Synthesis of a Small Fingerprint Database through a Deep Generative Model for Indoor Localisation

Dwi Joko Suroso, Panarat Cherntanomwong*, Pitikhate Sooraksa
School of Engineering, King Mongkut's Institute of Technology Ladkrabang,
1 Chalong Krung, 1 Alley, Lat Krabang, 10520 Bangkok, Thailand
*64601003@kmitl.ac.th, *panarat.ch@kmitl.ac.th, pitikhate.so@kmitl.ac.th*

***Abstract*—In deep learning (DL), the deep generative model is helpful for data augmentation objectives to tackle the lack of datasets that have a significant impact on learning performance. Data augmentation or synthesis is expected to solve the issue in a small/sparse database. The problem of databasing also exists in the fingerprint-based indoor localisation system. The dense offline fingerprint database must be constructed with the accuracy requirement. However, this will affect the high cost, massive laborious work, and increase the complexity of the system. Therefore, this paper proposes to address these issues by generating synthetic data via a deep generative model. The generative adversarial network (GAN) is selected to generate the synthetic fingerprint database for indoor localisation. Our database consideration consists of power-based parameters, i.e., the received signal strength indicator (RSSI) from Wi-Fi devices obtained from the actual measurement campaign. Some of the literature mainly discusses how GAN works in a vast and complex dataset. Here, we consider applying GAN in a relatively small dataset and for a simple setup. Our results show that by only using the 20 % fraction of actual RSSI data combined with the synthetic RSSI, the accuracy validation performance is slightly higher than when using all actual data usage. Moreover, in only 60 % of actual data usage and in combination with 625 samples of synthetic data, the accuracy performance is improved to 0.73 (1.37 times higher than the use of all actual data, 0.53). Thus, this result proves that the challenges of offline fingerprint databases can be alleviated by data synthesis through GAN by using only a small dataset.**

***Index Terms*—Deep generative model; GANs; Fingerprint technique; Indoor localisation.**

I. INTRODUCTION

Indoor localisation is still an active research topic today. The most widely used indoor localisation techniques are distance-based and distance-free. Distance-based refers to the technique when the parameter needs to be converted to a distance and use a specific method to find the location, e.g., trilateration, angle-based, and time-based methods. However, this technique is reliable only when achieving good quality from the parameter-distance conversion result. The complex indoor environment yields a multipath effect in radio propagation; the non-line-of-sight condition often degrades the quality of the parameter [1], [2].

On the other hand, the distance-free method, i.e., the fingerprint method, does not need the parameter-distance conversion. It has benefit on using spatial information directly which consist of the location of the fingerprint points and their corresponding localisation parameter values. The multipath effect in the indoor environment does not affect the localisation process, because all these effects are considered in the recorded database [3].

The fingerprint technique has been a central and exciting topic in indoor localisation. Despite its accuracy, performance, and simplicity of parameters, the fingerprint technique has a significant drawback in its offline database construction. This database needs to be densely designed to ensure the reliability of accuracy performance. However, much laborious and cost efforts are needed to build a very dense database. Instead of collecting the dense database in the offline phase, this paper tackles this problem. The small number or scarce location of fingerprints is emphasized. Then, data synthesis is applied to make the database denser [4].

Previous publications offered some approaches to the fingerprint issue in offline fingerprint construction, including the crowd-sourcing data; unfortunately, active participants are needed in the data collection process. The basic idea of our proposal is how we can generate synthesis/fake database to compensate for the low density of the fingerprint database. We apply deep generative model-based data synthesis as deep learning (DL) implementations have been widely proven to be helpful in a wide area of research [5], [6].

The deep generative model comprises two main types: implicit and explicit density interpretation [7]. One type of explicit density is the generative adversarial network (GAN) [8]. Explicit density is a generating function that draws a sample from the actual input distribution. Some preliminary works on GAN for indoor localisation are in [9]–[11]. Especially for the work in [12], we cite this publication as our primary reference. Other proposals implemented variational autoencoders (VAE) that utilised received signal strength indicator (RSSI) and channel state information (CSI) [13]–[15].

Furthermore, the modified VAE and GAN to semi-supervised method also proved promising for indoor

Manuscript received 22 February, 2022; accepted 29 November, 2022.
This research was supported by King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand.

localisation [15]–[19]. Semi-supervised, on the other hand, needs a source of unlabelled data. However, there are many rooms to improve and implement the deep generative model, especially for the small and scarce fingerprint database.

We propose investigating how using a small dataset, then compensating it with the synthetic data, will yield similar or better performance than using all available data. We hypothesise that by applying database synthesis, the database grid will increase, resulting in a denser database. We would like to see that these denser data will improve the localisation performance of the fingerprint technique. Hopefully, by inspecting how the synthetic data are helpful, we can tackle the sparse/small dataset issue in fingerprint-based indoor localisation.

We organise this paper as follows. The introduction is given in Section I, including key references to preliminary works. In Section II, we present materials and methods. We then discuss the implementation of GAN, the results, and the discussion in Section III. Finally, Section IV is the conclusions and future work.

