Sensor applications

A Fingerprint Method for Indoor Localization Using Autoencoder Based Deep Extreme Learning Machine

Zahra Ezzati Khatab[1*], Amirhosein Hajihoseini[1*], and Seyed Ali Ghorashi[1,2**]

[1] Cognitive Telecommunication Research Group, Department of Electrical Engineering, Shahid Beheshti University, Tehran 1983969411, Iran

[2] Cyberspace Research Institute, Shahid Beheshti University, Tehran 1983969411, Iran

[*] Student Member, IEEE

[**] Senior Member, IEEE

Manuscript received December 11, 2017; accepted December 19, 2017. Date of publication December 27, 2017; date of current version January 23, 2018.

Abstract—By growing the demand for location based services in indoor environments in recent years, fingerprint based indoor localization has attracted much research interest. The fingerprint localization method works based on received signal strength (RSS) in wireless sensor networks. This method uses RSS measurements from available transmitter sensors, which are collected by a smart phone with internal sensors. In this article, we propose a novel algorithm that takes advantage of deep learning, extreme learning machines, and high level extracted features by autoencoder to improve the localization performance in the feature extraction and the classification. Furthermore, as the fingerprint database needs to be updated (due to the dynamic nature of environment), we also increase the number of training data, in order to improve the localization performance, gradually. Simulation results indicate that the proposed method provides a significant improvement in localization performance by using high level extracted features by autoencoder and increasing the number of training data.

Index Terms—Sensor applications, indoor localization, fingerprint, wireless sensor network, autoencoder, deep extreme learning machine.

I. INTRODUCTION

By increasing the demand for location based services in indoor areas, indoor localization has attracted much research interest. In recent years, fingerprint based methods have been used for indoor localization in Wireless Sensor Networks (WSNs), due to their advantages of lower cost and flexible applications [1], [2]. A WSN is a network formed by a large number of sensor nodes to monitor physical or environmental conditions. In fingerprint based localization, the received signal strength (RSS) is measured, and these measured RSSs and the corresponding measurement positions are used to estimate the location of target nodes [3], [4].

Fingerprint based methods usually consist of two offline and online phases. During the offline phase, RSS samples of each transmitter sensors are taken at different reference points (RPs) by a smartphone with internal sensors and are stored in a database. During the online phase, the samples of received power on the target location are compared with the database by pattern matching algorithms, and target location is estimated, accordingly. Pattern matching algorithms include deterministic and probabilistic methods [5]. Training probabilistic methods such as Horus systems may require largest dataset than that of deterministic methods [6], [7]. Deterministic method, such as k nearest neighbors (k-NN), support vector machine (SVM), and linear discriminant analysis, use the similarity metrics such as Euclidean distance to differentiate the received power of target. Then, the target location is estimated based on the closest stored fingerprint in the database [7]. In [1], an indoor localization algorithm for noisy WSNs based on an innovative multi-objective evolutionary model, which optimizes the weight of the calibration point in fingerprint localization in noisy environments, is introduced. Another localization algorithm which utilizes trilateration method in WSNs is presented in [8]. In [9] a semi supervised manifold learning method is used for mobile node localization in WSNs.

Neural networks are also used for fingerprinting [10]. Deep learning (DL) based neural networks can fully explore the features by a multilayer feature representation framework and get the optimal weights [11]. DL performs better than traditional neural networks in terms of discriminative feature extraction and performance; however, due to the slow gradient based learning algorithms used for training, as well as parameters tuned iteratively for the entire system, the training of DL is too cumbersome and time consuming [12]. To address this problem, an Extreme Learning Machine (ELM) is utilized, whose learning speed is faster than DL [13]. Random generation of input weights and the bias of ELM, leads to the fast speed in comparison with DL. ELM has been successfully employed in many applications such as classification and regression problems [14], [15]. In [16], two kinds of robust ELMs are presented to address the noisy measurements in indoor positioning systems. In [17], Semi supervised Deep Extreme Learning Machine (SDELM) approach is explained, which uses random weights and bias for network training. This method uses graph Laplacian regularization in ELM formulation in order to import large number of training data and performs good training and testing speeds. In ELM method, the parameters in hidden nodes, the input weights and the bias, are assigned randomly [13]. In [18], to deal with the dynamics of the indoor environment, an algorithm is proposed which uses the average of a number of maximum RSS observations by analyzing of the spatial resolution of the signal strength.

