Article

Data Augmentation Schemes for Deep Learning in an Indoor Positioning Application

Rashmi Sharan Sinha ¹, Sang-Moon Lee ², Minjoong Rim ¹ and Seung-Hoon Hwang ^{1,*}

¹ Division of Electronics and Electrical Engineering, Dongguk University-Seoul, Seoul 04620, Korea; rashmisinha@dongguk.edu (R.S.S.); minjoong@dongguk.edu (M.R.)

² JMP Systems Co., Ltd, Gyeonggi-do 12930, Korea; smlee@jmpsystem.com

* Correspondence: shwang@dongguk.edu; Tel.: +82-2-2260-3994

Received: 25 April 2019; Accepted: 13 May 2019; Published: 17 May 2019

Abstract: In this paper, we propose two data augmentation schemes for deep learning architecture that can be used to directly estimate user location in an indoor environment using mobile phone tracking and electronic fingerprints based on reference points and access points. Using a pretrained model, the deep learning approach can significantly reduce data collection time, while the runtime is also significantly reduced. Numerical results indicate that an augmented training database containing seven days' worth of measurements is sufficient to generate acceptable performance using a pretrained model. Experimental results find that the proposed augmentation schemes can achieve a test accuracy of 89.73% and an average location error that is as low as 2.54 m. Therefore, the proposed schemes demonstrate the feasibility of data augmentation using a deep neural network (DNN)-based indoor localization system that lowers the complexity required for use on mobile devices.

Keywords: augmentation; deep learning; CNN; indoor positioning; fingerprint

1. Introduction

Identifying the location of a mobile user is an important challenge in pervasive computing because their location provides a lot of information about the user with which adaptive computer systems can be created. The challenge of accurately estimating position both indoors and outdoors has thus received significant attention within both industry and academia. In particular, the successful application of Global Positioning System (GPS) has enabled travelers to move around the world more freely. However, GPS is sensitive to occlusion and does not work in indoor environments. A large number of methods has been proposed to overcome this limitation, including estimating indoor location using mobile devices such as smartphones. Measuring the intensity of a received signal makes interior localization using wireless signals such as Wi-Fi possible. Measurement-based wireless positioning systems infer position based on the time of arrival (TOA) or time difference of arrival (TDOA) of the signals. However, general wireless signal receivers are not capable of measuring round-trip times or angles. Additional devices are thus required, which makes this type of system impractical for many applications. An alternative option is the use of fingerprint-based approaches, which do not need any special devices and are thus more feasible. The fingerprint method proposed in the present study consists of two stages (Figure 1). In the offline stage, the received signal strength indications (RSSIs) from all access points (APs) are collected from known positions, referred to as reference points (RPs), to build a fingerprint database for the environment. Therefore, each RP has a fingerprint characterized by its position and the captured RSSIs from all APs at that location. At the positioning stage, the currently captured RSSIs are matched with those of the RPs and position is determined using the positions of several of the best-fitting RPs. However, a major issue for accurate fingerprint-based

localization is the variation in RSSIs due to the fluctuating nature of wireless signals, caused by multipath fading and attenuation by static or dynamic objects such as walls or moving people.

Figure 1. The fingerprint method (For offline deep neural network (DNN) training; a four-layer DNN-based localizer is trained to extract reliable features from massive noisy received signal strength indication (RSSI) samples from a pre-built fingerprint data set. For online positioning, the preprocessed RSSI readings are fed into the localizer to estimate the final position).

It is also necessary to collect more RPs to allow more accurate positioning, especially when the target environment has a large area, which leads to an extremely large fingerprint database. Consequently, the challenge in wireless positioning is how to extract reliable features and optimize the mapping function using a large collection of RPs with widely fluctuating RSSI signals. The methods represent a form of shallow learning architecture that has limited modeling and representational power when dealing with large and noisy volumes of data. To extract complex structures and build accurate internal representations from rich data sources, human information processing mechanisms (e.g., vision and speech) suggest the need for deep learning architecture consisting of multiple layers of nonlinear processing [1]. Deep learning simulates the hierarchical structure of the human brain, processing data from a lower to a higher level, gradually producing more semantic concepts. Deep neural networks (DNNs) have been employed to address these types of issue with notable success, outperforming state-of-the-art techniques in certain areas such as vision [2], audio [3], and robotics [4,5]. However, the use of DNNs in Wi-Fi localization has largely remained uninvestigated. With this in mind, we present a novel Wi-Fi positioning method based on deep learning. In this paper, we propose data augmentation schemes to estimate user position and demonstrate how our methods can be employed in convolutional neural network (CNN) settings.