II. MATERIALS AND METHODS

This section will introduce the fingerprint technique, GAN, and our measurement campaign system and setup, system model, and data splitting.

A. Fingerprint Technique

The fingerprint technique involves a two-stage process to estimate the target location. The offline stage is the essential stage where the fingerprint database is obtained based on the parameter and fingerprint location. In the online stage, the new parameter values exhibited by the target or object will be compared to those in the offline database [20].

Figure 1 shows the overall process of the fingerprint technique. For example, when the parameter used for localisation is RSSI values exhibited from 3 (three) reference nodes (RNs), then the fingerprint database is a collection of RSSI from three RNs in a specific location as the fingerprint location. The fingerprint technique applies spatial information to obtain the target location. In the online phase, the pattern matching algorithm is employed. This algorithm has the task of finding the similarity between the target and the database. The algorithm used can be a simple distance metric or an advanced method that employs machine learning (ML) algorithms.

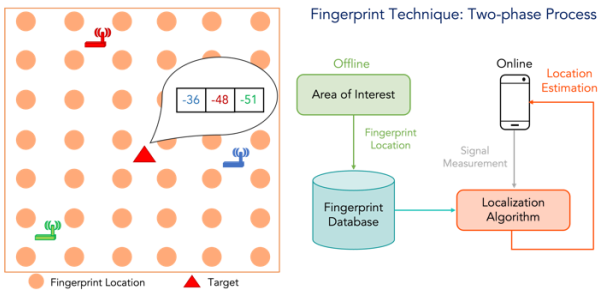


Fig. 1. Illustration of the fingerprint technique.

B. Generative Adversarial Network (GAN)

The basic idea of GAN is to generate new data through the adversarial process of two networks. GAN was first introduced by the authors in [8] and consists of two networks, e.g., generative as G and discriminative as D. These two networks will duel in a minimax two-player game. The training of G aims to maximise the probability that D is making a mistake in discrimination. Finally, in an arbitrary function space, G can recover the training/input data distribution while D is equal to 0.5 everywhere or achieve the Nash equilibrium. In this state, the quality of the data generated by G has been like the input, so that D cannot distinguish it anymore.

Let us assume that both networks are in the form of multilayer perceptron (MLP) networks. MLP is a fully connected class of feedforward artificial neural networks (ANNs) so that the entire GAN can be trained with backpropagation. Compared to other methods, GAN does not need the Markov chain or unrolled approximate Bayesian inference either in training or in generating the sample data. The GAN illustration is depicted in Fig. 2.

As we assumed that both networks are MLP, we can describe the GAN process as follows:

1. Defining a prior distribution of the input noise, such as $p_z(z)$ to learn the distribution of the network G, p_g over the input data, x;
2. Mapping to the data space, $G(z,\theta_g)$, is a differentiable function by MLP with parameter θ_g;
3. The second MLP, $D(x,\theta_d)$ has an output single scalar. $D(x)$ represents the probability that x comes from the actual data rather than from the G, p_g;
4. The D is trained to minimise $\log\left(1-D\left(G\left(z\right)\right)\right)$ or value function of $V(D, G)$ as a two-player minimax game as

$$min_G max_D V(D,G) = \mathrm{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \\ +\mathrm{E}_{z\sim p_z(z)}\left[1-\log D\left(G\left(z\right)\right)\right]. \quad (1)$$

$V(D,G)$ in practise is approached by an iterative, numerical procedure. In this manner, we alternate between the k steps of optimising D and one step optimising G, resulting in D is being maintained near its optimal solution if G is slowly changing. In (1), we can observe that instead of minimising $1-\log D\left(G\left(z\right)\right)$, we can train to maximise $\log D\left(G\left(z\right)\right)$.

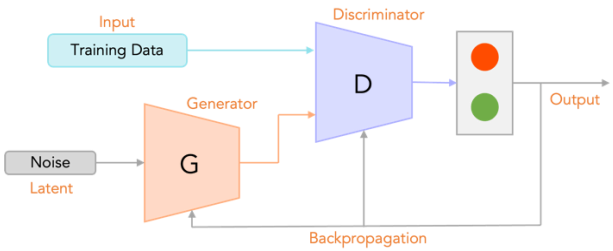


Fig. 2. Illustration of GAN.

C. Measurement Campaign

We consider the low-cost and straightforward measurement campaign system and setup. The primary device we used was the Wi-Fi-based ESP32. Table I shows the details of the devices, tools, and other properties of the

measurement campaign.

We consider the flow of the measurement system by using the illustration in Fig. 3. First, we send the personal computer (PC) command to the sink node for RSSI broadcast via the server node. Then, the sink node will collect the RSSI from a total of 8 reference nodes. The last step is to send RSSI values back to the server node and to acquire the RSSI values and store them on the PC.

TABLE I. MEASUREMENT DEVICES AND TOOLS.