Considering the aforementioned challenges, in this paper, we propose an Autoencoder based Deep Extreme Learning Machine (ADELM) fingerprint localization algorithm. It uses autoencoder

Corresponding author: Seyed Ali Ghorashi (e-mail: a_ghorashi@sbu.ac.ir).

Associate Editor: F. Falcone.

Digital Object Identifier 10.1109/LSENS.2017.2787651

instead of random weight generation that leads to a discriminative feature extraction and improves localization performance. Also, in order to improve the localization performance regarding the dynamic nature of the environment, we show that how the increase of the number of training data can affect the proposed fingerprint based localization.

The structure of this paper is as follows. Section II introduces the basic knowledge of ELM and autoencoder. The proposed localization algorithm (ADELM) is presented in Section III. Section IV shows the experimental results and confirms the effectiveness of the proposed algorithm, and the article is concluded in Section V.

II. PRELIMINARIES

A. Extreme Learning Machine

As mentioned above, deep neural networks are far slower than required. ELM is a supervised learning method in which the input weights and bias for network training are randomly generated and leads to higher learning speed [16]. Considering N distinct labeled samples $(\mathbf{x}_i, \mathbf{t}_i)$; $\mathbf{x}_i = [x_{i1}, ..., x_{in}]^T \in \Re^n$ is the ith RSS sample, and $\mathbf{t}_i = [t_{i1}, ..., t_{im}]^T \in \Re^m$ is its corresponding label, where n and m are numbers of transmitter sensors and classes, respectively. If $\mathbf{x}_j$ belongs to class k, the kth value of $\mathbf{t}_j$ is set to be 1 and the rest $(m-1)$ is set to be -1. ELM with $\tilde{L}$ hidden nodes can be represented by

$$f_{\tilde{L}}(\mathbf{x}_j) = \sum_{i=1}^{\tilde{L}} \boldsymbol{\beta}_i G(\boldsymbol{\rho}_i, b_i, \mathbf{x}_j),\ \boldsymbol{\rho}_i \in \Re^n,\ \mathrm{b}_i \in \Re,\ \boldsymbol{\beta}_i \in \Re^m$$
$$j = 1, 2, ..., N \tag{1}$$

where $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the output weight vector connecting the ith hidden neuron and output neurons, and $\boldsymbol{\rho}_i = [\rho_{i1}, \rho_{i2}, ..., \rho_{in}]^T$ is the input weight vector connecting the input neurons and the ith hidden neuron. b_i is the bias of the ith hidden neuron, and $G(\boldsymbol{\rho}_i, b_i, \mathbf{x}_j)$ is the output of the ith hidden neuron. The output of the ith hidden neuron with the activation function $g(x)$ is given by

$$G(\boldsymbol{\rho}_i, b_i, \mathbf{x}_j) = g(\boldsymbol{\rho}_i . \mathbf{x}_j + b_i). \tag{2}$$

The above equation can be summarized in the matrix form

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T};\quad \mathbf{H} = \begin{bmatrix} G(\boldsymbol{\rho}_1, b_1, \mathbf{x}_1) & \cdots & G(\boldsymbol{\rho}_{\tilde{L}}, b_{\tilde{L}}, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{\rho}_1, b_1, \mathbf{x}_N) & \cdots & G(\boldsymbol{\rho}_{\tilde{L}}, b_{\tilde{L}}, \mathbf{x}_N) \end{bmatrix}_{N\times\tilde{L}}$$
$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{\tilde{L}}^T \end{bmatrix}_{\tilde{L}\times m}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N\times m}. \tag{3}$$

The goal of ELM is to minimize the error between real output and expected one and its solution can be represented as [17]

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{H}\boldsymbol{\beta} - \mathbf{T}|| \ \rightarrow\ \boldsymbol{\beta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}\mathbf{T} \tag{4}$$

B. Autoencoder

Autoencoder is a neural network that is trained to replicate its input to its output. Training an autoencoder is unsupervised in the sense that no labeled data is needed. The autoencoder network may be viewed as consisting of encoding and decoding procedures. The hidden layer of an autoencoder h describes a code for representing the input data and the decoder produces a reconstruction. If the input data is $\mathbf{x} \in \Re^{D_x}$, then the encoder maps the input to $\mathbf{z} \in \Re^{D^{(1)}}$ as (5), shown below, where D_x and $D^{(1)}$ are input dimension and encoded representaion dimension, respectively.

$$\mathbf{z}^{(1)} = h^{(1)}\left(\Psi^{(1)}\mathbf{x} + b^{(1)}\right) \tag{5}$$