A deep learning structure is built to extract features from widely fluctuating massive Wi-Fi data, one which automatically performs probabilistic position estimation. In addition, a CNN-based localizer is introduced to formulate the relationship between adjacent position states and to intuitively reduce variation in the estimates. In this work, a Wi-Fi-based indoor RSSI fingerprint positioning system is proposed and implemented using data augmentation, which is a common technique that has been proven to benefit the training of machine learning models in general and deep architecture in particular. It either speeds up convergence or acts as a regularizer, thus avoiding overfitting and increasing generalization power [6,7]. Data augmentation typically involves applying a set of transformations to either the data space or the feature space, or both. The most common augmentations are performed in the data space, generating new samples by applying transformations to the already existing data. There are many transformations that can be applied, such as translation, rotation, warping, scaling, color space shifts, and cropping. The goal of these transformations is to generate more samples to create a larger data set, to prevent overfitting, to presumably regularize the model, to balance the classes within the database, and to even synthetically produce new samples that are more representative

for the use case or the task at hand. Augmentation is especially useful for small data sets and has been proven successful in many previous reports. For instance, in Shan et al. [8], a data set of 1500 portrait images was augmented by applying four new scales (0.6, 0.8, 1.2, and 1.5), four new rotations (-45 , -22 , 22 , and 45), and four gamma variations (0.5, 0.8, 1.2, and 1.5) to generate a new data set of 19,000 training images. This process increased the accuracy of their system for portrait segmentation from an Intersection over Union (IoU) score of 73.09 to one of 94.20 when the augmented data set was included for fine-tuning. Although this augmentation example was implemented for a small data set, the synthesizing process with rotation increases the complexity of the algorithms. In our augmentation technique, a new data set is generated using AP number variation and uniform random numbers. In terms of complexity, our augmentation technique is thus more convenient to implement. The remainder of the paper is structured as follows. First, Section 2 describes the augmentation schemes to conduct the experiment. The augmented database is input into the classifier for training in Section 3. In Section 4, the simulation results for the prediction of position using the CNN classifier is compared with results using the unaugmented data set. Finally, the positioning success rate of the proposed system is verified in a real-time indoor positioning experiment.

2. Data Augmentation Schemes

Data augmentation is a common method in deep learning used to reduce the effect of overfitting. The idea is to expand an existing data set using only the available data so that the learning algorithm can more effectively extract those features essential to the task. To train deep learning models, typically big data sets are required, usually from manual data collection or from already existing databases. However, in some cases only a limited data set is available. Therefore, to expand the size of the data set, data augmentation can be employed. The complex indoor environment and APs may cause problems [9,10] because of the limited coverage of Wi-Fi APs and faulty RSSI measurements. The purpose of data augmentation in this case is to detect and remove faulty measurement data or to remove invalid data, thus improving the accuracy and efficiency of the entire positioning system by creating a database representation that is more suitable for downstream deep learning classifiers. Data augmentation adds value to base data by adding information derived from internal and external sources within the database. It can also reduce the manual intervention required to develop meaningful information and gain insight from business data, as well as significantly enhancing data quality. In this way, we can produce multiple copies of available data with slight variations. Some common techniques used in data augmentation include extrapolation [11], in which the relevant fields are updated or assigned values based on heuristics; tagging [12,13], in which common records are tagged to a group, making it easier for the group to be understood and differentiated; aggregation [14], in which values are estimated for relevant fields if needed using mathematical averages and means; and probability [15], in which values are populated based on the probability of events based on heuristics and analytical statistics.

The data set collected during the measurement is in text format. However, the deep learning code is designed for a comma-separated values (CSV) input file, so the text files are converted into a CSV file for training and testing. The training and testing CSV files contain 257 columns and 74 RPs in the 257th column as labels (Figure 2). The data conversion of the text dataset from the experiment is conducted using Python. Folders containing text files are used as input to file converter code designed in Python. The APs visible from one RP each have a unique media access control (MAC) address. After defining the number of visible APs (N), the individual MAC addresses and their RSSI values are extracted. Each MAC address has an individual column. If the current MAC address is already present in the header list, the corresponding RSSI is placed below that MAC address; if not, a new column is allocated to this MAC address. For the other MAC addresses that are not present in the current text file for the current row, the RSSI values are replaced with zero. The process continues until all of the text files from all of the folders are processed. If the total number of columns in final CSV file is lower than 256, then the remaining columns are filled with zeros. If the total number of columns is greater than

256, then the extra columns are deleted on the basis of minimum RSSI entries for an individual MAC address. If the total number of columns in the final CSV file is equal to 256, then the reference data set is complete. Figure 2 shows the CSV format for input file where NN represent the RSSI values.

	A	B	C	D	E	F	G
1	MAC ₁	MAC ₂	MAC ₃	MAC ₄	MAC ₅	MAC ₆	MAC ₇
2	NN ₁₁	NN ₁₂	NN ₁₃	NN ₁₄	NN ₁₅	NN ₁₆	NN ₁₇
3	NN ₂₁	NN ₂₂	NN ₂₃	NN ₂₄	NN ₂₅	NN ₂₆	NN ₂₇
4	NN ₃₁	NN ₃₂	NN ₃₃	NN ₃₄	NN ₃₅	NN ₃₆	NN ₃₇
5	NN ₄₁	NN ₄₂	NN ₄₃	NN ₄₄	NN ₄₅	NN ₄₆	NN ₄₇
6	NN ₅₁	NN ₅₂	NN ₅₃	NN ₅₄	NN ₅₅	NN ₅₆	NN ₅₇

II	IU	IV	IW
MAC ₂₅₄	MAC ₂₅₅	MAC ₂₅₆	256
NN ₁₂₅₄	NN ₁₂₅₅	NN ₁₂₅₆	L ₁₁
NN ₂₂₅₄	NN ₂₂₅₅	NN ₂₂₅₆	L ₁₂
NN ₃₂₅₄	NN ₃₂₅₅	NN ₃₂₅₆	L ₁₃
NN ₄₂₅₄	NN ₄₂₅₅	NN ₄₂₅₆	L ₁₄
NN ₅₂₅₄	NN ₅₂₅₅	NN ₅₂₅₆	L ₁₅

Figure 2. The comma-separated values (CSV) file format used as the input file for data augmentation.