Name	Device	Specification	Use
Reference, server, sink (target) node	ESP32 Toolkit	IEEE 802.15.11 standard, Memory 520 kB	Transceiver for RSSI values
Software	Arduino IDE, Jupyter Notebook	1.8.5 version (for windows), Python 3.7, TensorFlow, Keras	ESP32 basic program, algorithm implementation
Personal Computer (PC) 1	CPU (AMD Ryzen 5 3600) GPU (NVIDIA GeForce GTX 1660)	RAM 16 GB	For collecting the data, program ESP32
Personal Computer (PC) 2	CPU (AMD Ryzen 9 3900X), Windows 10 (64-bits), GPU (NVIDIA GeForce RTX 2080)	12-Core Processor, 3.79 GHz. RAM 32 GB, VRAM 8 GB and shared memory 16 GB (total 24 GB)	Algorithm implementation

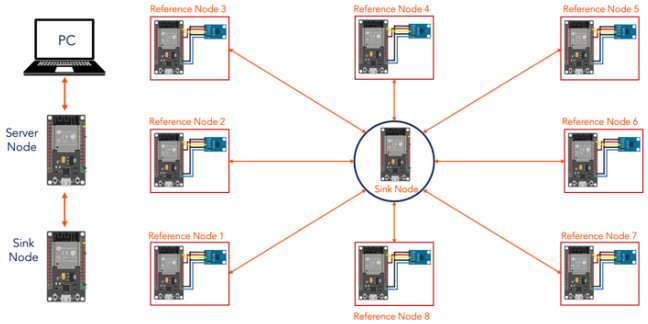


Fig. 3. Illustration of the measurement system.

We conducted the measurement setup in a typical classroom with a total area of 15.14 m × 10 m. The area of interest that we built is 5 m × 5 m divided into 25 fingerprint locations with a grid of 1 m × 1 m. The measurement layout and actual can be depicted in Figs. 4 and 5, respectively.

The total data obtained from the illustration in Fig. 3 is the 1,250 rows (samples) and 8 column RSSI values data (-dBm). The rows indicate the number of all RSSI values from 25 fingerprint locations (each location has 50 RSSI samples), while the columns indicate the 8 (eight) reference nodes (RNs). Figure 6 shows the structure of the measurement data obtained.

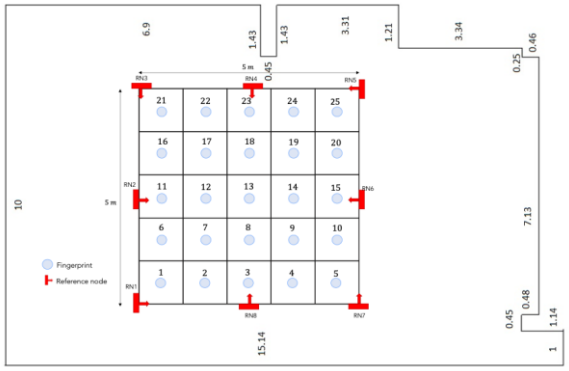


Fig. 4. Layout of measurement setup.

Fig. 5. Actual measurement setup.

Parameter	RSSI	RSSI	RSSI	RSSI	RSSI	RSSI	RSSI	RSSI	Coordinate	
Node ID	1	2	3	4	5	6	7	8	x	y
0	-52	-56	-58	-67	-65	-64	-65	-60	1	1
1	-52	-55	-56	-64	-63	-67	-68	-55	1	1
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1250	-67	-65	-69	-58	-43	-57	-69	-73	5	5

Fig. 6. Structure of the measurement data.

D. System Model

The system model of this approach is based on the work in [12]. We make a difference in terms of class dataset point-of-view due to the small dataset. We refer to "class" as the location of the fingerprints, not a classroom. Let $F(\boldsymbol{x},\theta_C)$ with $\boldsymbol{x}\in\mathbb{R}^{1\times M}$ be a column vector of RSSI values, with M is 8 RNs. θ_C is the parameter of deep learning during training and C represents the class output $\mathbb{R}$ nodes. In vector form, the equation of the system model can be written as

$$\mathcal{L}(\theta_C) = -\sum_{i=1}^{N} \boldsymbol{y}_i \log \hat{\boldsymbol{y}}_i^T. \tag{2}$$

$\mathbf{y}_i \in \mathbb{B}^{1\times C}$, $\mathbb{B}\in\{0,1\}$. $\mathbf{y}_i$ is one encoded, if 1, we will have $[1\ 0\ 0\ \ldots\ 0]_1$, $[0\ 1\ 0\ \ldots\ 0]_2$, … … $[0\ 0\ 0\ \ldots\ 1]_C$ and $\hat{y}_i$ is a real number between 0 and 1 predicted by the deep model. The log function satisfies the hard selection of a single class.

Backpropagation is used to train the system via minimising the log-likelihood cost function. We utilised the Adam optimiser to obtain $F(\mathbf{x},\theta_C)$, such that the maximum value of the outputs of C nodes constitutes the class predicted for a given input vector by the trained system model.