$h^{(1)}$ is a transfer function for the encoder. $\Psi^{(1)}$ and $b^{(1)}$ are encode weight matrix and the encoding bias, respectively. The decoder maps the encoded representation $\mathbf{z}$ back into an estimate of the original input vector

$$\hat{\mathbf{x}} = h^{(2)}\left(\Psi^{(2)}\mathbf{z}^{(1)} + b^{(2)}\right) \tag{6}$$

where $h^{(2)}$ is the decoder function, and $\Psi^{(2)}$ and $b^{(2)}$ are decoding weight matrix and decoder bias, respectively. The dimension of an undercomplete autoencoder code is less than the input dimension, and by learning such an autoencoder, it captures the most important features of the training data. The learning process is defined as minimizing a loss function that measures the error between the input and its reconstruction [19]:

$$J(\Psi, b) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}||\mathbf{z}_i^{(l-1)} - \hat{\mathbf{x}}_i^{(l)}||^2\right)\right] + \frac{\vartheta}{2}\sum_{j=1}^{u_{l-1}}\sum_{i=1}^{u_l}\left(\Psi_{ij}^{(l-1)}\right)^2 \tag{7}$$

where N and u_l are the total number of training samples and the number of units in layer l, respectively. We define $\mathbf{z}_i^1 = \mathbf{x}_i$ and the second term denotes the weight decay which is utilized to avoid overfitting, and ϑ is the weight decay parameter. If the capacity of encoder and decoder are too much, or if the hidden layer has a dimension equal to or greater than the input, this autoencoder is called overcomplete, and fails to learn anything useful [20]. A sparse autoencoder is simply an autoencoder that adds a regularizer to the cost function [21]. This regularizer is a function of the average output activation value of a neuron. In particular, if an sparsity constraint is imposed on hidden neurons, the autoencoder can extract useful features even if the hidden layer dimension equals to or be greater than the input dimension. The cost function for training an sparse autoencoder is defined as [20]

$$\hat{J}(\Psi, b) = J(\Psi, b) + \gamma . \Omega_{\text{sparsity}} \tag{8}$$

where the second term is sparsity penalty term and γ controls the weight of sparsity term. We can learn the parameters Ψ and b by minimizing the overall cost function [19]. Research has shown that the efficiency of deep autoencoders is much better than that of shallow or linear autoencoders. In a deep autoencoder network, each layer is trained as a one layer autoencoder by minimizing the error in reconstructing [20].

III. PROPOSED AUTOENCODER BASED DEEP EXTREME LEARNING MACHINE (ADELM)

Graph based learning, which uses deep autoencoder networks to extract high level features, is one of the methods to improve localization performance. As mentioned above, in the ELM network, the randomly generated input weights and bias are used to train and develop the matrix $\mathbf{H}$ in (3). In this article, we use the extracted features by autoencoder instead of traditional $\mathbf{H}$ matrix, which is developed by random weight matrix and bias. The ELM can resolve the following learning problem [22]:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{X} \tag{9}$$

where $\mathbf{X}$ represents N labeled data $(\mathbf{x}_i, \mathbf{t}_i)$. According to ELM, the loss function of deep extreme feature learning is [17]

$$l_1 = \min_{\boldsymbol{\beta}} \frac{1}{2}||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||^2 + \frac{c_1}{2}||\mathbf{H}\boldsymbol{\beta}||^2 + \frac{c_2}{2}||\boldsymbol{\beta}||^2 \tag{10}$$

where c_1 and c_2 are smoothness and regularization parameters, respectively. In order to optimize the problem in (10), the derivative of $\boldsymbol{\beta}$ is calculated with respect to l_1. For the kth layer of feature learning, the output weight is [17]

$$\boldsymbol{\beta}^k = \left(\mathbf{H}_k^T\left((c_1+1).\mathbf{I}_1\right)\mathbf{H}_k + c_2\mathbf{I}_2\right)^{-1}\mathbf{H}_k^T\mathbf{X}^{k-1} \tag{11}$$

where $\mathbf{I}_1 \in \Re^{N\times N}$ and $\mathbf{I}_2 \in \Re^{\tilde{N}\times\tilde{N}}$ are identity matrixes ($\tilde{N}$ is the number of hidden neurons), and $\mathbf{X}^{k-1}$ is the output of the $(k-1)$th layer. If the activation function is $g(x)$, then the output of the kth layer is given by using $\boldsymbol{\beta}^k$ [17]

$$\mathbf{X}^k = g\left(\mathbf{X}^{k-1}.\left(\boldsymbol{\beta}^k\right)^T\right). \tag{12}$$

