Sensors and Materials, Vol. 34, No. 8 (2022) 3047–3061
MYU Tokyo

S & M 3022

Indoor Fingerprinting Positioning System Using Deep Learning with Data Augmentation

Luomeng Liu,[1] Qianyue Zhao,[2] Shoma Miki,[1] Jumpei Tokunaga,[1] and Hiroyuki Ebara[2*]

[1]Graduate School of Science and Engineering, Kansai University,
3-3-35 Yamate-cho, Suita-shi, Osaka 564-8680 Japan
[2]Faculty of Engineering Science, Kansai University, 3-3-35 Yamate-cho, Suita-shi, Osaka 564-8680 Japan

(Received April 1, 2022; accepted May 27, 2022)

Keywords: indoor positioning, fingerprinting, residual network, data augmentation, deep learning

We propose an indoor positioning system based on deep learning and fingerprinting. On the mobile side, we designed an Android application with received signal strength information (RSSI) signal reading, database storage, and real-time online positioning module functions. In addition, we placed a trained neural network model on the built server to achieve real-time positioning using the developed Android application. The deep learning framework of this paper uses a residual network (ResNet) and a data augmentation technique called mean and uniform random numbers in the preparation of the dataset. By using this data augmentation method, we significantly reduced the collection time of the dataset and increased the test accuracy of the neural network from 20.4% before the augmentation to 97.5% after the augmentation.

1. Introduction

In recent years, location-based services (LBSs) for indoor and outdoor environments have become an integral part of our lives, and an increasing number of LBSs have emerged.(1) Outdoors, people can easily access location information through the Global Positioning System (GPS),(2) and LBS providers can provide information about the weather, traffic, hotels, and travel based on the location of the user's mobile device. However, in an indoor environment, GPS cannot provide high accuracy because it is sensitive to occlusion.(3) There are many technologies for indoor positioning, such as Wi-Fi,(4) Radio Frequency Identification (RFID),(5) Bluetooth Low Energy (BLE),(6) and Ultra-wideband (UWB) technologies.(7) The iBeacon technology was introduced by Apple in 2013 with the aim of solving the problem of indoor positioning. iBeacon encapsulates the upper layer protocol on the basis of BLE. As a result, iBeacon technology has no system limitations, and both iOS and Android terminals can use iBeacon technology.(8) iBeacon is widely used owing to its low power consumption, small size, low price, easy deployment, and high stability compared with other technologies.

Traditional indoor positioning algorithms are mainly based on trilateral or triangular positioning algorithms, as well as fingerprinting positioning algorithms. However, the

*Corresponding author: e-mail: ebara@kansai-u.ac.jp
https://doi.org/10.18494/SAM3912

ISSN 0914-4935
https://myukk.org/

performance of geometric positioning algorithms is affected by non-line-of-sight (NLOS) propagation, multipath propagation effects, and shadow fading,[9] and the positioning performance of such methods degrades in indoor environments. Fingerprinting works by saving the Received Signal Strength Information (RSSI) signals of the beacon at each reference point (RP) as a fingerprint in a fingerprint library in the offline phase. Then, in the positioning phase, the position coordinates are predicted and calculated by comparing the collected RSSI with the information in the fingerprint library, resulting in high positioning accuracy.[10] Compared with other positioning algorithms, the fingerprinting method has many advantages in terms of economy and practicality in system deployment, but its disadvantages include the noise signal in the fingerprint database affecting the positioning accuracy, and it is more cumbersome, requiring the establishment of a database with a huge amount of data in the data collection phase.

The use of machine learning techniques can overcome the noise and uncertainty in the fingerprint positioning process.[11] Although traditional machine learning techniques work well in approximating simpler input-output functions, computationally intensive deep learning models are able to handle more complex input-output mappings and provide superior accuracy.[12]

In this paper, we aim to use deep learning for indoor positioning to identify the coordinates of mobile devices with high accuracy. We conduct experiments using iBeacon BLE transmitters. Multiple iBeacon transmitters are installed in the laboratory, and the coordinates of the mobile device are calculated using the RSSI values received from them. Deep learning is then used to improve the positioning accuracy. We use ResNet as the neural network model for this study. ResNet is a 152-layer model reported by Microsoft in 2015.[13] The gradient disappearance problem is solved by introducing a mechanism called Shortcut Connection, which adds the input from the front layer directly to the back layer. At the same time, to provide the huge data volume of the fingerprint database and the huge amount of data needed for neural network training, we use and improve a data augmentation method called mean and uniform random numbers (MURn).[14]

The main contributions of this research are as follows: this paper constructs a system for indoor localization by using the fingerprinting approach and ResNet as the deep neural network. This system greatly reduces the data collection time as well as the construction time for building a local fingerprint database by using an improved MURn method. The superiority of the method proposed in this study is verified by conducting a comparison of experimental results by putting unaugmented data as well as augmented data into the produced neural network in the experiments.