2.1. Scheme 1: Augmentation with the Total Number of APs

In a fingerprinting algorithm, the fingerprint of each training point is a sequence of an equal number of measurements from different APs. Therefore for M measurements per RP with A APs, the fingerprint at each RP has the dimensions $A \times M$. The aim of the algorithm is to classify a test point as one of those partitions using CNN classifiers. However, if the problem contains only a few observations for a given class, the CNN will not be able to gather enough information for a particular location. Therefore, multiple observations are required for each class. To increase the training data set to an appropriate size, augmentation is employed. In the first augmentation scheme, the data is augmented based on the number of APs in each reading. In this augmentation scheme, the number of APs visible at one RP is used to augment the data. A global declaration for the total number of MAC addresses, i.e., the header and a constant value, is made in this scheme, with the constant value used in this paper set at 5. This constant is used throughout Scheme 1 to introduce diversity to the augmented numbers. The diversity of the data is controlled by repeating the same RSSIs and changing only one RSSI by subtracting the constant. The total number of APs with a nonzero RSSI is calculated for each RP. These nonzero values will be the number of repetitions for the current RSSI. The RSSIs for each RP are collectively arranged in the form of rows containing 256 numbers in the present study. The zero RSSI value is repeated same as nonzero values. Most of the RSSIs are 0 because a maximum of 26 APs is visible from one RP. Therefore, approximately 230 RSSIs are 0. In the example shown in Figure 3, a reading containing nine APs can be used to produce nine copies to obtain 10 readings at the reference point under consideration. For every copy, 5 is subtracted from the RSSI value of one AP at a time. In this way, 10 copies of the original reading can be made with only a single difference in one RSSI value for each. Scheme 1 focuses on less detailed data and keeps the augmentation simple with respect to the RSSIs. From a small input data size of 3 to 7 kilobytes, this data augmentation technique increases the size to ~30–50 megabytes. The pseudocode for Scheme 1 outlines the technique in more detail. A detailed pseudocode for this augmentation scheme is presented in Scheme 1.

1. **Define** Header
2. **Define** Constant number 'C'
3. **for** input csv file
4. **for** total number of reference points (RP)
5. **for** each RSSI value at reference point 'RP'
6. Calculate total number of RSSI $\neq 0$ 'N' on that RP
7. Repeat each RSSI = N
8. Subtract C from each RSSI for (N - 1)
9. **end for**
10. **end for**
11. **end for**

Scheme 1. Pseudocode for data augmentation with total number of AP.

Figure 3. Scheme 1 augmentation: (a) RSSI values present at one reference point with nine visible APs. (b) The augmented data set with nine copies and changes in only one RSSI value by 5. The solid boxes represent RSSIs without any change, while the dotted red box represent RSSIs with a difference of 5.

2.2. Scheme 2: Augmentation with the Mean and Uniform Random Numbers

This data augmentation technique uses mean value and uniform random numbers to add information into the reference data set. The total number of RSSIs in the reference data set is increased in a fashion such that overall RSSI remains in range of minimum and mean with original RSSI. To identify the potential range for the uniform random number generator, the calculation of the mean RSSI value for each RP is conducted. The global repetition number N is declared and the input data set established in the form of CSV files. For each RP, the mean is calculated for the visible APs, excluding those with $\text{RSSI} = 0$. After calculating the mean, the range is determined for the generation of uniform random numbers. This range is calculated by subtracting the current RSSI value from the mean. The total number of random numbers will be equal to N . For example, let the global N be 60, the current RSSI element be 33, and the RP mean be 45. In this case, the potential range for the uniform random numbers will be 12 ($= 45 - 33$), which means the uniform random number will generate any of the 13 numbers (including 33) within the 33–45 range 60 times. If the range is equal to zero, the current RSSI will be generated 60 times. In Figure 4, there are only nine visible APs at one RP. Figure 4a shows the input data for augmentation, while Figure 4b shows the output after augmentation. The RSSI values 53 and 28 in Figure 4a have zero difference from the mean, thus there is no change in their augmented RSSI values shown in Figure 4b. Each RSSI element forms an image of 16×16 pixels. The addition of a random number to the augmented data set increases the information for each image by keeping the random number within the range set by the mean and the current RSSI. Compared to Scheme 1, the information for every image in Scheme 2 increases by the constant number N ($N = 60$ in our paper). Therefore, the information in each image for machine learning is equal. However, in Scheme 1, the size of the augmented data set is equal to the number of visible APs, which is not constant. Hence, the information available for machine learning is limited by the number of visible APs. The input file size for Scheme 2 is 3 to 7 kilobytes. The output augmented data size is approximately 300 to 700 megabytes. This larger data set helps the CNN classifier to extract more information for each location. The mean of the data set for the individual APs in each column is calculated with respect to individual RPs. Therefore, there are 74 mean values for each AP. The augmentation further generates N uniform random numbers to achieve the desired augmentation. The pseudocode explains the coding details for Scheme 2.