The idea is to observe how the synthetic RSSI added to fraction variation of the actual dataset can enhance the performance compared to only using the actual data. The RSSI dataset, $\boldsymbol{R}_c$, can be described as

$$\boldsymbol{R}_c = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{K1} & r_{K2} & \cdots & r_{KM} \end{bmatrix}, \quad (3)$$

with r_{ij} is the magnitude of the *i*-th observation from the *j*-th reference node. Each column of the matrix constitutes a distribution over the desired class. We define $\mathrm{x} \in \mathbb{R}^{1\times M}$ such that the generator's goal is to map prior noise latent variable $\mathrm{z} \in \mathbb{R}^{1\times L}$ to the distribution of R_c. The process of producing the synthetic data is based on the cost function in (1). Convergence occurs when $D(\mathbf{x},\theta_d) = 0.5,$ meaning that the discriminator can no longer distinguish between actual and synthetic data. After this, the generator is ready to produce synthetic samples for the desired class via the same prior noise distribution $\boldsymbol{z} \sim p_z(z)$.

We consider the classification through MLP used for training by combining actual and synthetic RSSI data. Thus, the full set of RSSI (**FR**) data, consisting actual RSSI (**A**) and synthetic RSSI (**SR**) data for the desired class *C*, can be defined as follows

$$\mathbf{FR}_C = \begin{pmatrix} \mathbf{A}_C \\ \mathbf{SR}_C \end{pmatrix}, \quad (4)$$

where $\mathbf{A}_C \in \mathbb{R}^{K\times M}$, $\mathbf{SR}_C \in \mathbb{R}^{Q\times M}$, $\mathbf{FR}_C \in \mathbb{R}^{(K+Q)\times M}$, and *Q* is the number of added synthetic samples to class *C*. In our case, *Q* is the two-times number of fractions of the actual dataset use.

E. GANs Implementation in Keras

We implemented GAN for our dataset on Python 3.7 using Keras (https://github.com/fchollet/keras) (see Algorithm 1). The GAN parameter is detailed in Table II.

Algorithm 1. GAN implementation: Pseudo-Code.

	Input: RSSI dataset
	Output: RSSI actual + synthesis (full RSSI dataset)
1.	Keras, TensorFlow, and other libraries
2.	Data parameter (25 labels), MLP parameter, GAN Parameter
3.	Set the percentage of actual and data to generate (synthesis)
4.	Import dataset and data pre-processing (data frame, drop)
5.	*MLP classification with actual data*
6.	Generating synthetic data: Generator and Discriminator
7.	GAN training
8.	*MLP classification with actual + synthetic data*
9.	Save results

The standard procedure begins with the importing libraries needed. Then, the parameters are defined, e.g., the number of labels, 25, as we discussed related to each fingerprint locations as a label. The MLP parameter is set, including the number of iterations (epochs) and validation splitting. The GAN parameter includes the number of latent dimensions and epochs for GAN training.

The percentage or fraction of data use is set in the beginning. Dataset is imported and pre-processed for the next stage. Then, MLP classification (only actual data) is utilised to observe the system performance by comparing training and validation for accuracy and loss. $(N_{true} / N_{total})\times 100$ calculates accuracy. N_{true} is the number of correct predictions, while the loss is calculated by (2). The next step is to build GAN (*D* and *G* networks) to produce the synthetic RSSI data. We considered having 125 and 625 additional samples for each class by RSSI synthesis. After GAN training, the actual + synthetic data classification procedure is similar to the MLP classification for only actual data usage. Save results containing accuracy and losses for both training and validation, respectively.

As our dataset contains 1,250 samples collected from 8 reference nodes, for training and testing, we divide this dataset by half. We focus on training and validation for both accuracy and testing. For every fingerprint location that we label as "class", we have 25 classes, and each class then has 25 RSSI samples. We evaluate the actual percentage data use compared to the actual and synthetic data performance. We observed that the increment data of 10 % compensated for it with the synthetic data we set to 125 and 625 samples.

We expected that by having more synthetic data, even if we only use a small percentage of training data (actual RSSI values), the accuracy and loss performance should be comparable to or even better than those that use 100 % of only actual data.

TABLE II. GAN PARAMETER.

Hyperparameter	Value
Number of Data Parameters	25
Data Shape	8 × 1
MLP Parameters: Epochs and Validation Split	20 and 0.1
GAN Parameter: Latent Dimension and GAN Epochs	50 and 2,500
Generator: Dense Layer	3, Activation Function: Leaky ReLU, Alpha: 0.2
Discriminator: Dense Layer	Same with Generator, Different Size of Dense Layer
Learning Rate	0.0002
Normalisation Technique	Batch Normalisation with Momentum: 0.8.
Training Batch Size	128
Loss Function	GAN Loss
Optimiser in Generate Data Process	Adaptive Moment Estimation (Adam)

III. RESULTS AND DISCUSSION

We discuss our findings on the performance comparison of increment of actual data usage compared to the full set of RSSI data. As in (4), the full RSSI set contains the actual and synthetic data. Then, training and validation for accuracy and loss for the actual and full set of RSSI. Synthetic database exploration, including the description of actual vs. synthetic data distribution, will also be discussed. Finally, we discuss it related the computational time.