The rest of the paper is structured as follows. In Sect. 2, two indoor positioning methods based on machine learning used in related works are presented. In Sect. 3, the framework of the indoor fingerprinting technique used is introduced. In Sect. 4, we describe the deep learning and data augmentation techniques we use and how we improve them. In Sect. 5, the proposed indoor positioning system is experimentally evaluated, and the experimental results are presented. Finally, the paper is summarized in Sect. 6.

2. Related Works

Over the past decade, most research on indoor positioning has emphasized the use of machine learning. Many researchers have utilized machine learning models, such as k-nearest neighbors (k-NN), decision trees (DT), naive Bayes (NB) classifier, support vector machine (SVM), and random forest (RF).[15,16]

Cong and Meng proposed an indoor positioning algorithm based on NB and Wi-Fi fingerprint recognition.[17] In their experiment, a router was selected as the generator of Wi-Fi signals and the RSSI fingerprint of the signal was collected to form a fingerprint library. The data were trained using a plain Bayesian model and the location was calculated using a server to achieve the fast positioning of smart terminals. The experimental results showed that the system and algorithm performed well, and the positioning accuracy was higher than 80%.

Hsieh *et al.* presented a study of indoor positioning application using BLE-based devices combining Kalman filtering and machine learning.[18] They improved the RSSI stability of Bluetooth transmitters by using a Kalman filtering algorithm and used iBeacon and Android smartphones as experimental devices to test and compare the K-NN, SVM, and RF algorithms. The experimental results showed that the optimal signal collection density for indoor positioning was about 1 m and that the accuracy reached more than 85%.

Recently, some researchers have introduced and applied deep learning techniques to the field of fingerprint localization. Chen *et al.* proposed a new method to train a dilated convolutional neural network (D-CNN) model using images formed from received signal strength (RSS) and to train a SVR model using the error of the D-CNN prediction results.[19] Jondhale and Deshpande proposed a real-time target tracking framework that applies generalized regression neural nets (GRNN) to indoor localization.[20] Combining smartphone and fingerprinting methods to determine location is more accurate and reliable, but the amount of work required to build the required database is huge, especially for large buildings.[21]

On the basis of the above-mentioned research in related literature, an indoor fingerprint localization system using deep learning and data augmentation is proposed in this paper. To further improve the localization accuracy of the RSSI fingerprint localization method, this study uses the deep learning network ResNet. At the same time, to solve the problem that a large amount of data needs to be collected and it takes a lot of time to build the fingerprint database, we propose an improved MURn data augmentation method.

3. Proposed Indoor Positioning by Fingerprinting Method

In this section, the structure of the fingerprinting indoor positioning system used in this paper is presented, as shown in Fig.1. The fingerprinting technique is divided into two phases: an offline fingerprint calibration phase and an online position estimation phase.[22]

In the offline phase, we collect the RSSI values from all beacons at each RP and store them in the database. Some of the data are augmented, and the unaugmented and augmented data are divided into different datasets and put into the CNN model for training. In the online phase, we obtain the RSSI values from random RPs, process the data, and send them to the trained model

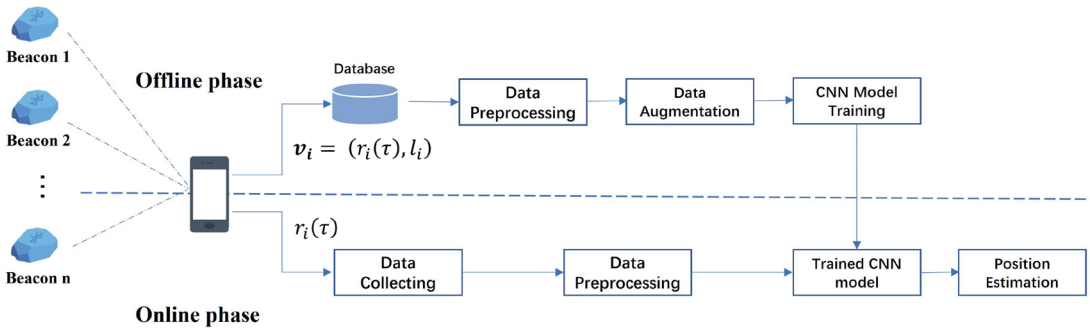

Fig. 1. (Color online) Structure of fingerprinting positioning system with online and offline phases.

deployed on the server. Finally, the server sends the location information processed by the model to the user's mobile terminal.

