WiDeep: WiFi-based Accurate and Robust Indoor Localization System using Deep Learning

Moustafa Abbas
Dept. of Comp. and Sys. Eng.,
Alexandria University
Alexandria, Egypt
m.abbas@alexu.edu.eg

Moustafa Elhamshary
Dept. of Comp. and Cont. Eng.,
Tanta University
Tanta, Egypt
mostafa.elhamshary@f-eng.tanta.edu.eg

Hamada Rizk
Dept. of Comp. Sci. and Eng.,
EJUST, Alexandria, Egypt
& Tanta University, Tanta, Egypt
hamada.rizk@ejust.edu.eg

Marwan Torki
Dept. of Comp. and Sys. Eng.,
Alexandria University
Alexandria, Egypt
mtorki@alexu.edu.eg

Moustafa Youssef
Dept. of Comp. and Sys. Eng.,
Alexandria University
Alexandria, Egypt
moustafa@alexu.edu.eg

***Abstract*—Robust and accurate indoor localization has been the goal of several research efforts over the past decade. Due to the ubiquitous availability of WiFi indoors, many indoor localization systems have been proposed relying on WiFi fingerprinting. However, due to the inherent noise and instability of the wireless signals, the localization accuracy usually degrades and is not robust to dynamic changes in the environment.**

We present *WiDeep*, a deep learning-based indoor localization system that achieves a fine-grained and robust accuracy in the presence of noise. Specifically, *WiDeep* combines a stacked denoising autoencoders deep learning model and a probabilistic framework to handle the noise in the received WiFi signal and capture the complex relationship between the WiFi APs signals heard by the mobile phone and its location. *WiDeep* also introduces a number of modules to address practical challenges such as avoiding over-training and handling heterogeneous devices.

We evaluate *WiDeep* in two testbeds of different sizes and densities of access points. The results show that it can achieve a mean localization accuracy of 2.64m and 1.21m for the larger and the smaller testbeds, respectively. This accuracy outperforms the state-of-the-art techniques in all test scenarios and is robust to heterogeneous devices.

***Index Terms*—WiFi, Deep learning, indoor, localization, fingerprinting**

I. Introduction

As people spend most of their time indoors, academia and industry have recognized the value of the indoor localization problem and have devoted much effort and resources into solving it [1], [2]. Due to the wide-spread coverage of WiFi and the support of the IEEE 802.11 standard by the majority of mobile devices, most proposed indoor localization systems are WiFi-based including propagation- and fingerprinting-based techniques [3]–[10]. Propagation-based techniques, e.g. [3], [4], [11], [12], aim to model the relation between the received signal and distance without site surveying. Despite their ease of deployment without the need for prior calibration, these techniques do not work well with heterogeneous phones and their accuracy is usually less than fingerprinting-based techniques.

On the other hand, fingerprinting techniques leverage the recorded WiFi APs signatures (i.e. fingerprints) to estimate the device location. Typical fingerprint-based WiFi localization techniques work in two phases: The first one is the offline phase (i.e., calibration) during which the received signal strength (RSS) readings from the multiple access points (APs) installed in the area of interest are recorded at known locations. Then, in the tracking phase, RSS measurements from the detected APs at an unknown location are matched against the stored fingerprints to estimate the best location match either deterministically, e.g. [13], or probabilistically, e.g. [6]. Fingerprinting-based techniques are widely adopted due to their relatively good accuracy. Practically however, the deployment of such techniques faces major challenges due to the inherent noise in the wireless signals that affects localization accuracy [14], [15]. Therefore, many systems have been proposed to address these challenges over the years, e.g. [6], [16]–[18]. Probabilistic techniques such as [6], [19] can counter the inherent wireless signal noise in a better way than deterministic techniques [13]. However, they usually assume that the signals from different access points are independent to avoid the curse of dimensionality problem [20]. This leads to coarse-grained accuracy. Hybrid techniques, e.g. [16], [17], [21]–[23], leverage the sensors that are available on high-end smartphones to combat the wireless channel noise. Other techniques, e.g. [8], [24], leverage the detailed channel state information ***obtained from specialized WiFi chips*** to combat the noise. However, both of the last two categories are not supported by the vast majority of mobile devices, limiting their ubiquitous deployment.

