor classification and location coordinates regression using a single-input and multi-output deep neural network for large-scale indoor localization based on Wi-Fi fingerprinting. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), Takayama, Japan, 27–30 November 2018; pp. 196–201.
34. Nowicki, M.; Wietrzykowski, J. Low-effort place recognition with WiFi fingerprints using deep learning. In *International Conference Automation*; Springer: Berlin, Germany, 2017; pp. 575–584.
35. Wang, X.; Wang, X.; Mao, S. Deep Convolutional Neural Networks for Indoor Localization with CSI Images. *IEEE Trans. Netw. Sci. Eng.* **2018**. [CrossRef]
36. Wang, X.; Wang, X.; Mao, S. CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
37. Shao, W.; Luo, H.; Zhao, F.; Ma, Y.; Zhao, Z.; Crivello, A. Indoor Positioning Based on Fingerprint-Image and Deep Learning. *IEEE Access* **2018**, *6*, 74699–74712. [CrossRef]
38. Wang, X.; Yu, Z.; Mao, S. DeepML: Deep LSTM for Indoor Localization with Smartphone Magnetic and Light Sensors. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–8.
39. Hsieh, H.-Y.; Prakosa, S.W.; Leu, J.-S. Towards the Implementation of Recurrent Neural Network Schemes for WiFi Fingerprint-Based Indoor Positioning. In Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, IL, USA, 27–30 August 2018; pp. 74699–74712.
40. Lu, X.; Long, Y.; Zou, H.; Yu, C.; Xie, L. Robust extreme learning machine for regression problems with its application to wifi based indoor positioning system. In Proceedings of the 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), Reims, France, 21–24 September 2014; pp. 1–6.
41. Zhou, R.; Chen, J.; Lu, X.; Wu, J. CSI fingerprinting with SVM regression to achieve device-free passive localization. In Proceedings of the 2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Macau, China, 12–15 June 2017; pp. 1–9.
42. Sun, W.; Xue, M.; Yu, H.; Tang, H.; Lin, A. Augmentation of Fingerprints for Indoor WiFi Localization Based on Gaussian Process Regression. *IEEE Trans. Veh. Technol.* **2018**, *67*, 10896–10905. [CrossRef]
43. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.-A. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408. [CrossRef]
44. Simon, H. *Neural Networks: A Comprehensive Foundation*; Macmillan College Publishing Company: New York, NY, USA, 1994; Volume 13, pp. 409–412.
45. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **1989**, *2*, 303–314. [CrossRef]
46. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning(ICML), Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.
47. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.

48. Ioffe, S.; Christian, S. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1–13.
49. Diederik, P.; Kingma, J.B. Adam: A Method for Stochastic Optimization. In Proceedings of the ICLR 2015, San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
50. Tieleman, T.; Hinton, G.E.; Srivastava, N.; Swersky, K. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. COURSERA: Neural Networks for Machine Learning, 4, 26–30. Available online: http://www.cs.toronto.edu/~{}tijmen/csc321/slides/lecture_slides_lec6.pdf (accessed on 20 December 2018).
51. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 1–9.
52. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16), SanFrancisco, CA, USA, 13–17 August 2016; ACM Press: New York, NY, USA, 2016; pp. 785–794.
53. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
54. Zhou, Z.-H. *Ensemble Methods: Foundations and Algorithms*; Chapman & Hall/CRC: New York, NY, USA, 2012; Volume 8, pp. 77–79.