3.1 Offline phase

The main work of the offline phase is to collect RSSI values from each RP and construct a location fingerprint database. For example, the mobile terminal collects a set of RSSI values from M beacons and samples them t times at N RPs:

$$r_i(\tau) = \left[r_{i,1}(\tau), r_{i,2}(\tau), \ldots, r_{i,M}(\tau) \right]^T, \quad \tau = 1, 2, \ldots, t, \quad t > 1, \tag{1}$$

where $r_{i,i}(\tau)$ denotes the RSSI value collected at time τ from the j-th beacon and t is the sampling period.

The fingerprint is sent to the server as a vector $\boldsymbol{v}_i = (r_i(\tau), l_i)$, where l_i is the known location. The collected fingerprint data are referred to as radio maps,[23] which are described as

$$\boldsymbol{V} = \left[\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n \right]. \tag{2}$$

The final step in the offline phase is to send the collected radio maps to the server for training and to store the results of the training for use in the online phase.

3.2 Online phase

In the online phase, the user uses the mobile terminal to obtain location information and sends this information to the server. In this process, the terminal collects RSSI from all beacons at this unknown location and sends it to the server. The server receives this information and converts it into the fingerprint form. Next, the server compares the data in the fingerprint database with a trained neural network model. Finally, the server returns the estimated location to the mobile terminal.

4. Proposed Deep Learning Model and Improved Data Augmentation Technique

To solve the problems of the traditional methods mentioned in Sect. 1, we propose an indoor positioning algorithm using deep learning with data augmentation.

4.1 Deep learning

Deep learning is a method for learning deep neural networks with a multilayer structure.[24] A neural network is a computational model with the ability to simulate functions in the cranial nervous system by combining artificial neurons that simulate the nerves of a living organism. The artificial neurons that are part of the neural network are represented by the schematic diagram in Fig. 2 and have weights $w = (w_1,\ldots,w_n)$ and biases b as learning parameters. Using them, the following linear combination of the input $x = (x_1,\ldots,x_n)$ multiplied by the activation function φ is output:

$$y = \varphi w^T + b\,. \tag{3}$$

4.2 Residual network

In this paper, we use a 152-layer ResNet model. In deep learning, as a rule of thumb, deeper layers provide better performance. However, by making the layers deeper, certain layers cannot be updated during learning. Therefore, in ResNet, we prepare a route that connects the layers in series, as shown in Fig. 3, connecting the input to the output of each layer and flowing the input values as they are. This route allows the trained layer to understand how it differs from the input and to be updated when necessary. Alternatively, the route that skips layers is called Shortcut Connection and requires more than two layers to be skipped to learn.

It has been shown that high accuracy can be achieved in image classification using ResNet. Therefore, even in the class classification in our method, we aim to improve the accuracy of the solution by using ResNet.

In this experiment, we use multiclass classification to determine where we are in the laboratory. Our method is to make the number of elements, K, of the output layer the same as the number of classes to be classified, and when the training data is in the k th class, the teacher data is [0,..., 0, 1, 0..., 0], i.e., only the k th element is 1. Each element of the output y of the neural network model is associated with the probability of appearance of each class, each output from the output element is in the range of 0 to 1, and the sum is normalized by the softmax method so that it is 1.

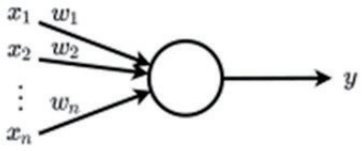

Fig. 2. Diagram of artificial neuron.

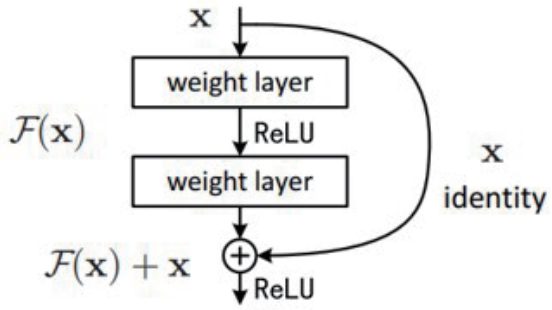

Fig. 3. Features of ResNet.