A. *Actual and Full Set of RSSI Performance Comparison*

The basic idea of GANs implementation for RSSI synthesis is how we can employ small datasets and still have acceptable accuracy. In this result, we compare the percentage actual and synthesis data combination. We considered adding the actual data in a 10 % increment, while the sample generation of synthetic data is 125 and 625 samples. These numbers are selected based on the number of synthetic data generated in the final 3 and 11 times greater than the actual data size. We expect that increasing the number of data will increase the performance of the MLP in classifying data.

Figure 7 shows that the accuracy is low when only actual data applied to MLP are used. Although all data training considered was used, the mean accuracy is 0.53 (53 %). Thus, by adding the synthetic data, as observed, e.g., by only using 20 % actual data and combining with the synthetic data both 125 and 625 samples from GANs, the mean accuracy is 0.54 (54 %), slightly higher with less use of actual training data.

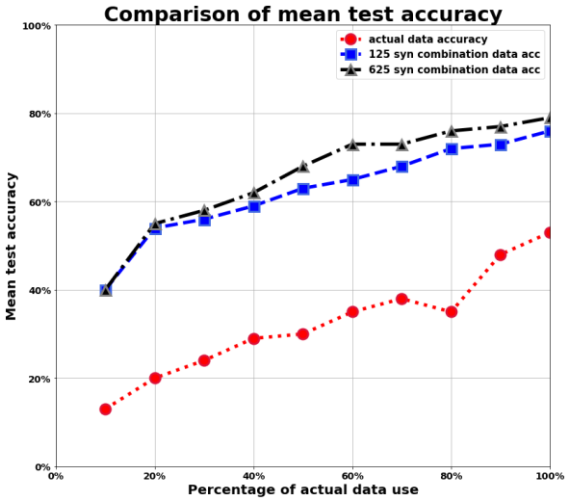


Fig. 7. Mean accuracy: actual vs. full set of RSSI.

If using only 60 % of the actual data, the mean accuracy and loss are 0.35 (35 %) and 2.41, respectively. Then, we added the synthetic data of 125 RSSI samples from the 60 % training data by GANs and combined it to the data as the data combination of 15 actual and 125 synthetics for each class. From this new combination dataset, we got the mean accuracy and loss of 0.65 (65 %) and 1.36, respectively. The improvement in accuracy is 1.85 times higher and 1.7 times better than the loss from using actual data only. Furthermore, when we added the synthetic to 625 samples to each class, the mean accuracy improved to 0.73 (73 %) or twice better accuracy than using only 60% of the actual data for training. The loss performance also improved almost two times (1.95x) better.

Interestingly, when we used 100 % actual data only, we obtained the accuracy and loss of only 0.53 (53 %) and 1.76, respectively. Compared to the 60 % actual data and combined with 125 samples of synthetic data, we obtained a much improvement in accuracy and loss for 0.65 and 1.36, respectively. So, using only 60 % data and a synthetic combination, we can achieve a better performance by about 1.2 times. Thus, the synthetic data method by implementing GANs can be used to tackle data scarcity and alleviate the burden in offline database construction by taking only the small dataset for the fingerprint database.

Moreover, as expected, when we added the 100 % usage of actual data with the synthetic data, e.g., adding the 125 samples of synthetic data, the mean accuracy improved to 0.76 (76 %) and loss of 0.85. When increasing the number of synthetic data samples to 625, the mean accuracy becomes 0.79 (79 %), and the mean loss is 1.03.

Table III shows that the mean accuracy performance is improved by adding synthetic data. However, when adding the 645 samples, some loss performance decreased. This high number of additional samples means that the synthetic data have some optimal numbers. Here, we are not yet considering the optimal number. However, seeing some improvement by adding 125 samples, it is accepted that that number of samples can be applied. An additional 125 samples to each fingerprint location/class give better performance in accuracy and loss in all the increments of data percentage used. Thus, the synthetic dataset became 3,125 samples (almost three times larger than the actual data; 1,250).

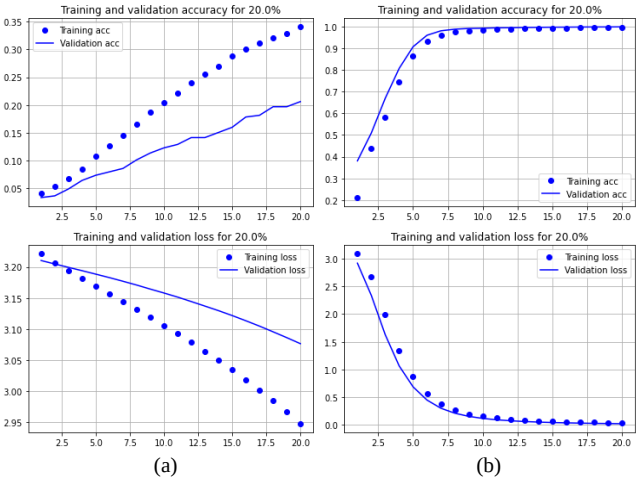
TABLE III. PERFORMANCE COMPARISON.

