Wireless indoor localization using convolutional neural network

Tianqi Qu¹, Meng Li and Dong Liang
Beijing University of Posts and Telecommunications, Beijing 100876, China
¹E-mail: keyonqu@163.com

Abstract. This paper provides a wireless localization method using convolutional neural networks (CNN) and based on WIFI fingerprinting data. We pre-processed WIFI fingerprinting data from Jaume I University public database and used it to train our CNN model. A WIFI fingerprint is composed of a series of received signals’ strength (RSS) released by many access points (APs). We changed the RSS’s value at the edge of the signal disappearance. This operation can help the CNN model extract the features of fingerprints and have a better performance. After this operation, a CNN model can have about 21.2% lower loss and 23.5% higher accuracy than usual. Changing edge RSS is shown to be a good way to improve the performance of indoor localization.

1. Introduction
Nowadays, because of the big improvements in Internet of Things industry and navigation systems, people are facing a great need of more precise indoor localization methods, which can help them solve many problems of location-based services like marketing guide and car parking. It is shown that GPS navigation system is unreliable in indoor environment because of loss of signal and obstruction of buildings. As a result, there is a great demand of efficient and accurate indoor positioning methods. At present, many indoor positioning technologies have been proposed, including Bluetooth positioning, WiFi positioning, inertial sensor positioning, RFID positioning, laser positioning and so on, and they are used in different situations. During these positioning methods, WIFI positioning and Bluetooth positioning are received signal strength indication (RSSI) based and are most widely used. WiFi signals exist in most indoor environments, so it is a very cost saving method to use WiFi to locate without additional hardware deployment. People collect WIFI received signal strength (RSS) at different locations and store them as a ‘fingerprint’ database. Then they can determine their position by comparing the signals receiving with the signals stored. In Paper [1], Marina Md Din and her coworkers provides a comprehensive review of several indoor localization calculation methods based on RSS. All of these traditional methods use geometric formulas to calculate the positions.

Machine learning methods are used to improve the performance of WIFI positioning. For example, WIFI positioning problem can be regarded as a classification or regression problem, so k-Nearest Neighbor (KNN) model and its variant k-Weighted Nearest Neighbor (WKNN) are always used to classify the signals and predict the coordinates of the new nodes. They calculate the similarity between the real-time RSS with the RSS stored. Then select one or several sampling points with the best similarity, and use the average or weighted average of each sampling point to obtain the user's location estimation.[2] Besides, in [3] Mauro Brunato and Roberto Battiti proved the superiority of SVM in regression and classification by comparing it with KNN, Bayesian a posteriori likelihood (BAY) and
Multi-layer perception (MLP). SVM model shows many advantages in solving high dimensional nonlinear problems.

The base of training good models is a good data source. In this paper, we used a WIFI fingerprinting database provided by Jaume I University as the source of data, as the test points at the data collection site are well distributed and have high RSSI dimensions. Then we designed a wireless positioning model using convolutional neural networks (CNN). CNN is commonly used in image identification fields and also be used to support indoor positioning by processing photos. The input images would be processed to multi-channel pixels’ data to form the input layer. And then the convolutional layer will extract the features and build a connection between the feature data and labels. The principles of classical CNN models can be learned from the Paper [4] of Matthew D Zeiler and Rob Fergus. And in our case, when it is used to process RSS, the most vital point is how to process data to form the ‘images’ of a CNN model. Every RSS data in a reference point’s (RP) RSS vector is detected by a particular AP and the vector’s multiple RSS can fit into multi-channels. Therefore, a particular RP’s RSS vectors detected at different times can form an ‘image’. In Paper [5], Qinghe Zheng and his coworkers provided a novel data augmentation method which can improve the performance of deep CNN models. So, it enlightened us that we can improve the models by changing the original data. And in order to enhance the performance of feature extraction, we highlighted the edge at which the signal disappears by setting the edge undetectable signals to a weak value after we extracted the RSS vector data of each RP. The process of edge value is also the main innovative point of our model. After we trained the CNN model by inputting all RP’s RSS ‘images’, we used test data to calculate the final loss and accuracy. By comparing our model’s result with normal CNN model’s result, we found that this process of edge value does contribute to the enhancement of positioning accuracy.