4.3 Model learning

Figure 4 shows the structure of the ResBlock used in this experiment. In the dense layer, normalization is performed for each mini-batch by batch normalization. In addition, all negative values are removed by the rectified linear unit (ReLU) activation function.

Next, Fig. 5 shows the ResNet model used and Table 1 shows the parameters during training. The network consists of 12 processing layers. The number at the top of each processing layer and block in Fig. 5 indicates the number of output units. In each layer, batch normalization and activation functions are applied to the outputs. The activation function uses softmax for the output layer and ReLU for the other layers. The learning rate is 10^{-4}.

The cross-entropy of Eq. (4), defined by the one-hot expression p_i representing the actual locus and the predicted probability q_i of the locus, is used as the loss function. In addition, the loss function is used to perform gradient descent to update the weights of the model. One epoch means that all the data in the dataset are used only once to update the weights, and in this experiment, the number of epochs for learning is set to 4000.

$$H(p,q) = -\sum_i p_i \log q_i \tag{4}$$

4.4 Data augmentation

This experiment uses the MURn data augmentation method. The original idea of this method is to add information to the reference dataset by using mean and uniform random numbers. The total number of RSSIs in the reference dataset is increased in such a way that the overall RSSI remains between the minimum and average of the original RSSI values. For each RP, the average of the visible APs is calculated. After calculating the average, the range for generating uniform random numbers is calculated by subtracting the current RSSI value from the average.

We improve this method by adding the most frequent occurrence S for data collected from a particular beacon at the same location. We add a weight to S in the generation of the uniform and

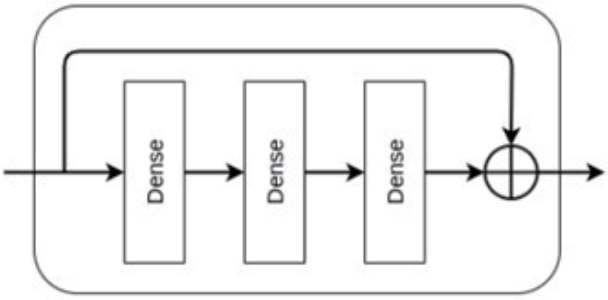

Fig. 4. Construction of the ResBlock used.

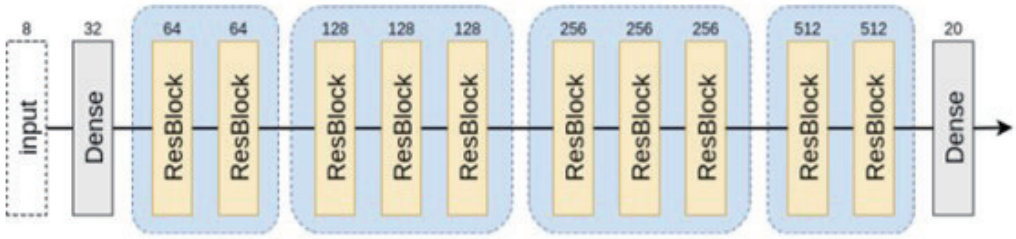

Fig. 5. (Color online) ResNet model.

Table 1
Parameters during model training.

Item		Parameters
Model configuration	Number of mini-batches	8
Weight setting	Momentum weights for batch	0.99
	regularization, bias regularization	L2 (10^{-6})
	Weight initialization	HeNormal
	Bias initialization	Zero
Optimization method	Algorithm	Nesterov accelerated gradient
	Learning coefficient	10^{-4}
	Learning coefficient decay	0.0
	Momentum	0.9

random numbers, increasing the relevance of the generated data and the verification accuracy. Figure 6 illustrates the pseudocode of this augmentation scheme.

In this study, we first stored the RSSI data received at all RPs in the developed Android application and then extract the data to a computer. By our own coded python program, for example, the augmentation number N is defined as 400 for this experiment, then the data from each RP is augmented to 400. This is performed by first finding the average of all RSSI values, M, for any beacon of a certain RP and finding the mode S. While increasing the weight of S, the data are generated randomly for a range of M and all RSSI values. For example, if we make the number of generators $N = 40$, the current RSSI value equal to 75, and the average value of the beacon in the RP equal to 61, then the potential range of uniform random numbers will be 15 (61 to 75 inclusive), which means that the uniform random numbers will be generated 40 times at random for 15 numbers in the range 61–75.