After feature extraction by neurons in hidden layers, these features are put into the classifier. The training aims to find an optimal classification function by optimizing $\boldsymbol{\beta}$:

$$\min_{\boldsymbol{\beta}} \frac{1}{2}||\mathbf{F} - \mathbf{T}||^2 + \frac{c}{2}||\boldsymbol{\beta}||^2 \tag{13}$$

where $\mathbf{F} = \mathbf{H}\boldsymbol{\beta}$ and the second term $\frac{c}{2}||\boldsymbol{\beta}||^2$ is the $\ell_{2,1}$ norm regularization which is added to guarantee the generalization ability, and c is a balance factor. In addition to $\ell_{2,1}$ norm regularization, Laplacian regularization can also be added into (13) to provide a closed form solution for optimization problem [23]

$$\min_{\boldsymbol{\beta}} \frac{1}{2}||\mathbf{F} - \mathbf{T}||^2 + \frac{c}{2}||\boldsymbol{\beta}||^2 + \lambda Tr\left(\mathbf{F}^T\mathbf{L}\mathbf{F}\right) \tag{14}$$

where λ is a balance factor to control the influence of manifold regularization. $Tr(.)$ is the trace of matrix and $\mathbf{L}$ is Laplacian matrix. The graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where the degree matrix $\mathbf{D}$ is a diagonal matrix and each diagonal element is $\mathbf{D}_{ii} = \sum_{i=1}^{N}\mathbf{W}_{ij}$ where $\mathbf{W}$ is the weight edge matrix. When the edge weights are not naturally defined, one common way to process $\mathbf{W}$ is via Gaussian kernel [23]

$$\mathbf{W}_{ij} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right) \tag{15}$$

Considering $\mathbf{F} = \mathbf{H}\boldsymbol{\beta}$, the final goal of the classifier becomes

$$l_2 = \min_{\boldsymbol{\beta}} ||\mathbf{J}\mathbf{H}\boldsymbol{\beta} - \mathbf{T}||^2 + c||\boldsymbol{\beta}||^2 + \lambda Tr\left((\mathbf{H}\boldsymbol{\beta})^T\mathbf{L}\left(\mathbf{H}\boldsymbol{\beta}\right)\right) \tag{16}$$

where $\mathbf{J} = \text{diag}(1, ..., 1) \in \Re^{N\times N}$, and $\mathbf{T} \in \Re^{N\times m}$ is the corresponding labels. The optimum solution of $\boldsymbol{\beta}$ is given by

$$\frac{\partial l_2}{\partial \boldsymbol{\beta}} = 0 \;\rightarrow\; \boldsymbol{\beta} = \left(c\mathbf{I} + \mathbf{H}^T\left(\mathbf{J} + \lambda\mathbf{L}\right)\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{J}\mathbf{T}. \tag{17}$$

Since the parameters of ADELM, including deep learning parameters (c_1, c_2), and the final classifier parameters (c, λ) affect the localization performance, it is necessary to adjust these parameters. For this purpose, similar to [17] we adopt hierarchical tuning mechanism. At first we allocate the empirical values to initialize the parameters. Then, for a specified depth of network and fix deep learning parameters, we tune the classifier parameters. When the classifier parameters are optimized, deep learning parameters are adjusted. When all of the parameters are tuned, the localization performance is calculated. To study the influence of deep network depth on the performance, we increase the depth by 1 and adopt the same procedure for parameter optimization, to calculate the localization performance.

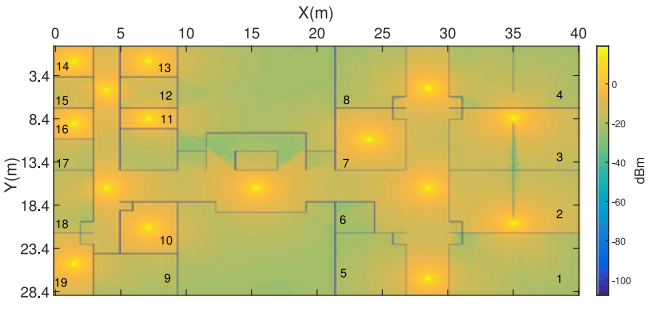

Fig. 1. Simulation area layout with transmitters' coverage and room numbers.