Main contribution of this paper:
• A wireless indoor localization model using CNN based on WIFI fingerprinting
• Performance Analysis of this model
• Comparison between new model and normal CNN model

2. Data pre-processing
The quality of data pre-process will determine the quality of the whole model. And this part is also where we tried to improve in the CNN model. This part of the paper will introduce the background of WIFI fingerprinting localization technology and data, features of the data we used and the operations we did. Figure 1 below can show the basic principles of WIFI fingerprinting.

![Figure 1. Working diagram of WIFI fingerprinting localization.](image)

2.1. WIFI fingerprinting localization
According to [6], any feature that could be used to identify a location is a positional fingerprint, such as the multipath structure of a signal, RSS detected, round trip time or delay of a signal and so on. During these kinds of fingerprints, multi-path structure and RSS are the most commonly used. In our paper, we choose RSS data to be the fingerprints of locations. RSS is determined by the location relationship between the RP and AP, so if we make sure the location of AP and corresponding RSS,
we can deduce RP’s location. Therefore, the RSS values form multiple APs detected at a particular RP can form a RSS vector, which is the fingerprint we used in our model. And as most daily wireless devices process RSS data to control their information transportation, it is easier to access RSS data in daily life.

As Figure 2 shows below, there are two phases in the fingerprinting localization process: offline phase and online phase. During the offline phase, people set the locations of RP and collect the fingerprints of all RPs and build a database. And during the online phase, people compare RSS data receiving at test points (TP) with fingerprints data stored to deduce the location of TPs.

2.2. Data pre-process operations

The WIFI fingerprint database Jaume I University was collected in a rectangular area with 620 APs for 25 months. It provides the RSS value detected, coordinates of RPs, time and identifiers. However, we only use the RSS data and coordinates data in our case. Besides, this database was divided to two types: training data and test data. In the training dataset, there are 24 different locations of RPs. And in each file, there are 24 records of each RP, so there are 576 rows of data in each file. To process these data to suit our model, the first thing we did is to give labels to each coordinate. In other words, we use one-dimensional labels to instead two-dimensional coordinates. Coordinates data of RPs are between (4.13477, 16.69981) and (12.91385, 29.21654) as Figure 3, and labels are from 1 to 24.

Secondly, we extracted RSS data of each RP from those files based on the RSS vectors’ labels and store then separately. That’s because our model’s input is the data of one particular RP at a time. After that, we changed the values at the boundaries at which the signal disappears. In a RP’s RSS vector, the RSS values from the APs far away are displayed as ‘100’ dBm, which means no signal is received. And for other receivable signals, their RSS are negative integers, and the larger the overall value, the stronger the signal. The strongest signal in the database is ‘-33’ dBm and the weakest is ‘-97’ dBm. So we replaced the ‘100’ dBm signals at the edges to ‘-99’ dBm to highlight obvious changes in the 'images' as Figure 4. Figure 5 shows the whole work we did in the data pre-process.
Figure 4. Examples of changing the RSSs at the edge of signal disappearance.

Figure 5. Flow diagram of data pre-process.

After that, we normalized the data with and without changed boundary values and trained the model separately. The purpose of data normalization is to eliminate the effect of units and improve the effect of data analysis. After data normalization, all the values can be re-sized into zero to one. In our paper, we used Min-Max Normalization method to process raw data. In this way, we are ready to use the data to train CNN model.

\[ x^* = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \]  

(1)

3. CNN Architecture and training

Our CNN model structure referred to the model in Paper [7]. However, our model used edge value changed data to train and we want to show that it can perform better than other CNN model using original data. The first phase of this model is using the edge value changed training dataset to train CNN model. Secondly, we used original training dataset trained the same CNN model. Then we used test dataset to evaluate these two models and compared the result. To form the input layer of our CNN model, we reshaped the first 256 fingerprints of each RP from matrix 256×620 to matrix 16×16×620. And we also did the same operations to TPs to prepare for the test after model training.

