Using Synthetic Data to Enhance the Accuracy of Fingerprint-based Localization: A Deep Learning Approach

Mohammad Nabati[1], Hojjat Navidan[1], Reza Shahbazian[1] , Seyed Ali Ghorashi[1,2] and David Windridge[3]

[1]*Cognitive Telecommunication Research Group, Department of Telecommunications, Faculty of Electrical Engineering, Shahid Beheshti University, Tehran 19839 69411, Iran*

[2]*Department of Computer Science & Informatics, School of Architecture, Computing and Engineering, University of East London, E16 2RD London, U.K.*

[3]*Department of Computer Science, School of Science and Technology, Middlesex University, NW4 4BT London, U.K.*

Abstract—Data collection is costly and imposes privacy issues. Many solutions are proposed in the literature to reduce this cost, such as crowd-sourcing methods in data collection or using semi-supervised algorithms to enhance the positioning accuracy. However, semi-supervised algorithms need unlabeled data, and crowd-sourcing based methods need many participants. Fingerprint-based localization methods use received signal strength (RSS) or channel state information (CSI) in wireless sensor networks to localize the users in indoor or harsh outdoor environments. In this paper, we introduce a new method to reduce the data collection costs by using synthetic data in fingerprint-based localization. We use generative adversarial networks (GANs) to learn the distribution of a limited collected data and further, produce synthetic data and feed them to the system in addition to the real collected data in order to increase the positioning accuracy. Experimental results on a benchmark dataset show that by applying the proposed method and using a combination of 10% collected data and 90% synthetic data, we can get completely close to the accuracy as if we would use the whole collected data. It means that we can use 90% less real data and reduce the data collection costs while reaching the acceptable accuracy,
. using GAN generated synthetic data

.Index Terms—Synthetic Data, GAN, Deep Learning, Fingerprint, Localization, Wireless Sensor Networks

I. INTRODUCTION

The rapid development of smartphones has led to increasing the demands of location-based services (LBSs) based on wireless sensor networks (WSN) in different areas such as academic researches, industries, and commercial applications [1]–[3]. Localization services such as global positioning system (GPS) or global navigation satellite system (GNSS) are only available in outdoor environments, and these satellite-based methods do not provide acceptable accuracy in indoor or harsh outdoor environments due to non-line of sight error (NLOS), fading, and shadowing effects [4]. Ranging based methods such as time of arrival (TOA), angle of arrival (AOA) and receive signal strength (RSS) are employed to extract the distance between transmitters (or BSs) and receiver [2]. In these methods, the locations of BSs are known in advance. After obtaining the distance between the transmitter and receiver, triangulation methods are employed to estimate the location of the user [5]. Both TOA and AOA incur an enormous cost but do not provide the expected accuracy in indoor environments because of practical issues such as NLOS and fading effect [1]. On the other hand, all modern smartphones are equipped with energy detector hardware to extract the ranging information from the RSS. The accuracy of these techniques are highly impressed by the multi-path effect. The RSS also can be used for fingerprinting based methods which is described in the following.

Corresponding author: S. Ali Ghorashi

Fingerprinting methods are used in indoor environments to improve the accuracy of RSS based localization [2]. They are a subset of localization approaches in which the signal of multiple base stations (BSs) such as WiFi, Bluetooth, ZigBee, light, and RFID is used [4] to determine the location of a receiver, based on the received strength of signal. Among the mentioned wireless systems, WiFi has attracted much interest than others, because it is readily available in modern smartphones and other communication devices [2]. In this paper, we use the term APs to refer to WiFi BSs. Fingerprinting methods have two-phases: offline (or training) and online (or test) phase. The way of data collection in the offline phase depends on localization purposes. Sometimes it is needed to find the area that users are located on. In these problems, class labels are assigned to each area, and fingerprints such as RSS or CSI are collected for each class, separately (classification problem). On the other side, when exact location estimation is required (regression problem), in the training phase, fingerprints such as RSS or CSI are collected in specific locations of the environment, known as reference points (RPs). In the online phase, the target user receives RSS or CSI from multiple BSs or APs. This information is further fed to the system model that is already trained in the offline phase and finally, the area or location of the user is estimated. In this paper, we focus on classification problem and use RSS as the power of receiving signals from multiple APs.