Figure 4. Scheme 2 augmentation. (a) RSSI data set without augmentation. (b) For $N = 60$, there are a total of 60 augmented RSSI data sets with mean and uniform random numbers. The solid black boxes represent the original RSSI value and the dotted red boxes represent augmented RSSI values.

```

1. Define 'N'
2. Define Header
3. for input csv file
4.   for total number of reference points (RP)
5.   for individual AP address points RP
6.     if current RSSI == 0
7.       Repeat 0 for N times
8.     for current RSSI ≠ 0, do summation 'S'
9.     for each S calculate mean 'M'
10.      if |M - RSSI| ≠ 0
11.        Calculate range 'R' for difference between |M - RSSI|
12.        Generate N random numbers for corresponding R
13.      else if |M - RSSI| == Mean
14.        Write current RSSI for N times
15.        Repeat for all the RSSI in current RP
16.      else |M - RSSI| == 0
17.        Write current RSSI for N times
18.        Repeat for all the RSSI in current RP
19.      end else
20.    end else if
21.  end else
22. end for
23. end for
24. end if
25. end for
26. end for
27. end for

```

Scheme 2. Pseudocode for data augmentation with mean and uniform random numbers.

3. System Model and Experiment Setup

3.1. System Model

In our CNN model, a 5-layer network is designed to predict 74 classes. The input image is generated from the RSSI values received during the experiment by the 74 RPs. At each RP, the RSSI value is recorded for 256 APs, though only a small subset of these APs are visible at each RP. These RSSI values from different APs create a 16 x 16 image. As shown in the example in Figure 5, there are a total of nine visible APs out of 256 with RSSI values between 25 to 70, with the other APs having a value of 0. The RSSIs from different APs are converted into a grayscale image. The image has different levels of brightness depending on the recorded RSSI values, with higher RSSI values being brighter. As shown in Figure 5a, the highest RSSI value is 70, which produces the brightest spot in the grayscale image shown in Figure 5b, while the lowest value is 25, which is represented by the darkest nonblack spot. RSSI values of 0 produce no brightness, thus the remaining 247 spots are black. Similarly, the input RSSI files for the other 73 RPs will produce different images for input into the deep learning network. This leads to a total of 9602 input images for the 74 RPs without augmentation. With augmentation, there are total 122,760 input training images using Scheme 1 augmentation and 585,722 input training images using Scheme 2 augmentation. The total number of test images for the lab simulations is 1479.

Figure 5. Deep learning input file conversion from a CSV file to an image. (a) Input CSV readings of the nine visible RSSIs from a total of 256 APs. (b) Converted grayscale image with nine bright spots representing APs visible at the RP.

A deep learning architecture is useful for audio, video and image data types. Especially, feed-forward neural networks (FNNs) and artificial neural networks (ANNs) spur the idea for CNNs in which the output from one layer is the input for the subsequent layer. In FNNs, each neuron is interconnected and it has weight associated to each neuron in the following layer. As a result, the number of required connections in this type of network rapidly grows as the input size increases to an unmanageable level. For example, in [16] if the input to an FNN network is through a VGA camera ($640 \times 480 \times 3$ pixels), there would be a weight difference of 921,600 between an input neuron and a single hidden neuron. In addition first hidden layer needs to comprise of thousands of neurons to manage the dimensionality of the input, leading to a model with billions of weight and all needed to be learned. It is very difficult to work with this many weights since it increases the computation complexity as well as the memory requirements. A CNN is a variant of the standard multilayer perceptron (MLP). A significant advantage of this method compared with conventional approaches, especially for pattern recognition, is its ability to reduce the dimensions of the data, extract features sequentially, and classify the image at the output of CNN network [17]. Our CNN network comprises five layers, with the first layer having input grayscale images of size $16 \times 16 \times 1$, Rectified Linear Unit (ReLU), and dropout. Due to the small size of the input data set, max pooling is not used in the first layer. The second layer consists of a 16×16 convolution with ReLU and then an 8×8 max pooling layer with a total of 18,496 parameters. This produces output for the third 8×8 convolution layer with ReLU and a 4×4 max pooling layer. This output is fed directly to a fully connected (FC) layer with 3072 nodes, which leads to the next hidden FC layer with 1024 nodes. Finally, the output is calculated using a softmax layer with 74 nodes, which is the total number of RPs in our set-up. The inner width is 1024, and the first four layers use a dropout of 0.5. The learning rate of our model is 0.001. The total number of parameters in our model is 2,266,698. Figure 6 provides a complete summary of the CNN model used in the present study.

Figure 6. CNN architecture for the indoor Wi-Fi positioning system proposed in this study.

3.2. Experiment Setup

RSSI fingerprint data collection and the final experiment were both performed on the 7th floor of the new engineering building at Dongguk University, Seoul, South Korea. As shown in Figure 7, the 52×32 m target area is divided into 74 target RPs spaced at intervals of 2 m. The positioning server used in this study is a Dell Alienware Model P31E (Alienware, hardware subsidiary of Dell, Miami, FL, USA), while a Samsung SHV-E310K smartphone (Yongin-si, Gyeonggi-do, Korea) is used for data collection. The fingerprint database construction, classification (i.e. position prediction), and online experimental setup are developed using Python. The experimental system setup, in which an Android device listens to the RSSIs from surrounding APs, is depicted in Figure 8.