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

Fig. 6. Pseudocode for data augmentation with MURn.

5. Evaluation

In this section, we compare the raw and augmented data.

5.1 Experimental environment

In the experiment, an Aplix BLE transmitter is used for indoor positioning. The mobile terminal that receives RSSI signals uses a TCL T770B smartphone. The specifications of these devices are given in Tables 2 and 3.

Next, Table 4 shows the configuration of the computer used in this experiment. Python 3.8 is used as the programming language and Keras, a TensorFlow backend, is used as the machine learning library.

In this experiment, eight BLE transmitters are installed in the ceiling of a laboratory of 10.29 m width, 8.23 m length, and 2.90 m height. Figure 7 shows the 23 RPs and the locations of the BLE transmitters. During the data collection phase, the experimentalists collected RSSI data at 23 locations approximately 1.5 m above the ground with handheld smartphones.

In addition, Fig. 8 shows the laboratory environment as well as the ceiling. The floor of this laboratory is carpet and the ceiling, south wall and east wall are made of gypsum board. The west side includes windows as well as reinforced concrete, and the north wall is completely reinforced concrete.

Table 2
Aplix BLE transmitter parameters.

Item	Parameters
Device name	Aplix MyBeacon® MB00Ac-DR1
Power supply	Two dry batteries
Bluetooth specifications	Bluetooth Ver.4.1
Frequency band	2.4 GHz
Output power	−12 dBm
Measured power	−70 dBm
Beacon transmission interval	100 ms
Transmission data	Minor value, Major value, Proximity UUID

Table 3
Specifications of mobile terminal used in the experiment.

Item	Parameters
Device name	TCL T770B
OS version	Android 10
CPU	Snapdragon 665
Memory	6 GB
Storage	128 GB
Bluetooth specifications	Bluetooth v5.0 Low Energy
Frequency band	5 GHz
Beacon transmission interval	1000 ms

Table 4
Specifications of computer used in the experiment.

Software and hardware configuration	Configuration
CPU	Intel® Core™ i7-11700
Memory	32 GB
Graphics card	NVIDIA Geforce RTX 3090
CUDA	CUDA 11.2
CuDNN	CuDNN 8.1
Python	Python 3.8
TensorFlow	TensorFlow 2.5.0

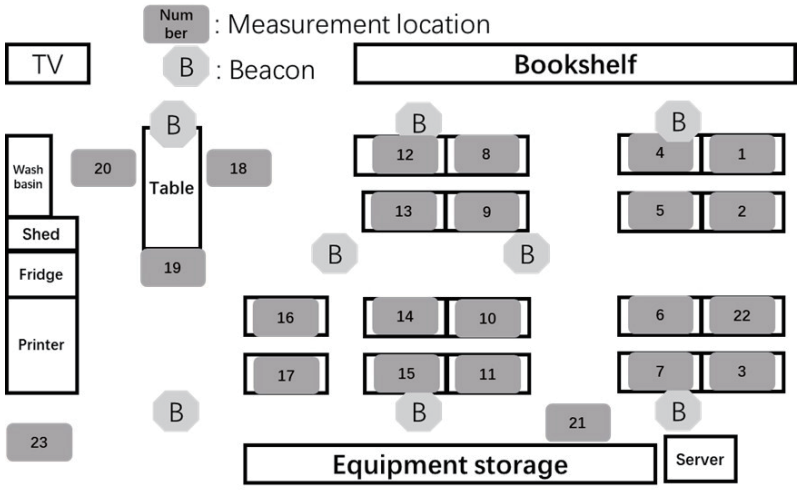

Fig. 7. Locations of BLE transmitters set in the laboratory.

Fig. 8. (Color online) Actual laboratory environment.

As shown in Fig. 9, a self-developed Android app for RSSI data collection is used in the experiment. Eight beacons are collected every 10 s at 23 locations in the calibration, and a total of 113 data are collected, 13 of which are used as teacher data, and the remaining 100 are used as test data. We used the MURn data augmentation method described in Sect. 4.4 to expand the 13 data collected from each RP to 400. The data are shown in Table 5.