Real / Syn	20 %		60 %		100 %	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
0	20 %	3.04	35 %	2.41	53 %	1.76
125	54 %	2.52	65 %	1.36	76 %	0.85
625	55 %	4.26	73 %	1.23	79 %	1.03

B. *Training and Validation: Accuracy and Loss*

Our dataset is 10 % smaller than our primary reference dataset [12]. However, the trend is similar; when we combined this small dataset with the synthetic data, we generated by GANs, the performance of both accuracy and loss in classification is comparable to even better than using all the actual training datasets.

As an additional number of samples of 625 is giving marginal improvement, we will consider only the synthetic data sample of 125 in the following discussion. Figure 8 shows how synthetic databases can improve performance in training and validation, accuracy, and loss. The 20 % actual data usage performed will be compared with the synthetic data combination.

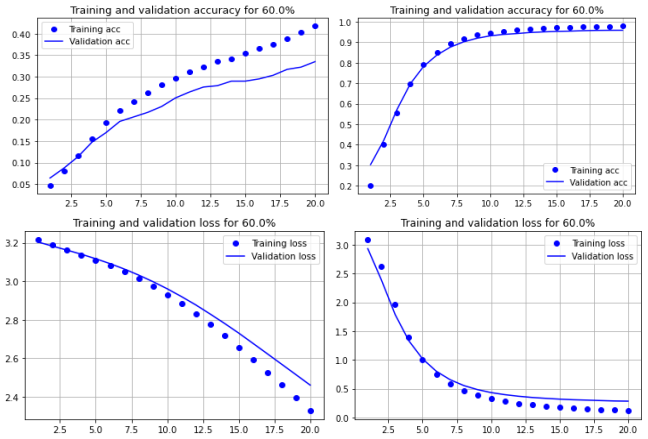


(a) (b)

Fig. 8. Training and validation: accuracy and loss for (a) actual vs. (b) full set of RSSI (20 % actual data).

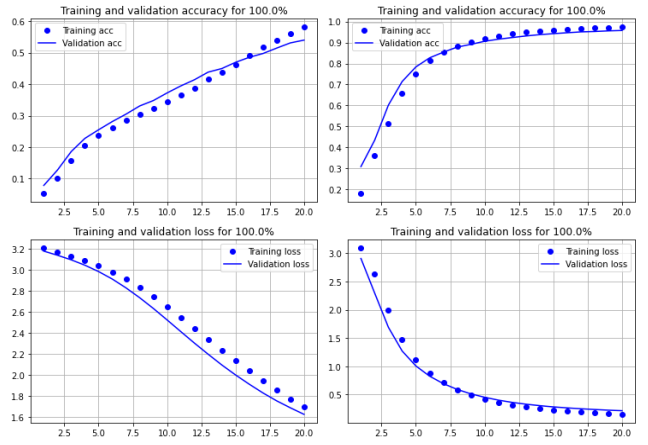
Naturally, when we added more data for training, the

accuracy and loss performance improved. The benefit of additional synthetic data is that we can start improving only by 20 % and later for actual data usage. Let us compare the 60 % and 100 % data usage as shown in Fig. 9 and Fig. 10.



(a) (b)

Fig. 9. Training and validation: accuracy and loss for (a) actual vs. (b) full set of RSSI (60 % actual data).



(a) (b)

Fig. 10. Training and validation: accuracy and loss for (a) actual vs. (b) full set of RSSI (100 % actual data).

C. *Synthetic RSSI Exploration*

We need to explore the description of the actual and synthetic database, e.g., the mean, standard deviation (std), max, and min values. In this example, we consider exploring within the 20 % data usage. We observed how its data distribution properties differ or are similar between the actual and the synthesis. This discussion evaluated the RSSI values from AP1 or reference node one as an example.

From Figs. 11 and 12 in the AP1 description, we can observe that the mean, std, max, and min values both for actual and synthesis are relatively similar. For instance, the mean RSSI and std discrepancies are -0.13 dBm and -0.27, respectively. As the discrepancy is relatively low, the synthetic data distribution is similar to the actual data. We concluded that GANs implementation is working as expected for our RSSI synthesis.

Figures 13 and 14 show the RSSI distribution for AP1 in both actual and synthetic data, respectively. The x axis shows the RSSI values in dBm and y axis as the probability density. We can observe that the RSSI values are seen to be normally distributed. Both also range from around -80 dBm to -50 dBm, while the peak RSSI value is around -6 dBm.