IV. PERFORMANCE EVALUATION

In this section, we describe our simulation setup to test our proposed algorithm for fingerprint data adopted from transmitters. In order to generate the RSS values at different points to construct the fingerprint database, the large scale path-loss model with lognormal shadowing is used [24]

$$PL(d) = PL(d_0) - 10n\log_{10}\frac{d}{d_0} - \sum_{p=1}^{P} WAF(p) + X(\sigma) \tag{18}$$

where d_0 is the reference distance and d is the distance between transmitter and the receiver node. $PL(d_0)$ is the received signal in the reference distance and its value for different transmitters is different. n is the path-loss exponent and $X(\sigma)$ is a zero mean Gaussian random variable with standard deviation σ, which models the shadowing effect. Also, WAF is the wall attenuation factor, and P is the number of walls between transmitter and the receiver node. In this article, we consider 10 dB power loss for each wall. All of the simulation are done with MATLAB 2016(a) software. The characteristics of system where these simulations are done, are as follows: The processor is Intel(R) Core(TM) i3-4160 CPU @3.60 GHz, with 8 GB RAM.

A. Model Generation

Simulations are conducted in a laboratory of 40.4×28.8 m^2, including 19 rooms in the second floor of Electrical Engineering Department of Shahid Beheshti University. The locations of the transmitters used in this simulation are shown in Fig. 1. We use 10000 fingerprint data, 1200 of them are randomly chosen as testing data and the rest as training data. All RSSs obtained in any location are normalized to the maximum power at the same point. In the first stage, a limited number of data are used as training data and the localization performance is calculated. As mentioned, increasing the number of training data improves the localization performance. To show this, we increase the number of training data step by step, and it can be seen that the localization performance is improved.

To verify the performance of the proposed algorithm, we construct the ADELM network with two hidden layers. The number of neurons in the layers are 600−200, respectively. In the first step, we use 434 data from training set to train the network and measure the localization accuracy for the test data. In order to investigate the effect of increasing the number of training data on the localization performance, we add

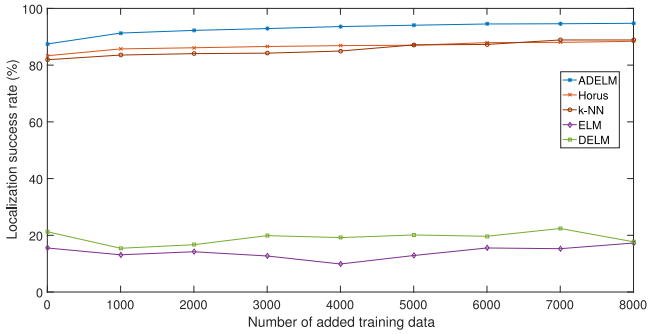

Fig. 2. Localization performance with the same 434 primary training data and adding extra training data for ADELM, Horus, *k*-NN with $k = 7$, and ELM and DELM based localization.

Table 1. Localization Success Rate for Different Methods.

Method name	ADELM	Horus	*k*-NN	ELM	DELM
Success rate(%)	92.82	86.66	85.64	14.04	19.14

1000 extra data in each step to training data and measure the localization accuracy, again. In simulation, localization success rate is defined as $N_{\text{correct}}/N_{\text{total}}$, where N_{correct} and N_{total} are the number of correct localizations and the number of total localizations, respectively.

B. Localization Performance

In this section, we evaluate the localization performance of our proposed method (ADELM) compared to Deep Extreme Learning Machine (DELM), which is an ELM network with more than one hidden layer and uses random input weights and bias for training, which is explained in [17]. In ADELM, by hierarchical optimization mechanism, the optimal values for classifier parameters are $c = 0.01$, $\lambda = 0.03$, and the optimal values for deep learning parameters are $c_1 = 0.001$ and $c_2 = 0.01$. As shown in Fig. 2 in the first stage, our proposed algorithm leads to 87.45% success in localization. By increasing the number of training data in next stages, it can be seen that the localization performance is improved. In the last stage, the localization success exceeds 94.75%. As can be seen, our proposed algorithm can improve the localization success rate by more than 73% compared with traditional DELM. Also, adding extra training data does not affect *k*-NN based localization considerably.

In Table 1, the localization success rate for the proposed method (ADELM) is compared with the performance of some different methods.

V. CONCLUSION

In this article, we proposed the autoencoder based Deep Extreme Learning Machine indoor localization method in WSNs, which utilizes the high level extracted features by autoencoder from collected data by smartphones with internal sensors. Also in order to improve the localization performance, we showed that using more labeled data as extra training data can lead to higher localization success rate. Also, we compared our algorithm performance with a traditional deep extreme learning machine which uses random input weights and bias. Simulation results confirmed that our proposed autoencoder based DELM improves the performance, and also using more training data improves the localization performance even more. This last achievement means that by using smart-phones, we can improve the localization accuracy by updating the training data set.