Machine learning methods and particularly deep learning ones are recently used to recognize the statistical patterns of gathered dataset to train the system model in the offline phase of fingerprint-

based localization [3]. A deep model usually consists of several layers of neural networks which are connected by links (weights) and activation functions are used on the output of each layer. The accuracy of fingerprint-based localization methods depends on the number of samples in training phase. Therefore, it is crucial to work on new methods that can reduce data collection costs, while reaching an acceptable accuracy.

Many researchers have tried to increase the accuracy of fingerprint-based localization methods. For the tracking of users, authors in [6] have focused on using additional hardware besides the fingerprints including magnetometer, gyroscope, accelerometer, and barometer to measure the direction of the magnetic field, angular velocity, 3D acceleration, and atmospheric pressure, respectively. Authors in [7] propose a hybrid generative discriminative approach for handling unlabeled data using a small number of labeled data. In [8] Authors proposed a semi-supervised deep learning approach to reduce the cost of collecting labeled data. This method uses all collected signals in the smooth trajectory of a user, based on constructing a neighborhood graph (similarity matrix) and minimizing introduced cost function. The conventional methods to construct neighborhood matrix do not consider the physical location of labeled data. Authors in [9] propose GRASSMA approach for taking the location of labeled data into account.

All the methods mentioned above need "real unlabeled" data to reach an acceptable accuracy. Data gathering without losing the privacy of users is a challenging issue. In this paper, we propose a new method to improve the accuracy of localization while using less real data. We use GANs to learn the distribution of collected data which has a limited size because of cost considerations. This trained GAN model further generates synthetic data with maximum similarity to the real collected data that can be used as an extra input to the localization system. GAN already have been used for augmentation of data and classification problems in the other research areas including palm-print recognition [10], eye contact detection [11], remote sensing image scene [12], object classification [13], and more generally in computer vision [14]. To the best of our knowledge, this is the first time that synthetic data generated by GAN is used to increase the fingerprint-based localization method.

The rest of this paper is organized as follows: in section II, the conventional system model for fingerprint-based classification problem is presented. In section III, the proposed deep learning-based model to generate synthetic data is explained. experimental results and conclusion are presented in sections IV and V, respectively.

II. SYSTEM MODEL

We use small letters (e.g. a) for scalar numbers, boldface small letters (e.g. $\mathbf{a}$) for vectors, capital letters (e.g. A) for a function and boldface capital letters (e.g. $\mathbf{A}$) for showing matrices. The schematic of an indoor environment for classification problem has depicted in Fig. 1. In the first step, labels are assigned for each area or room. Then, RSS values from multiple APs are gathered for each area and after that system model should be trained from collected dataset to complete the training phase. A deep neural network can extract the patterns of a dataset for the desired task, and we use a deep structure, $F(\mathbf{x}, \theta_c)$ to train the system model for identifying the patterns of each room from RSS values; where, $\mathbf{x} \in \mathbb{R}^{1\times\mathbf{M}}$ is the input vector

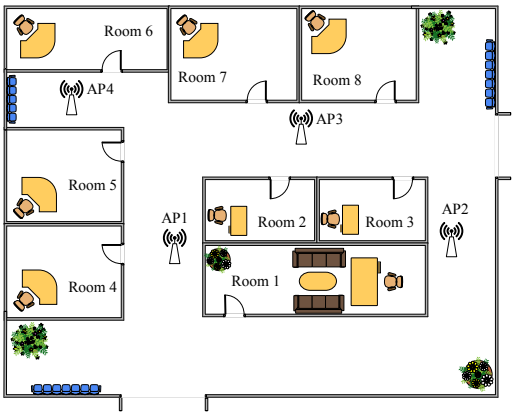

Fig. 1: A schematic of the indoor environment for localization.