The data read by the Android device is stored in a buffer. If there is an error in the recorded data, an error message is displayed on the serially connected console. Otherwise, the RSSI data is stored in the buffer and, after a complete scan, is transferred by the Android console connected by an interface cable to the server through a Wi-Fi AP. The server determines the Android device's location by comparing the measured RSSI values with reference data. The Android device used in our experiment is shown in Figure 8. It is serially connected to the Android console and processes the RSSIs from surrounding APs with its CPU unit. The operating frequency of the device is 2.412–2.480 GHz for the 802.11bgn wireless standard. The input/output sensitivity is 15–93 dBm. The reference file, represented by REF in text file format, is saved as a REF000000XX file, where XX is the RP number. The size of each file is 1 kilobyte (Figure 9).

In collecting the data, the wireless mobile phone first measures the RSSIs from the available APs at each RP. The measurements are combined with the MAC addresses of the APs and the RP label as a single fingerprint data point and transmitted to the positioning server. Considering the multipath effects that exist in indoor environments, the reference data (Figure 9) are collected five times at each RP. Data is collected 4 times a day and total for 7 days. In total, 10,360 RSSI fingerprint data points are acquired using this process. All of these data points are saved as one CSV file on the positioning server. The data set is then used as reference data set to train the deep learning classifier on the server. The proposed indoor positioning system depends on the installed APs and the observation of their RSSI values using a user's mobile device. RSSI values are highly vulnerable to interference from walls, human body movement, and the direction in which the user is facing. Therefore, the overall accuracy of the indoor positioning system can be weakened. To overcome this problem, we build our data set using data collected from four orientations, i.e., forward, backward, left, and right, at each individual reference point. The flexibility of the system is higher if the reference data set includes all four orientations. The location of a user can thus be identified without them having to face in one direction.

Figure 7. Indoor environment (radio map) for data collection divided into 74 reference points (2 m × 2 m each).

Figure 8. Illustration of the proposed fingerprint-based Wi-Fi positioning system.

Name	Type	Size
REF_00000001	Text Document	1 KB
REF_00000001	Text Document	1 KB
REF_00000001	Text Document	1 KB
REF_00000001	Text Document	1 KB
REF_00000001	Text Document	1 KB
REF_00000002	Text Document	1 KB
REF_00000002	Text Document	1 KB
REF_00000002	Text Document	1 KB
REF_00000002	Text Document	1 KB
REF_00000002	Text Document	1 KB

Figure 9. Reference data files for RP 1 and RP 2 collected during the experiment.

4. Numerical Results

4.1. Simulation Results

In order to evaluate the validity of our approach, we create several data sets over a period of four weeks. These data sets are then used to assess which CNN layer is best used to transfer knowledge from classification to indoor positioning and to determine the optimal classification algorithm. We show that a relatively simple classification model fits the data well, producing generalization results over a one-week period of 91% in the lab-based simulations and 89% in real-time testing with Scheme 2. The long-term introduction of new APs and drift in the existing APs need to be trained and learned. To generate the data set, the data is gathered at the 74 RPs over 7 days in four directions. The data set is then divided into four sets (Set 1: 7 days of data; Set 2: 5 days of data; Set 3: 3 days of data; and Set 4: 2 days of data), and each set is divided further into separate cases based on the ratio of reference to trial data. For example, Set 1 (7 days of data) is divided into the three cases 6-1, 5-2, and 4-3. Table 1 summarizes the data sets and cases. The dataset with 7 days of data has the maximum number of input files, thus its overall test accuracy is higher than that of the other sets (Table 2). Set 1/Case 1, which has six days of data set for training and 1 day of data for testing, produced a highest test accuracy of 91%, with a minimum loss of 0.3. Similarly, Set 2/Case 1 with a 4:1 data set reference to testing data ratio has a highest accuracy of 88%, with a loss of 0.2. Set 3 exhibits a similar accuracy to that of Set 2. Set 4 produces the lowest accuracy with 84%. Therefore, it can be concluded that training with only 1 day of data is not sufficient for the CNN classifier. From these lab simulations, it is clear that a large data set is very important for a classifier. The accuracy of our CNN classifier drops with a decrease in the size of the training data set. Thus, with an augmented data set, the accuracy of the complete system can be increased in both lab simulations and in real-time positioning.

Table 1. Data sets and cases.

DATA SET	Case 1	Case 2	Case 3
	Reference-Test		
Set 1: 7 Days	6-1	5-2	4-3
Set 2: 5 Days	4-1	3-2	-
Set 3: 3 Days	2-1	-	-
Set 4: 2 Days	1-1	-	-

Table 2. Lab simulation results for accuracy and loss.