In this paper, we propose *WiDeep*: a WiFi-based indoor fingerprinting localization system that can achieve robust and high accuracy tracking in the presence of device heterogeneity. To do this, *WiDeep* builds on deep learning to automatically capture the ***non-linear and correlated*** relation between the different access points at different fingerprint locations, without

978-1-5386-9148-9/19/$31.00 ©2019 IEEE

assuming access points' independence as in current probabilistic techniques. However, leveraging a deep network alone as in [8] may not lead to the required performance in the presence of device heterogeneity which can be considered as a form of noise. To ensure the robustness and the generalization ability of the system in this challenging scenario, we adopt a deep network model utilizing stacked denoising autoencoders to robustly extract a good representation of the relation between the noisy WiFi scans and the different fingerprint locations. Furthermore, *WiDeep* also employs a regularization technique to avoid model over-fitting and boost the robustness of the system.

During tracking, the output of the deep learning models is fused using a probabilistic framework to further handle the noise in the input signal.

We implemented and deployed *WiDeep* on different Android phones and evaluated its performance in two different testbeds: a $629m^2$ university building and a $65m^2$ residential apartment. The two buildings have different layouts and WiFi APs densities. Our results show that *WiDeep* can achieve a consistent mean accuracy of 2.64m and 1.21m in the two testbeds respectively under different scenarios. This is better in mean accuracy than traditional fingerprinting techniques and other basic deep learning techniques by at least 29.8% and up to 168%. This accuracy is maintained under significant decrease in the access points density as well as under heterogeneous devices, highlighting *WiDeep* promise as a robust and accurate indoor localization technique.

The rest of the paper is organized as follows: Section II presents an overview on how *WiDeep* works and its mathematical model. Section III presents the details of the *WiDeep* system. We evaluate the system performance in Section IV. Finally, sections V and VI discuss related work and conclude the paper respectively.

II. System Overview and Mathematical Model

A. System Overview

Fig. 1 shows the system architecture. The system runs in two phases: offline training phase and online localization phase. During the offline phase, the system builds N deep neural networks corresponding to N fingerprint training points (i.e., a deep neural network for each reference point). This helps in scaling the system to large areas as well as keeping the model size small and easier to train.

To collect the training data, the **Signature Collector** module is used to scan for the APs and their associated signal strengths at the different fingerprint locations in the area of interest. These measurements are opportunistically transfered to the *WiDeep* server in the cloud. The **Preprocessor** module is used to transform the WiFi measurements to fit the format required in the deep network training model as well as perform the required normalization. The preprocessed data is then fed to the **Noise Injector** module to corrupt the original measurements by injecting artificial noise to the collected WiFi scans. This helps not only in simulating the distorted wireless channel but also in reducing the model over-fitting by forcing

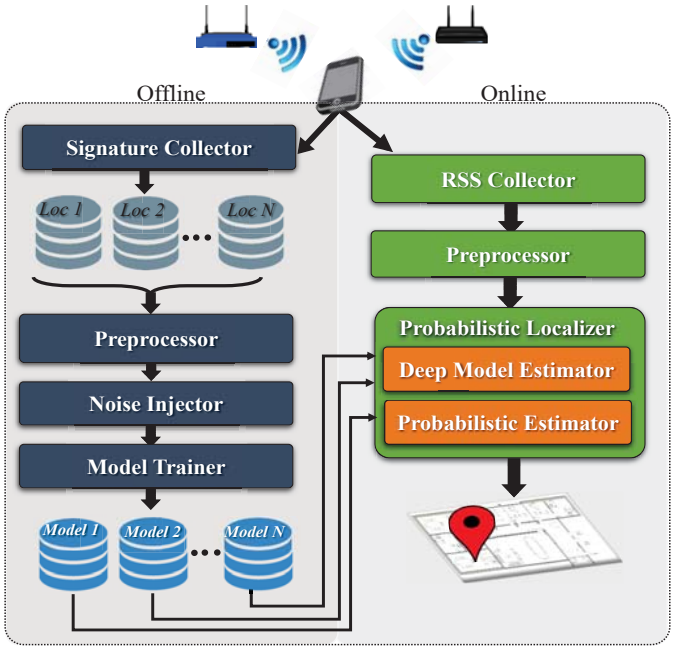


Fig. 1. *WiDeep* system architecture.

the model to learn the inherent feature of the data. For each fingerprint point, the original and corrupted noisy data are forwarded as input to the **Model Trainer** module. This module is responsible for creating and training a stacked denoising auto-encoders deep model corresponding to ***each*** fingerprint point. Moreover, it enhances the robustness of the model by avoiding over-training. Finally, all the trained models for the different fingerprint locations are stored to be used during the online localization phase.