(RSS vector in our problem) and the C nodes are designed for each class in the output side. Typically, the loss function in multi-class classification problem is cross-entropy or log-likelihood which is defined as follows [15]:

$$L(\theta_c) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log\hat{y}_{ij} \tag{1}$$

where, $y_{ij} = \{1 \;\; if \;\; i \in c, \;\; 0 \;\; o.w\}$, N is the number of all observations for classification, C is the number of classes, y_{ij} is real label value, $\hat{y}_{ij}$ is the predicted value from input values over $F(\mathbf{x}, \theta_c)$ in training phase, and θ_c is the parameters of the deep model in the training phase. Equation 1 can be written in vector form over the class summation as follows:

$$L(\theta_c) = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i\log\hat{\mathbf{y}}_i^T \tag{2}$$

where, $\mathbf{y}_i \in \mathbb{B}^{1\times C}$ and $\mathbb{B} \in \{0,1\}$. Therefore, $\mathbf{y}_i$ is one of the $[1,\ 0,\ 0,\ldots,\ 0]_1$, $[0,\ 1,\ 0,\ldots,\ 0]_2$, ..., $[0,\ 0,\ 0,\ldots,\ 1]_C$ categories. Parameter $\hat{y}_{ij}$ is a float number between 0 and 1 which is predicted by the deep model. The log function has a straight effect on $L(\theta_c)$ when $\hat{y}_{ij}$ increases. It means that if $y_{ij} = 1$ for any observation, $L(\theta_c)$ will be declined when $\hat{y}_{ij}$ tends to 1. Backpropagation algorithm with Adam [16] optimizer is used for minimizing the log-likelihood cost function. After optimization process in training phase, and obtaining the weights of $F(\mathbf{x}, \theta_c)$, it is ready to predict class labels from input values in test phase. Maximum value from outputs of C nodes, is the predicted class by the trained system model.

III. PROPOSED METHOD

Generative Adversarial Network (GAN), introduced by Goodfellow et al. in 2014 [17], is a method to learn the features of given data, and then generating synthetic data with maximum similarity to the input. GAN generally consists of two different parts: the Generator and the Discriminator. The generator is responsible for learning the distribution of the training dataset and generating simulated data by input noise, that matches the distribution of original data. Meanwhile, discriminator takes these data as input, and by comparing with real data, evaluates the authenticity of them. By continuously training these two networks together, the generator is finally able to create synthetic data that matches the distribution of real data and can fool the discriminator. GANs are mainly used in computer vision applications,

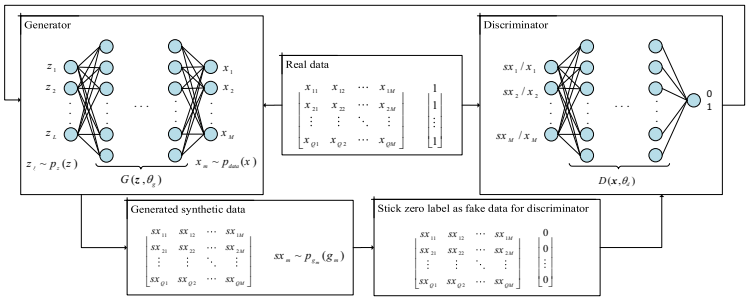

Fig. 2: The main structure of GAN in the training phase, where M is the dimension of input vector, and Q is the number of observation for training phase of GAN. The parameter Q is equal to the fraction or entire number of class observations.

and here we have used them to reach high accuracy performance in fingerprint-based localization when the real (collected) data is limited. We use a benchmark dataset in a classification problem with four classes. GAN produces synthetic data for each class; therefore, we describe the process for one class that is similar to other classes. RSS dataset for one class is as follows:

$$\mathbf{R}_c = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ r_{21} & r_{22} & \cdots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{K1} & r_{K2} & \cdots & r_{KM} \end{bmatrix} \tag{3}$$

where M is the number of APs in the environment and K is the number of labeled data in the desired class. Each column of the above matrix has a distribution over the desired class. Therefore, we define $\mathbf{x} \in \mathbb{R}^{1\times\mathbf{M}}$ and the goal of the generator is to map the prior noise $\mathbf{z} \in \mathbb{R}^{1\times\mathbf{L}}$ to $\mathbf{R}_c$ distribution. The main structure of GAN is depicted in Fig. 2. The process of producing synthetic data commences with using the bellow cost function:

$$\min_G \max_D \; L(D,G) = E_{x\sim p_{data}(x)}[\log D(\mathbf{x})] + E_{z\sim p_z(z)}[\log(1-D(G(\mathbf{z})))] \tag{4}$$