	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8
count	1251.000000	1251.000000	1251.000000	1251.000000	1251.000000	1251.000000	1251.000000	1251.000000
mean	-64.960032	-63.167066	-60.547562	-58.127098	-59.784972	-57.535572	-60.647482	-60.400480
std	5.318947	6.041727	5.283023	5.824005	5.651276	6.610366	4.870280	6.999221
min	-81.000000	-83.000000	-79.000000	-75.000000	-79.000000	-78.000000	-79.000000	-80.000000
25%	-68.000000	-67.000000	-65.000000	-61.000000	-64.000000	-62.000000	-64.000000	-66.000000
50%	-65.000000	-63.000000	-60.000000	-58.000000	-60.000000	-57.000000	-60.000000	-61.000000
75%	-61.000000	-59.000000	-57.000000	-55.000000	-56.000000	-54.000000	-57.000000	-55.000000
max	-51.000000	-45.000000	-43.000000	-35.000000	-43.000000	-35.000000	-48.000000	-43.000000

Fig. 11. Description of actual data.

	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8
count	3125.000000	3125.000000	3125.000000	3125.000000	3125.000000	3125.000000	3125.000000	3125.000000
mean	-64.828244	-62.345525	-61.133501	-58.000828	-60.079692	-57.449583	-60.877659	-59.826856
std	5.048397	6.052459	5.401121	5.802204	5.935214	6.757984	4.302743	7.068879
min	-78.900010	-79.999985	-73.999985	-71.948730	-72.999000	-76.999710	-71.853165	-75.000000
25%	-67.992760	-65.883120	-64.962395	-61.107315	-64.904590	-62.175540	-63.999970	-65.377090
50%	-64.158700	-63.071170	-60.954536	-57.866352	-60.001800	-57.422390	-60.377464	-58.998490
75%	-61.133244	-58.595700	-57.458595	-54.139410	-56.091064	-53.599106	-57.712220	-53.904663
max	-52.000103	-46.000000	-45.000000	-41.698257	-44.387844	-36.000000	-51.000000	-45.380405

Fig. 12. Synthetic data description (20 %, 125 samples).

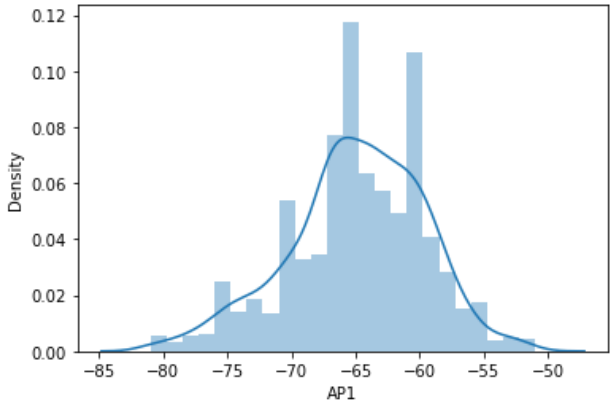


Fig. 13. Actual RSSI distribution for AP1.

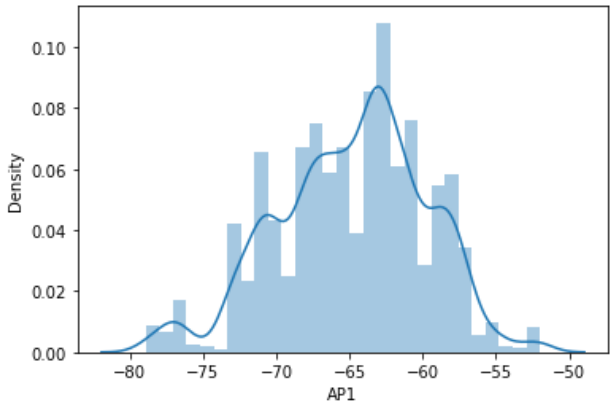


Fig. 14. Synthetic RSSI distribution for AP1.

D. *Computational Time*

GAN is an iterative method to generate the samples. By setting the hyperparameters as in Table IV, we obtained the varied computational time considering the number of actual data and the number of samples that were produced. For each iteration for 25 classes, the iteration time ranges from 124 s to 200 s.

TABLE IV. MLP ARCHITECTURE.

Hyperparameter	Value
Number of Input Neurons	8
Number of Hidden Layers	3
Number of Neurons in Hidden Layer	300
Number of Output Neurons	1
Activation Functions of Hidden Layers	ReLU

Hyperparameter	Value
Activation Functions of Output Layers	Softmax
Learning Rate	0.0002
Normalisation Technique	
Batch Size	
Loss Function	Categorical Crossentropy
Optimiser	Adaptive Moment Estimation (Adam)

So, considering the actual percentage and number of synthetic samples took around from ~50 minutes to ~90 minutes. These results are based on the PC and its specification stated in Table I and considering both hyperparameters of Table II and Table IV.

IV. CONCLUSIONS

We presented the effort to alleviate the drawbacks in offline database fingerprint construction by generating a synthetic database. This paper focused on how the deep learning approach can improve a small fingerprint database containing RSSI values. GAN has successfully generated the synthetic RSSI data. Both actual and full sets of RSSI have been compared by applying MLP to test the class output; in our case, the fingerprint location with a total of 25 locations. Our dataset is relatively small, by seeing the results of MLP classification, even by using all data, the accuracy is still relatively low (53 %).