During the online localization phase, the user is tracked in realtime. The process starts by scanning for the APs and their RSSs at the unknown user location. This data is first preprocessed (shaped and normalized) to fit with deep learning model input. Then, the **Probabilistic Localizer** module leverages the different deep models output in a probabilistic framework to estimate the most probable user location using the Deep Model Estimator and *Probabilistic Estimator* submodules respectively.

B. Mathematical Model

Without a loss of generality, we assume a 2D physical area of interest $\mathbb{L}$ containing M access points. N discrete fingerprint locations are spread over the entire area, where training data is collected. During the online localization phase, a user holding a mobile device at an unknown location $l \in \mathbb{L}$ scans for the nearby APs. Let a vector x_i of M dimensions represents a single WiFi scan. Each entry i in this vector is the received signal strength reading from access point i. The problem then becomes: given a signal strength vector $x = (x_1, ..., x_M)$, we seek to find the fingerprint location l_i that maximizes the probability $P(l_i|x)$. In the next section, we discuss the details of how *WiDeep* combines deep learning

and a probabilistic framework to achieve high accuracy and robust localization in the *continuous* space.

III. The *WiDeep* System

We present the details of the offline models construction phase and the online localization phase. We start by the preprocessor module as it is common to both phases.

A. The Preprocessor

This module is responsible for mapping the recorded WiFi RSS readings, x_i, to the corresponding feature vectors. Note that since not all M APs that are installed in the area of interest can be heard in every scan, this module assigns the weakest RSS, i.e. $-100dBm$, to the APs that are not heard in a given scan. This allows us to fix the feature vector size that is input to the machine learning model. After that, the input RSS values ranges are normalized to be in the range between [0,1] for each AP. Features normalization is known to speed up model training and increase the model robustness [25].

B. Offline Models Construction Phase

During this phase, the system builds N deep models corresponding to N fingerprint training points (i.e., a deep model for each point). The system also addresses a number of challenges including handling the noise and fluctuation of APs signals as well as reducing the model over-fitting to training data, allowing for better generalization and robustness.

We choose stacked denoising autoencoders as our model as they are able to extract the latent features from noisy data. Autoencoders are unsupervised learning models, where their goal is to learn a concise mapping that can regenerate the input to the autoencoder [26]. Denoising autoencoders extends traditional autoencoders to handle noisy data in a better way. Specifically, instead of feeding the original input to the denoising autoencoder, we feed it a noisy version. This allows the hidden layer of the autoencoder to learn important features (Fig. 2). Specifically, the denoising autoencoder is trained by first corrupting the input RSS vector x to obtain vector $\tilde{x}$. The goal is to learn the parameters of the hidden layer h so that the output ($\hat{x}$) of the autoencoder matches the original *uncorrupted* vector x. By using a noisy version of the input, the autoencoder is forced to learn the latent features of the input data. Note that the weights between the hidden and output layers are the transpose of the weight between the input and hidden layer, reflecting the decoding process of the autoencoder. Training is performed using the gradient descent algorithm, where the least square error between the *original* input data x and the reconstructed data $\hat{x}$ is used as the loss function to adjust the weights.

In the balance of this section, we first describe how to introduce noise to our input data. Then we provide the details of our deep model.

1) Noise Injector: This module aims to enhance the model ability to handle the noisy input data. This is achieved by generating corrupted variations of the collected WiFi scans. This also have the added advantage of reducing over-fitting and coping better with devices heterogeneity as we quantify in Section IV. To do that, the module adds stochastic noise to the input data through two different techniques: Masking corruption and Additive Gaussian corruption.

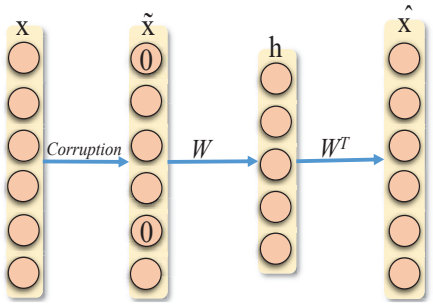


Fig. 2. Denoising Autoencoder: A sample x is stochastically corrupted to produce $\tilde{x}$. Next, the the hidden layer (h) maps $\tilde{x}$ to the output $\hat{x}$ to produce a reconstructed version of x. Error is measured by the $L(x, \hat{x})$ loss function that compares the reconstructed output with the **original noise-free** input.