This cost function consists of two-parts and the goal of the discriminator is to maximize the probability of correct assigning labels to real and synthetic data. Both G and D are differentiable functions represented by a multilayer perceptron (MLP). Generator learns how to map the latent noise $\mathbf{z} \sim p_z(z)$ to real data distribution by using $G(\mathbf{z}, \theta_g)$ structure, where, θ_g shows the parameters of the MLP in the generator. On the other side, discriminator learns how to distinct between real and synthetic data which are shown by using $D(\mathbf{x}, \theta_d)$ structure, where θ_d shows the parameters of the discriminator. Discriminator has a binary classification structure in output side which 0 and 1 show synthetic and real data, respectively. The cost function can be represented for generative and discriminative model separately when the other side is fixed. Therefore, the discriminator loss defines as follows:

$$L(\theta_d) = E_{x\sim p_{data}(x)}[\log D(\mathbf{x},\theta_d)] + E_{z\sim p_z(z)}[\log(1 - D(G(\mathbf{z},\theta_g)))] \tag{5}$$

Also, the loss function of generator defines as follows:

$$L(\theta_g) = E_{z\sim p_z(z)}[\log(1 - D(G(\mathbf{z},\theta_g)))] \tag{6}$$

The first term of Eq.(4) will vanish in gradient updating step because it does not effect on the generator when it is fixed. The process of updating θ_d and θ_g until convergence has depicted in algorithm

Algorithm 1 (learning process of GAN)

initiate { e = epochs, s = step size for updating θ_d; Adam parameters (e.g β_1, $\beta_2 \in [0,1)$ and η) }

for i in 1:e

  for t in 1:s

    sample batch $\mathbf{Z} \in \mathbb{R}^{B\times L} \sim p_z(z)$

    sample batch $\mathbf{X} \in \mathbb{R}^{B\times M}$ from data

    update θ_d by gradient ascent based optimizer (e.g. Adam)

$$\mathcal{L}(\theta_d) = \frac{1}{B}\sum_{b=1}^{B}\left[\log D(\mathbf{X}_b,\theta_d) + \log(1-D(G(\mathbf{Z}_b,\theta_g))\right]$$

$$\psi_d = \frac{\partial}{\partial\theta_d}\mathcal{L}(\theta_d)$$

$$\varsigma_d^t = \beta_1.\varsigma_d^{t-1} + (1-\beta_1).\psi_d, \quad \xi_d^t = \beta_2.\xi_d^{t-1} + (1-\beta_2).\psi_d^2$$

$$\hat{\varsigma}_d = \varsigma_d^t/(1-(\beta_1)^t), \quad \hat{\xi}_d = \xi_d^t/(1-(\beta_2)^t)$$

$$\theta_d^{t+1} = \theta_d^t + \eta\frac{\hat{\varsigma}_d}{\sqrt{\hat{\xi}_d + \varepsilon}}$$

  end

  sample batch $\mathbf{Z} \in \mathbb{R}^{B\times L} \sim p_z(z)$

  update θ_g by gradient descent based optimizer (e.g. Adam)

$$\mathcal{L}(\theta_g) = \frac{1}{B}\sum_{b=1}^{B}\log(1-D(G(\mathbf{Z}_b,\theta_g)))$$

$$\psi_g = \frac{\partial}{\partial\theta_g}\mathcal{L}(\theta_g)$$

$$\varsigma_g^i = \beta_1.\varsigma_g^{i-1} + (1-\beta_1).\psi_g, \quad \xi_g^i = \beta_2.\xi_g^{i-1} + (1-\beta_2).\psi_g^2$$

$$\hat{\varsigma}_g = \varsigma_g^i/(1-(\beta_1)^i), \quad \hat{\xi}_g = \xi_g^i/(1-(\beta_2)^i)$$

$$\theta_g^{i+1} = \theta_g^i - \eta\frac{\hat{\varsigma}_g}{\sqrt{\hat{\xi}_g + \varepsilon}}$$

end

Note 1: In all lines i and t are interpreted as iteration over loops unless in $(\beta_1)^t$, $(\beta_2)^t$, $(\beta_1)^i$, and $(\beta_2)^i$ which are symbols of power.