Sets	Approaches	Case 1		Case 2		Case 3	
		Loss	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
Set 1	Scheme 1	1.4	90.46	1.3	89.85	1.7	88.88
	Scheme 2	1.0	91.32	1.2	90.09	1.1	90.45
	Without Augmentation	1.5	89.45	1.5	89.55	1.4	88.83
Set 2	Scheme 1	1.3	86.73	0.9	87.10	-	-
	Scheme 2	1.0	87.34	1.2	88.08	-	-
	Without Augmentation	1.3	86.87	1.3	87.33	-	-
Set 3	Scheme 1	1.4	88.65	-	-	-	-
	Scheme 2	1.0	88.89	-	-	-	-
	Without Augmentation	1.2	87.67	-	-	-	-
Set 4	Scheme 1	0.8	84.11	-	-	-	-
	Scheme 2	1.0	84.79	-	-	-	-
	Without Augmentation	0.9	54.27	-	-	-	-

Figure 10a,b shows the loss and test accuracy for 6-1 Reference-Test Set 1, Case 1. The loss value for Schemes 1 and 2 starts at 2.8 and 1.7, which reduced to 0.4 and 0.3 after 200 epochs, respectively,

The unaugmented data set have initial loss value at 4.7 which comes down to 1.0, after 200 epochs. The loss decreases after 200 epochs which shows robustness of the CNN model however, the accuracy of the model also decreasing which is undesirable for indoor positioning application. The highest accuracy achieved during lab simulations for each method is used as optimum value to generate the metafile [7] for real time testing for each method. Therefore the highest accuracies for Scheme 1, Scheme 2, and the unaugmented data set are chosen in this work. The highest accuracy for 6-1 Reference-Test Set 1 is 90.46% (after five epochs, loss = 1.4), 91.32% (after three epochs, loss = 1), and 89.45% (after 43 epochs, loss = 1.5) for Scheme 1, Scheme 2, and unaugmented data, respectively. The accuracy of all three methods fall to 84.30% for Scheme 1, 85.50% Scheme 2, and 88.43% unaugmented data set after 200 epochs. Figure 10c,d shows the loss and test accuracy for 5-2 Reference-Test Set 1, Case 2. The initial and final loss values after 200 epochs are 2.8, 1.8, and 4.7, and 0.4, 0.3, and 1 for Scheme 1, Scheme 2, and the unaugmented data set, respectively. The highest lab accuracy achieved for this data set is 89.85% (after 7 epochs, loss = 1.3) for Scheme 1, 90.09% (after 2 epochs, loss = 1.2) for Scheme 2, and 89.55% (after 65 epochs, loss = 1.5) for unaugmented data set. After 200 epochs the accuracies are 82.91%, 81.50%, and 88.01% for each method, respectively. Figure 10e,f represents the loss and test accuracy for 4-3 Reference-Test Set 1, Case 3. The four days of reference data set is used as training data set for CNN classifier in this case. The initial loss values for Scheme 1, Scheme 2 and unaugmented data are 3.2, 2.0, and 4.9, while the final loss values after 200 epochs are 0.3, 0.2, and 1.0. The highest accuracies for this case are 88.88% (after seven epochs, loss = 1.7), 90.45% (after three epochs, loss = 1.1) and 88.83% (after 86 epochs, loss = 1.4). After 200 epochs the accuracies are 82.91%, 81.50%, and 88.01% for each method, respectively. Figure 11a,b shows the loss and test accuracy for 4-1 Reference-Test Set 2, Case 1. Total four days of reference data set is used as training data set for CNN classifier. The initial loss value for this case is 3.0, 1.9, and 4.8 for Scheme 1, Scheme 2, and the unaugmented data set, respectively. The highest accuracies achieved during lab simulation for this case are 86.73% (after six epochs, loss = 1.3) for Scheme 1, 87.34% (after three epochs, loss = 1.0) for Scheme 2, and 86.87% (after 77 epochs, loss = 1.3) for unaugmented data set. The accuracies after 200 epochs are 77.93%, 74.80%, and 84.63% for each method, respectively. Figure 11c,d shows the loss and test accuracy for 3-2 Reference-Test Set 2, Case 2. The total three days of the reference data set is used as training data set for CNN classifier. The initial loss values for this case are 3.1, 2.0, and 4.9, while the final loss values are 0.2, 0.1, and 0.8 for Scheme 1, Scheme 2, and unaugmented data, respectively. The highest accuracies achieved in this case are 87.10% (after 15 epochs, loss = 0.9) for Scheme 1, 88.08% (after two epochs, loss = 1.2) for Scheme 2, and 87.10% (after 75 epochs, loss = 1.3) for unaugmented data. The accuracies after 200 epochs are 80.90%, 78.50%, and 85.57% for each method, respectively. Figure 12a,b shows the loss and test accuracy for 2-1 Reference-Test Set 3, Case 1. In this case only two days of reference data set is used as training data set for CNN classifier. The initial loss values are 3.7, 2.3, 5.1, and final loss values are 0.2, 0.1, and 0.8 for Scheme 1, Scheme 2, and unaugmented data set. The highest accuracies achieved in this case are 84.11% (after six epochs, loss = 1.4) for Scheme 1, 84.79% (after three epochs, loss = 1.0) for Scheme 2, and 54.27 (after 109 epochs, loss = 0.2) for unaugmented data set. The accuracies decrease after 200 epochs to 76.07%, 72.23%, and 54.27% for each method, respectively. Figure 13a,b shows the loss and test accuracy for 1-1 Reference-Test Set 4, Case 1. In this case, only one day of the reference data set is used as training data set for CNN classifier. The initial and final loss values for Scheme 1, Scheme 2 and unaugmented data sets are 4.2, 2.7, 5.2, and 0.1, 0.0, and 0.9 for each method, respectively. The highest accuracies achieved during lab simulation for this case are 84.11% (after 16 epochs, loss = 0.8) for Scheme 1, 84.79% (after two epochs, loss = 0.8) for Scheme 2, and 54.27% (after 200 epochs, loss = 0.9). The accuracies after 200 epochs for this case are 76.07%, 72.23%, and 54.27% for each method, respectively.