Our CNN model was coded and compiled on Jupyter Notebook with Tensorflow environment. Figure 6 below can show the whole structure of this CNN model. It is consisted of three convolutional layers, three max-pooling layers and a fully connection layer. Every time we input a 16×16×620 matrix of one RP’s training data to the input layer. Then the first convolutional layer will create a 16×16×1000 matrix by processing the input layer with 1000 3×3 filters and a rectified linear unit (ReLU) activation function. Activation functions can add some nonlinear factors to the neural network, so that the neural network can better solve more complex problems. All the parameters are concluded in the Table 1 below.
Table 1. Structure and parameters of CNN model.

| Operation          | Input data size | Output data size | Description                   |
|--------------------|-----------------|------------------|-------------------------------|
| Convolutional Layer| 16×16×620       | 16×16×1000       | Filter: 3×3×1000  Stride: 1  |
|                    |                 |                  | Padding: SAME                | Activation: ReLU |
| Max-pooling Layer  | 16×16×1000      | 8×8×1000         | Pool_size: 2×2               | Dropout rate: 0.5 |
| Dropout            |                 |                  |                               |                  |
| Convolutional Layer| 8×8×1000        | 8×8×2000         | Filter: 3×3×2000  Stride: 1  |
|                    |                 |                  | Padding: SAME                | Activation: ReLU |
| Max-pooling Layer  | 8×8×2000        | 4×4×2000         | Pool_size: 2×2               | Dropout rate: 0.5 |
| Dropout            |                 |                  |                               |                  |
| Convolutional Layer| 4×4×2000        | 4×4×4000         | Filter: 3×3×4000  Stride: 1  |
|                    |                 |                  | Padding: SAME                | Activation: ReLU |
| Max-pooling Layer  | 4×4×4000        | 2×2×4000         | Pool_size: 2×2               | Dropout rate: 0.5 |
| Dropout            |                 |                  |                               |                  |
| Flatten            | 2×2×4000        | 16000            | Units: 256                   | Activation: ReLU |
|                    |                 |                  |                               | Dropout rate: 0.5 |
| Fully Connected Layer| 16000        | 256              | Units: 24                    | Activation: softmax |
| Dropout            |                 |                  |                               |                  |
| Fully Connected Layer| 256           | 24               | Optimizer: Adadelta           |                  |
| Compile            |                 |                  | Loss: Cross entropy           |                  |

Figure 6. Structure of CNN Model.

After that, there is a pooling layer to reduce the size of the ‘image’ to prevent overfitting. Max-pooling is one of the most commonly used pooling method. Our pooling size is 2×2, which means for every 2×2 data only the largest one can be remained. Then a dropout method will throw away a quarter of the network connections to prevent overfitting. Regularization methods are often used to improve the generalization ability and decrease the complexity of machine learning models. Qinghe Zheng and his coworkers provide in Paper [8] that regularization is an efficient way to maintain a lower overfitting and better performance. Paper [9] provides that use ReLU and dropout can improve the performance of LeNet CNN model, which is very similar to our CNN model. So in our model we use ReLU and dropout function too. Then we got an 8x8x1000 matrix. The second convolutional layer creates an 8×8×2000 matrix by convolved the 8x8x1000 matrix with 2000 3×3 filters. Also a ReLU activation function is used. Then a max-pooling layer would reduce the size to 4x4x2000. And then dropout a quarter of the network connections. The third convolutional layer creates a 4x4x4000 matrix by convolved the 4×4×2000 matrix with 4000 3×3 filters. And a max-pooling layer and drop out connections. Finally, a 2×2×4000 matrix will be flattened and going through a fully connection layer, then we get 24 output values. The output is 24 groups of possibilities. So if we want to calculate the loss, we need to change the original labels to one-hot codes. Then we used cross-entropy function to calculate the loss between the output and labels.

After written the structure of the CNN model, we complied it and use a keras function as below figure to train this model. Every time we ran this block, 30 epochs are going to be done and the loss and accuracy data would be updated every 10 inputs. And after ran this block for 6 times, the loss is
close to zero and the accuracy achieved one. So this model has been trained for about 180 epochs in total. Figure 7 shows the piece of code which was used to start to train the model.

![Figure 7](image)

**Figure 7.** A snippet of source code.

4. **Result analysis**

   We’ve trained the CNN model by edge-value changed training dataset and original training dataset separately. And then we used test dataset to evaluate the two models and compared the result.

4.1. **A loss and accuracy during the training phase**

   We set the batch size to 10, which means every ten times the model is trained, the computer calculates and updates the loss and accuracy. Since there are 24 groups of input data in one epoch, there are 2.4 batches of data in one epoch on average. We wrote the same CNN model code twice and used edge value changed data and original data to train these two models separately.