The two data models are trained separately using the deep learning indoor positioning method shown in Sect. 4.2 for the raw and augmented data. Since in the preparatory experiments, the previous researchers experimented by using 400 training data as well as 100 test data at each RP and conducted the experiments. Therefore, this paper also augments the training data of the raw data to 400 to conduct the experiments. Then, the localization accuracy of each point is verified.

5.2 Experimental results

Figures 10 and 11 show that the accuracy of both the training and test data improved as the number of epochs increased. Thus, from the figures, it can be seen that the training data are correctly learned.

Table 6 also shows that the average accuracies are 20.4% before augmentation and 97.5% after augmentation, an increase of 77.1%. Before data augmentation, the accuracy of many positions is very poor. However, the accuracy of almost all positions becomes excellent after data augmentation, thus showing that the data augmentation is effective. The accuracy also varies with the location.

In addition, we can see from Table 6 that the augmented data still perform poorly in some locations, such as positions 7 and 15, where the accuracy rates are 93 and 90%, respectively. The reason for this result is mainly due to the fact that these positions have in common the proximity to walls and the presence of more obstacles. Besides, by looking at the model's misestimation results, it is found that most of the misestimations are estimated in the neighboring positions, which may also be attributed to the proximity of these positions to the proximity when dividing the RPs. On the contrary, in some remote positions, such as position 23, these positions are likely more correct owing to less occlusion and the absence of neighboring positions around them.

Fig. 9. (Color online) Developed scanning signal interface for Android application.

Table 5
Datasets of the experiment.

Dataset	Training data	Test data
Raw data	13	100
Augmented data	400	100

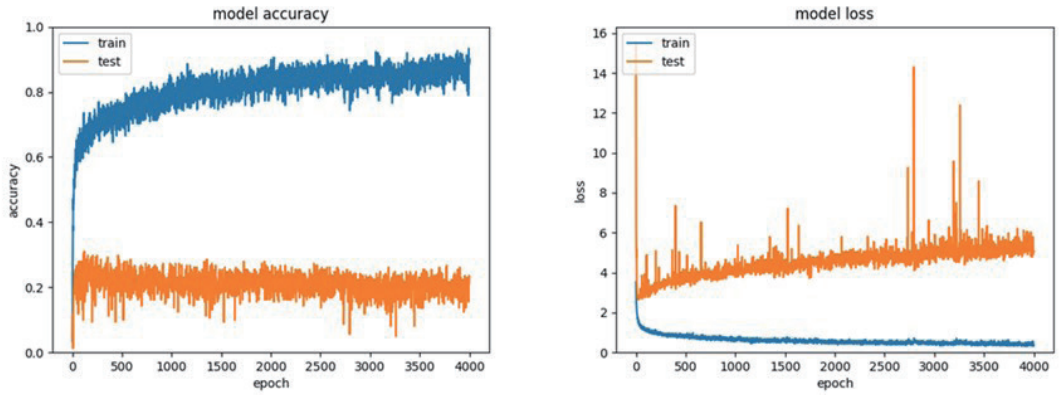

Fig. 10. (Color online) Discrimination accuracy and loss function during training of the model before data augmentation.

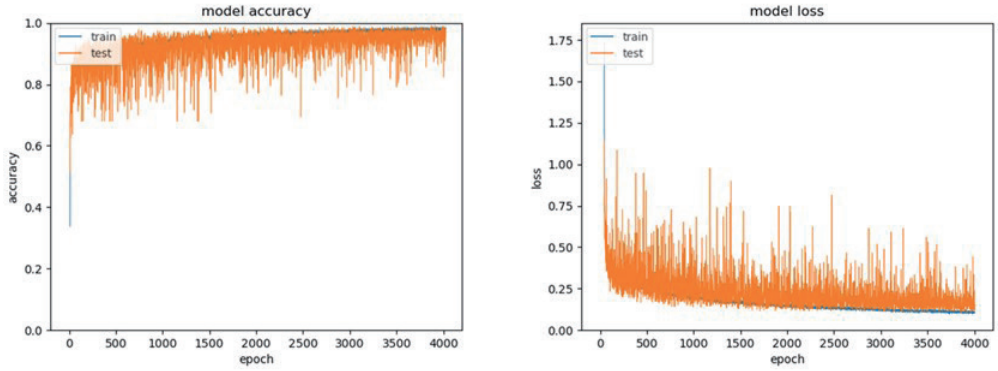

Fig. 11. (Color online) Discrimination accuracy and loss function during training of the model after data augmentation.