Figure 10. Lab simulation results. (a,c,e) Set 1 loss for the three cases. (b,d,f) Set 1 test accuracy for the three cases.

Figure 11. Lab simulation results. (a,c) Set 2 loss for the two cases. (b,d) Set 2 lab simulation test accuracy for the two cases.

Figure 12. Lab simulation results. (a) Set 3 loss and (b) Set 3 lab simulation test accuracy.

Figure 13. Lab simulation results. (a) Set 4 loss and (b) Set 4 lab simulation test accuracy.

4.2. Experiment Results

There are three approaches for the proposed CNN positioning system, as mentioned in Section 4.1. The highest accuracy of each scheme is: Scheme 1 shows 90.46%, Scheme 2 shows 91.32%, and Without Aug. shows 89.45%. Each approach is tested in an online real-time experiment to verify the actual performance of the proposed system. In the experiment, a mobile phone carried by a user captures the real-time RSSIs of visible APs and sends them with the associated MAC addresses to the positioning server. The trained classifier model is loaded and used to predict the user's position based on the received data. In our online experiment, one user carrying a mobile phone moved from RP 1 to RP 74 (i.e., in a forward direction) and then moved from RP 74 to RP 1 (i.e., in a backward direction). At each RP, RSSI measurements were taken and the user's position predicted five times in sequence. In the simulation, the test data set is constructed from data collected from four directions. As a result, the simulation results should be compared with the averaged results for both the forward and backward experiment.

As shown in Table 3, the accuracy achieved during real-time positioning is 89.73% with Scheme 2, which was slightly lower than the lab simulation results presented in Table 2 in terms of the highest accuracy (91.32%). Scheme 1 achieves an accuracy of 88.11%, 2.35% lower than the lab simulations. Table 4 compares the average location error for the three augmentation schemes. The augmentation schemes were tested for a total of 6 days to produce 30 test runs for each RP. It can be seen that the average mean location error for Scheme 2 is 2.54 m. In comparison, the mean location error for Scheme 1 is 2.70 m. The effectiveness of the augmentation schemes is also evaluated in terms of positioning accuracy, which is defined as the cumulative percentage of location error within a specified distance (Figure 14). Overall, the augmentation schemes are more effective, outperforming the unaugmented data over the entire range of the graph. Schemes 1 and 2 do not differ greatly in terms of positioning accuracy, e.g., cases where the error distance is within 5 m. Both schemes have probabilities of above 90% of being within an error distance of 5 m. However, for cumulative distribution functions over 90%, the positioning accuracy of Scheme 1 falls behind Scheme 2. Under 90%, the error distance for the augmentation schemes is about 2.5 m, with Scheme 2 ~0.15 m more accurate compared with Scheme 1. The gap between the two increases gradually and eventual the error distance rises to nearly 8 m. As shown in Table 3, the accuracy without augmentation is 74.68%, while the accuracy with Scheme 1 augmentation is 88.11% and Scheme 2 augmentation is 89.73%. The overall accuracy is improved by 13.43% for Scheme 1 and 15.05% for Scheme 2.

Figure 14. Cumulative distribution function curves comparing position accuracy using data augmentation and unaugmented data.

Table 3. Experimental results for real-time positioning.

	Accuracy (%) (0 RP Margin)	Accuracy (%) (1 RP Margin)	Accuracy (%) (2 RP Margin)
Scheme 1	33.78	71.76	88.11
Scheme 2	35.81	74.05	89.73
Without Augmentation	25.67	60.54	74.68

Table 4. Average positioning error for the augmentation schemes.

	Mean (m)	Standard Deviation (m)
Scheme 1	2.70	3.43
Scheme 2	2.54	3.25
Without Augmentation	5.62	8.68

In Figure 15, the average test accuracy for both augmentation schemes are compared for a real environment. The environment conditions are the same for all six days of testing. Day 3 has the highest accuracy (89.73% for Scheme 2 and 88.11% for Scheme 1), while Day 5 and Day 2 have the lowest test accuracies (86.76% for Scheme 2 and 84.05% for Scheme 1, respectively). Figure 15 shows the localization results for the augmented and unaugmented data: augmentation using the total number of APs in Scheme 1, augmentation using only the mean and uniform random numbers in Scheme 2, and the raw data. Overall, Scheme 2 performs slightly better than the others. For example, Scheme 2 has a mean error of 2.54 m when using a training data set with 7 days of data to assist in localization, while Scheme 1 has a mean error of 2.70 m. In other words, Scheme 2 performs 6% better than Scheme 1. The performance of Scheme 2 in terms of accuracy is also very close to the lower error bound, for which the median error is also 2.54 m. The unaugmented data performs poorly due to insufficient training data. It is interesting to observe that the raw data set performs better than the complex augmented data sets in the lab simulations. The reason is that the real environment is extremely complex, with a number of different types of wall, resulting in more localization errors. Figure 14 illustrates the improvement in terms of accuracy with the augmented data set as compared to raw data set. Overall, the proposed localization techniques can provide a higher accuracy (i.e., a smaller error). We observe that a DNN could exploit the additional measurements very well, making it a promising technique for environments with a high density of APs. In addition to the greater performance using Scheme 2 augmentation, extant fingerprinting approaches demand a laborious

offline calibration phase. For example, our proposed method requires 140 observations per direction and 560 observations per RP. The data needs to be measured in four directions. Thus, when applying the same calibration process to our experiment space with 60 test runs for each RP, at least 1 hour and 16 min (1 s per observation) is required.