On the other hand, by combining 20 % of actual data with its synthetic RSSI values, the accuracy is 0.54 (54 %). Moreover, when we used only 60 % of the actual data and combined them with the 625 generated RSSI samples for each fingerprint location, we can improve both performance accuracy and loss, with an accuracy of 0.73 %. As discussed earlier, the additional percentage of actual data improved the performance, naturally. We can apply synthetic data generated by GANs to consider less data when implementing fingerprint-based indoor localisation for real applications.

ACKNOWLEDGMENT

The authors would like to thank Mr. Refa Rupaksi and Mr. Aditya Bagus Krisnawan of Universitas Gadjah Mada, Indonesia for assistance in measurement campaign.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] F. Zafari, A. Gkelias, and K. K. Leung, "A Survey of indoor localization systems and technologies", *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019. DOI: 10.1109/COMST.2019.2911558.

[2] A. Yassin *et al.*, "Recent advances in indoor localization: A survey on theoretical approaches and applications", *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2017. DOI: 10.1109/COMST.2016.2632427.

[3] A. Basiri *et al.*, "Indoor location based services challenges, requirements and usability of current solutions", *Computer Science Review*, vol. 24, pp. 1–12, May 2017. DOI: 10.1016/j.cosrev.2017.03.002.

[4] P. Crane, Z. Huang, and H. Zhang, "CRAFT reducing the effort for indoor localisation", in *Proc. of 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1–5. DOI: 10.1109/PIMRC.2017.8292211.

[5] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions", *SN Computer Science*, vol. 2, art. no. 420, 2021. DOI: 10.1007/s42979-021-00815-1.

[6] F. Alhomayani and M. H. Mahoor, "Deep learning methods for fingerprint-based indoor positioning: A review", *Journal of Location Based Services*, vol. 14, no. 3, pp. 129–200, 2020. DOI: 10.1080/17489725.2020.1817582.

[7] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks", 2016. arXiv: 1701.00160v4.

[8] I. J. Goodfellow *et al.*, "Generative adversarial nets", in *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. Curran, Red Hook, NY, 2015, pp. 2672–2680. DOI: 10.3156/jsoft.29.5_177_2.

[9] W. Njima, M. Chafii, and R. M. Shubair, "GAN based data augmentation for indoor localization using labeled and unlabeled data", in *Proc. of 2021 International Balkan Conference on Communications and Networking (BalkanCom)*, 2021, pp. 36–39. DOI: 10.1109/BalkanCom53780.2021.9593240.

[10] S. Yean, P. Somani, B. S. Lee, and H. L. Oh, "GAN+: Data augmentation method using generative adversarial networks and Dirichlet for indoor localisation", in *Proc. of CEUR Workshop*, 2021. [Online]. Available: http://ceur-ws.org

[11] W. Njima, M. Chafii, A. Nimr, and G. Fettweis, "Deep learning based data recovery for localization", *IEEE Access*, vol. 8, pp. 175741–175752, 2020. DOI: 10.1109/ACCESS.2020.3026615.

[12] M. Nabati, H. Navidan, R. Shahbazian, S. A. Ghorashi, and D. Windridge, "Using synthetic data to enhance the accuracy of fingerprint-based localization: A deep learning approach", *IEEE Sensors Letters*, vol. 4, no. 4, pp. 1–4, Apr. 2020. DOI: 10.1109/LSENS.2020.2971555.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes", Dec. 2013. arXiv: 1312.6114.

[14] W. Qian, F. Lauri, and F. Gechter, "Supervised and semi-supervised deep probabilistic models for indoor positioning problems", *Neurocomputing*, vol. 435, pp. 228–238, May 2021. DOI: 10.1016/j.neucom.2020.12.131.

[15] K. M. Chen and R. Y. Chang, "Semi-supervised learning with GANs for device-free fingerprinting indoor localization", in *Proc. of GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322456.

[16] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models", in *Proc. of NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014, pp. 3581–3589. arxiv: 1406.5298.

[17] K. M. Chen and R. Y. Chang, "A comparative study of deep-learning-based semi-supervised device-free indoor localization", in *Proc. of 2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9685548.

[18] M. I. AlHajri, R. M. Shubair, and M. Chafii, "Indoor localization under limited measurements: A cross-environment joint semi-supervised and transfer learning approach", in *Proc. of 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2021, pp. 266–270. DOI: 10.1109/SPAWC51858.2021.9593245.

[19] D. J. Suroso, F. Y. M. Adiyatma, P. Cherntanomwong, and P. Sooraksa, "Fingerprint database enhancement by applying interpolation and regression techniques for IoT-based indoor localization", *Emerging Science Journal*, vol. 4, pp. 167–189, Jan. 2020, 2021. DOI: 10.28991/esj-2021-SP1-012.