Table 6
Localization accuracy of each position before and after data augmentation.

Position	Precision before augmentation (%)	Precision after augmentation (%)
1	40	98
2	16	100
3	13	98
4	37	100
5	11	97
6	26	100
7	18	93
8	10	100
9	26	99
10	6	96
11	20	99
12	16	95
13	30	100
14	18	99
15	7	90
16	24	96
17	29	100
18	14	94
19	22	100
20	18	94
21	17	96
22	30	100
23	22	98
Average	20.4	97.5

5.3 Limitations of this study

Since this study was conducted only in this laboratory to collect data as well as to conduct experiments, the conclusions drawn were also limited to the environment in this research laboratory. In future work, we hope to conduct experiments on indoor positioning in more

indoor environments with different rooms. In this way, we can study the effects on radio signals in different material environments. Also, during the collection phase of both training and test data, the experimentalists performed the collection without human presence. Therefore, additional work is required to support the data in the presence of additional distractors or people.

6. Conclusions

We proposed an indoor positioning system using fingerprinting techniques and deep learning. We also utilized data augmentation techniques to significantly reduce the time to collect data and build a database. We implemented and evaluated the model by testing the accuracies before and after data augmentation. It was shown that the localization accuracy after data augmentation improved by 77.1% compared with that for the raw data set. In addition, by comparing the accuracies of indoor positioning methods based on machine learning mentioned in related works, we found that the accuracy of the machine learning models tested is about 80–90% while the accuracy of the deep learning proposed in this paper exceeded 97%.

Two future topics to be studied are as follows. The first is the integration of servers and smartphones. In this paper, real-time location measurement was achieved by creating an application. However, there were problems with server maintenance and communication delays due to the communication between the smartphone and the server. Therefore, we want to port the deep learning model to a smartphone. Second, it is necessary to integrate the application and put it into practical use with applications that detect people entering and leaving the room, and to consider better user interfaces and designs.

Acknowledgments

This research was partly supported by the Information and Communication Technology Research Group of ORDIST Kansai University.

References

1 K. W. Kolodziej and J. Hjelm: Local positioning systems: LBS applications and services (CRC press, Boca Raton, 2017) 1st ed., Chap. 2. https://doi.org/10.1201/9781420005004
2 N. Bulusu, J. Heidemann, and D. Estrin: IEEE Pers. Commun. **7** (2000) 5. https://doi.org/10.1109/98.878533
3 M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk: Indoor positioning using GPS revisited (Springer, Heidelberg, 2010) 1st ed., pp. 38–56. https://doi.org/10.1007/978-3-642-12654-3_3
4 C. Yang and H. R. Shao: IEEE Commun. Mag. **53** (2015) 3. https://doi.org/10.1109/mcom.2015.7060497
5 S. S. Saab and Z. S. Nakad: IEEE Trans. Ind. Electron. **58** (2010) 5. https://doi.org/10.1109/tie.2010.2055774
6 Z. Jianyong, L. Haiyong, C. Zili, and L. Zhaohui: Proc. 2014 Int. Conf. Indoor Positioning and Indoor Navigation (IEEE, 2014) 526–533. http://dx.doi.org/10.1109/IPIN.2014.7275525
7 T. Gigl, G. J. Janssen, V. Dizdarevic, K. Witrisal, and Z. Irahhauten: Proc. 2007 4th Workshop on Positioning, Navigation and Communication (IEEE, 2007) 97–101. https://doi.org/10.1109/WPNC.2007.353618
8 Springer Link: https://link.springer.com/article/10.1057/dddmp.2014.7#Sec8 (accessed May 2022).
9 G. Ding, J. Zhang, l. Zhang, and Z. Tan: Proc. 2013 5th IEEE Int. Symp. Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications (IEEE, 2013) 160–164. https://doi.org/10.1109/MAPE.2013.6689973
10 S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang: ISPRS Int. J. Geo-Inf. **6** (2017) 5. https://doi.org/10.3390/ijgi6050135