Figure 15. Average test accuracy for six days with 30 test runs at each RP.

5. Conclusions

This research presents a novel approach to indoor localization that has been proven to be fast enough to run in real-time and robust to changes in environmental conditions. In this paper, we developed augmentation techniques for a deep learning scheme for Wi-Fi based localization. In deep learning, in the offline stage, a four-layer CNN structure is trained to extract features from fluctuating Wi-Fi signals and to build fingerprints. In the online positioning stage, the proposed augmentation technique with a CNN-based localizer estimates the position of the target. Three approaches to data augmentation for the CNN positioning system are proposed: Scheme 1, Scheme 2, and without augmentation. Each scheme produces a highest simulation success rate of 90% or higher. The real-time testing of Scheme 2 augmentation produces an accuracy of 89.73% with a 2-RP margin using seven days of data. This means that Scheme 2 augmentation with a CNN is more capable of handling the instability and variability of RSSIs for Wi-Fi signals in a complex indoor environment, and thus is more powerful for use in classification tasks in fingerprint indoor positioning. Future research will seek to expand the algorithm to work seamlessly throughout an entire building. The first step in this is testing and adapting the algorithm to work inside rooms rather than only hallways. Next, the individual maps of an entire building would need to be learned, and some way to identify which room or hallway the user is traveling through would need to be developed. This could be in the form of a global classification model that is trained to predict which local area of the building the user is located in. Then, from there, a local map and classifier could take over.

Author Contributions: R.S.S. and S.-H.H. contributed to the main idea of this research work. R.S.S. and S.-M.L. wrote computation codes, performed the simulations, experiments, and database collection. M.R. confirmed numerical results of the work. The research activity was planned and executed under the supervision of S.-H.H., R.S.S. and S.-H.H. contributed to the writing of this article.

Funding: This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program-Industrial site core technology Early development project) (10084909, Development of Improved RTLS engine and IoT tag using machine learning technique reflecting real environments) funded by the Ministry of Trade, Industry, and Energy (MOTIE, Korea). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2016R1A2B1008953).

Conflicts of Interest: The authors declare no conflict of interest regarding the publication of this article.

References

1. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [[CrossRef](#)] [[PubMed](#)]
2. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014; pp. 675–678.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
4. Szegedy, C.; Liu, W.; Jia, Y.Q.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Computer Vision and Pattern Recognition, Boston, MA, USA, 8–10 June 2015; pp. 1–9.
5. Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In Proceedings of the International Conference on Machine Learning, Beijing, China, 16–21 June 2014; pp. 647–655.
6. Graves, A.; Mohamed, A.R.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
7. Haider, A.; Wei, Y.; Liu, S.; Hwang, S.-H. Pre- and Post-Processing Algorithms with Deep Learning Classifier for Wi-Fi Fingerprint-Based Indoor Positioning. *Electronics* **2019**, *8*, 195. [[CrossRef](#)]
8. Shen, X.; Hertzmann, A.; Jia, J.; Paris, S.; Price, B.; Shechtman, E.; Sachs, I. Automatic portrait segmentation for image stylization. *Comput. Graph. Forum* **2016**, *35*, 93–102. [[CrossRef](#)]
9. Murata, M.; Ahmetovic, D.; Sato, D.; Takagi, H.; Kitani, K.M.; Asakawa, C. Smartphone-based Indoor Localization for Blind Navigation across Building Complexes. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom), Athens, Greece, 19–23 March 2018; pp. 1–10.
10. Singh, V.; Gorish, A.; Ujwal, B.V.S. Ensemble based real-time indoor localization using stray WiFi signal. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–14 January 2018; pp. 1–5.
11. DeVries, T.; Taylor, G.W. Dataset Augmentation in Feature Space. *arXiv* **2017**, arXiv:1702.05538.
12. Wang, Y.; Patel, A.; Jin, H. A New Concept of Deep Reinforcement Learning based Augmented General Tagging System. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–25 August 2018; pp. 1683–1693.
13. Iqbal, T.; Kong, Q.; Plumbly, M.D.; Wang, W. General-purpose audio tagging from noisy labels using convolutional neural networks. In Proceedings of the DCASE2018 Workshop on Detection and Classification of Acoustic Scenes and Events, Surrey, UK, 19–20 November 2018; pp. 212–216.
14. Tang, Q.; Kurnianggoro, L.; Jo, K.-H. Traffic sign classification with dataset augmentation and convolutional neural network. In Proceedings of the Ninth International Conference on Graphic and Image Processing (ICGIP 2017), Qingdao, China, 14–16 October 2017; p. 106152W.
15. Zhou, B.; Buyya, R. Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 13. [[CrossRef](#)]
16. Lauzon, J.F. Sensor Fusion and Deep Learning for Indoor Agent Localization. Master's Thesis, Rochester Institute of Technology, Rochester, NY, USA, June 2017.
17. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F.E. A survey of deep neural network architectures and their applications. *Neurocomputing* **2017**, *234*, 11–26. [[CrossRef](#)]