References

[1] X. Fang, L. Nan, Z. Jiang, and L. Chen, "Fingerprint localisation algorithm for noisy wireless sensor network based on multiobjective evolutionary model," *IET Commun.*, vol. 11, no. 8, pp. 1297–1304, Feb. 2017.

[2] M. Bshara, U. Orguner, F. Gustafsson, and L. V. Biesen, "Fingerprinting localization in wireless networks based on received-signal-strength measurements: A case study on WiMAX networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 283–294, Jan. 2010.

[3] A. H. Gazestani, R. Shahbazian, and S. A. Ghorashi, "Decentralized consensus based target localization in wireless sensor networks," *Wireless Pers. Commun.*, vol. 97, no. 3, pp. 3587–3599, Dec. 2017.

[4] A. Zhang, Y. Yuan, Q. Wu, Sh. Zhu, and J. Deng, "Wireless localization based on RSSI fingerprint feature vector," *Int. J. Distrib. Sens. Netw.*, vol. 11, no. 11, pp. 1–7, Nov. 2015.

[5] Z. Ezzati Khatab, V. Moghtadaiee, and S. A. Ghorashi, "A fingerprint-based technique for indoor localization using fuzzy least squares support vector machine," in *Proc. Iranian Conf. Elect. Eng.*, May 2017, pp. 1944–1949.

[6] M. Youssef and A. Agrawala, "Handling samples correlation in the Horus system," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Hong Kong, Mar. 2004, pp. 1023–1031.

[7] S. He and S.-H. G. Chan, "Wi-Fi fingerprint-based indoor positioning: recent advances and comparisons," *IEEE Commun. Survey Tut.*, vol. 18, no. 1, pp. 466–490, Jan. 2016.

[8] O. M. Elfadil, Y. M. Alkasim, and E. B. Abbas, "Indoor navigation algorithm for mobile robot using wireless sensor networks," in *Proc. Int. Conf. Commun. Control Comput. Electron. Eng.*, Jan. 2017. pp. 1–5.

[9] B. Yang, J. Xu, J. Yang, and M. Li, "Localization algorithm in wireless sensor networks based on semi-supervised manifold learning and its application," *Cluster Comput.*, vol. 13, no. 4, pp. 435–446, Feb. 2010.

[10] H. Dai, W. Ying, and J. Xu, "Multi-layer neural network for received signal strength-based indoor localisation," *IET Commun.*, vol. 10, no. 6, pp. 717–723, Apr. 2016.

[11] X. Wang, L. Gao, S. Mao, and S. Pandev, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[12] J. Tang, Ch. Deng, and G. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

[13] J. Liu, Y. Chen, M. Liu, and Zh. Zhao, "SELM: Semi-supervised ELM with application in sparse calibrated location estimation," *Neurocomputing*, vol. 74, no. 16, pp. 2566–2573, Sep. 2011.

[14] J. P. Nobrega and A. L. I. Oliveria, "Kalman filter based method for online sequential extreme learning machine for regression problems," *Eng. Appl. Artif. Intell.*, vol. 44, pp. 101–110, Sep. 2015.

[15] R. Savitha, S. Suresh, and H. J. Kim, "A meta-cognitive learning algorithm for an extreme learning machine classifier," *Cogn. Comput.*, vol. 6, no. 2, pp. 253–263, Jun. 2014.

[16] X. Lu, H. Zou, H. Zhou, L. Xie, and G. Haung, "Robust extreme learning machine with its application to indoor positioning," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 194–205, Jan. 2016.

[17] 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, Oct. 2015.

[18] W. Xue, W. Qiu, X. Hua, and K. Yu, "Improved Wi-Fi RSSI measurement for indoor localization," *IEEE Sensor J.*, vol. 17, no. 7, pp. 2224–2230, Apr. 2017.

[19] J, Wang, X. Zhang, Q. Gao, H. Yue, and H. Wang, "Device-free wireless localization and activity recognition: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6258–6267, Jul. 2017.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* Cambridge, MA, USA: MIT Press, 2016.

[21] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, Dec. 1997.

[22] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks" in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2004, pp. 985–990.

[23] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs" *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, Apr. 2013.

[24] G. Sun and W. Guo, "Robust mobile geo-location algorithm based on LS-SVM," *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 1037–1041, May 2005.