11 P. Roy and C. Chowdhury: J. Intell. Rob. Syst. **101** (2021) 3. https://doi.org/10.1007/s10846-021-01327-z
12 K. Y. Seok and J. H. Lee: Proc. 2020 Int. Conf. Artificial Intelligence in Information and Communication (IEEE, 2020) 312–314. https://doi.org/10.1109/ICAIIC48513.2020.9065054
13 K. He, X. Zhang, S. Ren, and J. Sun: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2016) 770–778. https://doi.org/10.48550/arXiv.1512.03385
14 R. S. Sinha, S. M. Lee, M. Rim, and S. H. Hwang: Electronics **8** (2019) 5. https://doi.org/10.3390/electronics8050554
15 A. Chriki, H. Touati, and H. Snoussi: Proc. 2017 13th Int. Wireless Communications and Mobile Computing Conf. (IEEE, 2017) 1144–1149. https://doi.org/10.1109/IWCMC.2017.7986446
16 M. Dziubany, R. Machhamer, H. Laux, A. Schmeink, K. U. Gollmer, G. Burger, and G. Dartmann: Proc. 2018 26th European Signal Processing Conf. (IEEE, 2018) 2050–2054. https://doi.org/10.23919/EUSIPCO.2018.8553155
17 C. Cong and X. Meng: Proc. Int. Conf. Signal And Information Processing, Networking And Computers, (Springer, Singapore, 2017) 238–246. https://doi.org/10.1007/978-981-10-7521-6_29
18 J. Hsieh, C. Fan, J. Liao, J. Hsu, and H. Chen: EasyChair Preprint **1198** (2019) 1. https://easychair.org/publications/preprint/VhvV
19 H. Chen, B. Wang, Y. Pei, and L. Zhang: 2020 Chinese Automation Congr. (CAC) (IEEE, 2020) 165–170. https://doi.org/10.1109/CAC51589.2020.9327326
20 S. R. Jondhale and R. S. Deshpande: Ad Hoc Networks **84** (2019) 19. https://doi.org/10.1016/j.adhoc.2018.09.017
21 F. Zafari, I. Papapanagiotou, and K. Christidis: IEEE Internet Things J. **3** (2016) 96. https://doi.org/10.1109/JIOT.2015.2442956
22 G. Félix, M. Siller, and E. N. Alvarez: Proc. 2016 8th Int. Conf. Ubiquitous and Future Networks (IEEE, 2016) 1006–1011. https://doi.org/10.1109/ICUFN.2016.7536949
23 S. Sorour, Y. Lostanlen, and S. Valaee: Proc. 2012 IEEE Global Communications Conf. (IEEE, 2012) 303–308. https://doi.org/10.1109/GLOCOM.2012.6503130
24 J. Schmidhuber: Neural Networks **61** (2015) 85. https://doi.org/10.1016/j.neunet.2014.09.003

About the Authors

Luomeng Liu received his B.S. degree from North China University of Water Resources and Electric Power, China, in 2020. He is currently pursuing his master's degree under the supervision of Hiroyuki Ebara at Graduate School of Science and Engineering, Kansai University, Japan. His research interests are in deep learning and indoor positioning. (k734257@kansai-u.ac.jp)

Qianyue Zhao received his B.S. degree from Faculty of Engineering Science, Kansai University, Japan, in 2022. His research interests are in Android development and indoor positioning. (zhaoqianyue1999@gmail.com)

Shoma Miki received his B.S. degree from Faculty of Systems Science and Engineering, Kansai University, Japan, in 2017. He completed his master's program in systems science and engineering at Graduate School of Science and Engineering, Kansai University, in 2019. He is currently enrolled in the doctoral program at the Department of Interdisciplinary Graduate School of Science and Engineering. He is engaged in research on solving combinatorial optimization problems using deep learning. (k154911@kansai-u.ac.jp)

Jumpei Tokunaga received his B.S. degree from Faculty of Systems Science and Engineering, Kansai University, Japan, in 2017. He completed his master's program in Systems Science and Engineering, Graduate School of Science and Engineering, Kansai University, in 2019. He is currently enrolled in the doctoral program at the Department of Graduate School of Science and Engineering, and he is a JSPS Research Fellow. He is engaged in research on delay-tolerant networks. (tokujun@fw.ipsj.or.jp)

Hiroyuki Ebara received his B.S., M.S., and Ph.D. degrees in communications engineering from Osaka University, Osaka, Japan, in 1982, 1984, and 1987, respectively. In 1987, he became an assistant professor of Osaka University. Since 1994, he has been with Kansai University, where he is currently a professor. His main research interests are in computational geometry, combinatorial optimization, and parallel computing. He is a member of IEEE, ACM, SIAM, IEICE, IPSJ, and the OR Society of Japan. (ebara@kansai-u.ac.jp)