![Figure 8](image)

**Figure 8.** Diagram of loss change during the training phase.

![Figure 9](image)

**Figure 9.** Diagram of accuracy change during the training phase.

We ran 30 epochs each time. And after 6 times of running, the loss is close to zero and the accuracy achieved one. So in total, we ran these two model 180 epochs separately. In order to clearly show the change of loss and accuracy in this process, we calculated mean values of every five epochs’ losses and accuracy and show them in Figure 8 below. In this diagram, model 1 means the CNN model
trained by original data, and model 2 means the CNN model trained by edge value changed data. Then we can infer from the diagram that during training phase, the loss of model 2 was going down more smoothly than model 1.

Figure 9 below is the accuracy change of these two models during the training phase. When the 22th iteration is finished, the accuracy of our model is about 0.3 higher than normal CNN model. We can see from the diagram that the accuracy was fluctuated a lot during training for both two models and finally it came to 1. And the accuracy is 1 means the training phase is finish.

4.2. Comparison of the two models’ result
In order to test the performance of a CNN model, we input the test RSS data into the model and get an output as the prediction of labels. Then we calculate the loss between the real label and label prediction as well as the accuracy. Firstly, Figure 10 below can show the process of loss change during the test phase. And from the fourth test, our model keeps a lower loss than the normal CNN model. This also means our model can have better performance. Finally, our model’s loss can achieve about 0.55 while the traditional model’s loss is about 0.75. Lower loss here means our model’s prediction is closer to the real label value of the test points.

![Figure 10: Diagram of loss change during the test phase.](image1)

Figure 11 below shows the test accuracy change process. And we can infer from this diagram that from the fourth test, our model has higher accuracy than the normal CNN model. Besides, when the training phase is finished, our model can achieve the accuracy about 0.2 higher, which means a better performance too. Finally, our model can achieve about 88% accuracy in prediction while the traditional CNN model can only achieve about 72%. And we believe that this difference is caused by the edge value substitution we did.

![Figure 11: Diagram of accuracy change during the test phase.](image2)
5. Conclusions
This paper provides a wireless indoor localization method using CNN. And it improves the performance of CNN model by changing the value at the edge where signals disappear. This operation can help the CNN to extract features of RSS vectors and have better localization performance. We compared the final loss and accuracy in the test phase and found that after this operation, a CNN model can finally have about 21.2% lower loss and 23.5% higher accuracy than before. Therefore, we can conclude that this way to improve wireless indoor localization CNN models’ performance is useful.

References
[1] Din Marina, Jamil Norziana, Maniam Jacentha and Mohamed Mohamad A 2018 Review of indoor localization techniques *International Journal of Engineering and Technology (UAE)* 7 201
[2] Chen Kong, Song Chunlei and Chen Jiabin 2016 An Indoor Location Fingerprint Algorithm Based on Improved WKNN *Navigation Positioning and Timing* **3**(4) 58
[3] Battiti R, Brunato M and Villani A 2005 Statistical learning theory for location fingerprinting in wireless LANs *Computer Networks: the International Journal of Computer and Telecommunications Networking* **47**(6) 825
[4] Matthew D Zeiler and Rob Fergus 2013 Visualizing and Understanding Convolutional Neural Networks *European Conference on Computer Vision*
[5] Zheng Qinghe, Yang Mingqiang, Tian Xinyu, Jiang Nan and Wang Deqiang 2020 A Full Stage Data Augmentation Method in Deep Convolutional Neural Network for Natural Image Classification *Discrete Dynamics in Nature and Society* **2** 1
[6] Prashant Krishnamurthy 2013 Advanced Location-Based Technologies and Services **2**
[7] Zhang Guolong, Wang Ping, Chen Haibing and Zhang Lan 2019 Wireless Indoor Localization Using Convolutional Neural Network and Gaussian Process Regression *Sensors* **19** 2508
[8] Zheng Qinhe, Yang Mingqiang, Yang Jiajie, Zhang Qingrui and Zhang Xinxin 2018 Improvement of Generalization Ability of Deep CNN via Implicit Regularization in Two-stage Training Process *IEEE Access* **99** 1
[9] Shu Jun, Yang Lu, Chen Yihong, Yang Li and Deng Fang 2019 Research on improved LeNet image classification based on small datasets *Journal of South-central University for Nationalities* **38**(4) 60