Note 2: $\mathbf{X}_b$ and $\mathbf{Z}_b$ are b'th rows of $\mathbf{X}$ and $\mathbf{Z}$, respectively.

1. Convergence occurs when $D(\mathbf{x}, \theta_d) = \frac{1}{2}$ and it means that the discriminator is not able to distinguish between real and synthetic data. After convergence, the generator side is ready to produce synthetic samples for desired class from the same prior noise distribution $\mathbf{z} \sim p_z(z)$. In the next step, synthetic data are combined with real data in each class, and the conventional classification with the deep model is used for the training stage, as mentioned in the previous section. Therefore, the whole RSS (**WR**) data which consists of real RSS (**R**) and synthetic RSS (**SR**) data, can be defined for each class as follows:

$$\mathbf{WS}_c = \begin{pmatrix} \mathbf{R}_c \\ \mathbf{SR}_c \end{pmatrix} \tag{7}$$

where, $\mathbf{R}_c \in \mathbb{R}^{K\times M}$, $\mathbf{SR}_c \in \mathbb{R}^{P\times M}$, $\mathbf{WR}_c \in \mathbb{R}^{(K+P)\times M}$.

IV. EXPERIMENT METHODOLOGY AND RESULTS

The dataset used in this paper is provided by Rajen Bhatt [18], and consists of 2000 RSS samples collected from 7 APs into four different rooms (4 classes). We randomly select half of the data for training and the other half for the test phase. Both training and test datasets contain 250 data samples of each class. All data (both train and test) are standardized before giving them to the classification model. The model used for classification is an MLP neural network that consists of 6 densely connected layers. The inputs are RSS samples, and outputs are the probability of the presence of input data in each class. Both classification and GAN models are implemented on Tensorflow 1.13 and accelerated by Geforce RTX 2060. To prove the validity of the method introduced in this paper, we perform multiple experiments. In

Table 1: Effect of adding synthetic data samples to 10% and 100% of real data on test accuracy and log-likelihood loss.

Real Data / Synthetic Data	10% (25 Samples)		100% (250 Samples)	
	Accuracy	Log Loss	Accuracy	Log Loss
0	62.0%	1.03	95.3%	0.14
250	92.6%	0.24	97.1%	0.08
500	93.0%	0.28	97.4%	0.08
750	94.5%	0.24	97.3%	0.08
1000	94.4%	0.25	97.2%	0.08

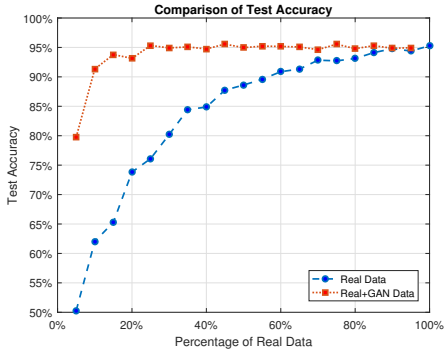

Fig. 3: Comparison of classification accuracy between purely real data (blue line) and real data combined with synthetic data (red line).

the first experiment, 10% of training data in each class (25 samples) is randomly selected, and is fed to the GAN model to generate synthetic data. We sweep over the data generated by GAN and increase the total number of data. The second experiment is similar to the first one, with the exception that all of the training data (250 samples) are selected. The results on the test accuracy and log-likelihood loss for both experiments are presented in Table 1. The accuracy is defined as $\left({}^{N_{true}}/_{N_{total}}\right)\times 100$, where N_{true} is the number of true predicted class of test data and N_{total} is the total number of test data which is equal to 1000 in our expriments. Also the log-likelihood loss is defined in Equation 1. In order to minimize the effects of randomness in the final results, the process of randomly selecting data, measuring the classification accuracy, generating synthetic data and then measuring the final accuracy are done several times. Each neural network model is trained and validated over 100 times with different model initial seeds. Afterward, the average of the test accuracy and test loss of these runs are reported as final results. In the final experiment, a fraction of real data is randomly selected to generate synthetic data, in a way that the total number of data after adding synthetic data will be equal to the total number of original data in each class (e.g. K=250). As depicted in Fig. 3, by sweeping over the fraction of real data used from 5% to 100%, we measure the test accuracy and loss to evaluate the effect of synthetic data in the classification's accuracy. As depicted in Fig. 3, the test accuracy of the model is around 50% when only a small fraction of real data is used; however by adding synthetic data, it will increase up to 80%. It can be concluded that adding synthetic data to different fractions of real data will increase the accuracy, significantly.

V. CONCLUSION

In this paper, we showed that how synthetic data can improve the accuracy of a fingerprint-based localization, where data collection process is time-consuming and costly. We proposed to use GAN in order to generate synthetic data to provide high accuracy localization, while only a limited amount of labeled data is used. Our experimental results showed that the proposed method for classification by using only 10% of real data combined with produced synthetic data can get close to the accuracy of a similar system using 100% of real labeled data. This reduces the expenses of data collection, significantly and encourages us to apply the idea to other fields rather than fingerprint-based localization.

REFERENCES

[1] A. Khalajmehrabadi, N. Gatsis, and D. Akopian, "Modern wlan fingerprinting indoor positioning methods and deployment challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1974–2002, 2017.

[2] E. Homayounvala, M. Nabati, R. Shahbazian, S. A. Ghorashi, and V. Moghtadaiee, "A novel smartphone application for indoor positioning of users based on machine learning," in *Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. ACM, 2019, pp. 430–437.

[3] Z. E. Khatab, A. Hajihoseini, and S. A. Ghorashi, "A fingerprint method for indoor localization using autoencoder based deep extreme learning machine," *IEEE sensors letters*, vol. 2, no. 1, pp. 1–4, 2017.

[4] X. Wang, L. Gao, S. Mao, and S. Pandey, "Csi-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2016.

[5] R. Shahbazian and S. A. Ghorashi, "Distributed cooperative target detection and localization in decentralized wireless sensor networks," *The Journal of Supercomputing*, vol. 73, no. 4, pp. 1715–1732, 2017.

[6] N. Yu, X. Zhan, S. Zhao, Y. Wu, and R. Feng, "A precise dead reckoning algorithm based on bluetooth and multiple sensors," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 336–351, 2017.

[7] R. W. Ouyang, A. K.-S. Wong, C.-T. Lea, and M. Chiang, "Indoor location estimation with reduced calibration exploiting unlabeled data via hybrid generative/discriminative learning," *IEEE transactions on mobile computing*, vol. 11, no. 11, pp. 1613–1626, 2011.

[8] Y. Gu, Y. Chen, J. Liu, and X. Jiang, "Semi-supervised deep extreme learning machine for wi-fi based localization," *Neurocomputing*, vol. 166, pp. 282–293, 2015.

[9] M. Zhou, Y. Tang, W. Nie, L. Xie, and X. Yang, "Grassma: graph-based semi-supervised manifold alignment for indoor wlan localization," *IEEE Sensors Journal*, vol. 17, no. 21, pp. 7086–7095, 2017.

[10] G. Wang, W. Kang, Q. Wu, Z. Wang, and J. Gao, "Generative adversarial network (gan) based data augmentation for palmprint recognition," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2018, pp. 1–7.

[11] Y. Mitsuzumi and A. Nakazawa, "Eye contact detection algorithms using deep learning and generative adversarial networks," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018, pp. 3927–3931.

[12] D. Ma, P. Tang, and L. Zhao, "Siftinggan: Generating and sifting labeled samples to improve the remote sensing image scene classification baseline in vitro," *IEEE Geoscience and Remote Sensing Letters*, 2019.

[13] M. B. Bejiga and F. Melgani, "Gan-based domain adaptation for object classification," in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2018, pp. 1264–1267.

[14] Y.-J. Cao, L.-L. Jia, Y.-X. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X.-X. Li, and H.-H. Dai, "Recent advances of generative adversarial networks in computer vision," *IEEE Access*, vol. 7, pp. 14 985–15 006, 2018.

[15] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[18] J. G. Rohra, B. Perumal, S. J. Narayanan, P. Thakur, and R. B. Bhatt, "User localization in an indoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks," in *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*. Springer, 2017, pp. 286–295.