Masking corruption method: The intuition behind this method is that the number of access points detected at a fixed location varies with time due to multipath and fading effects [27]. The masking method leverages this fact to emulate fluctuating access points [28]. The idea is to generate a random binary vector with specific probability of its elements to be zeros determined by the corruption fraction parameter (f). This generated binary vector is then multiplied by the original input to get a noisy input signal, where the entries of the APs corresponding to the zero random bits are dropped, emulating not hearing them (Fig. 3(a)).

Additive Gaussian corruption method: Due to the noise in the wireless channels and the diversity of the WiFi chips in different devices, the magnitude of the RSS may be shifted with some variance. Therefore, to emulate this behavior, this technique adds white Gaussian noise with a specific standard deviation s to the different entries of the RSS vector (Fig. 3(b)). The synthesized vector is finally re-normalized so that all entries are between 0 and 1.

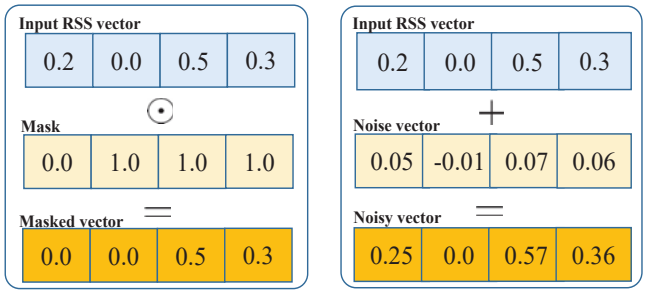
2) The Model Trainer: Fig. 5 shows the *WiDeep* deep model. It consists of a number of stacked denoising autoencoders, one stack for each fingerprint location. The used activation function is the Sigmoid function formulated as [26]:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

The Sigmoid function ensures that its output will range between 0 and 1.

To train *WiDeep* model end-to-end, we use two training stages: (1) greedy layer-wise pre-training stage and (2) a fine-tuning stage. Now, we discuss the details of each of these stages.

1) Pre-training Stage: Fig. 4 shows the steps of the pre-training deep neural network of stacked denoising autoencoders. The goal of this stage is to find good initial weights for the different layers of the network, instead of using initial random weights [29]. This is known to speed up the



(a) Masking corruption method. (b) Additive Gaussian corruption method.

Fig. 3. Examples of different corruption techniques applied to the normalized RSS signal.

convergence of the training, reduce the possibility of falling in a local minima, and avoid the vanishing gradient problem [30].

In this stage, each autoencoder in the stack is trained independently using the output of the previous autoencoder as its input (input data for the first autoencoder). Each denoising autoencoder consists of an encoder and a decoder. The encoder tries to generate a latent representation (hidden layer parameters) of the input data while the decoder tries to reconstruct the input data based on the generated (latent) code from the encoder.

2) **Fine-tuning Stage** In this stage, we train the model (Fig. 5) end-to-end. The weights are initialized to those obtained from pre-training stage. Then, each input training sample (WiFi scan) is passed through network to obtain the reconstructed data of the input scan using forward propagation. The sum of squared difference between the original input data and the reconstructed data (i.e., output of the deep network) is used as the loss function to adjust all the weights in different layers with the gradient descent algorithm [31].

C. Reducing Model Over-training

To further reduce the possibility of model over-fitting, we also use dropout regularization [28] during the fine-tuning phase. To do that, some hidden neurons in the deep neural network are temporarily dropped out stochastically (as illustrated by the crossed circles in Fig. 5). This dynamic change of the network structure allows the network to generalize better and hence become more robust to changes.

Finally, the learned weights of the N deep models for the N fingerprint locations in the area of interest are saved. Later on, during the online localization phase, these weights are used to estimate the unknown location of the mobile device as we explain in the next section.

D. Online Localization Phase

During this phase, we harness the learned models to estimate the unknown locations of the test inputs. Specifically, we input the current WiFi scan at the unknown user location to the different deep models at each fingerprinting location and leverage the reconstruction similarity in a probabilistic framework to estimate the most probable location. The intuition is that the reconstructed scan will be closer to the input scan in the fingerprinting locations near the actual user location. hence, these locations should have a higher score/probability compared to far-away locations.

More formally, the user is standing at an unknown location l receiving WiFi information that is preprocessed to obtain a signal strength vector $x = (x_1, ..., x_M)$, where M is the total number of APs in the environment. We want to find the probability of being at a fingerprint location l_i in the area of interest given the received signal strength vector x. That is, we want to find $P(l_i|x)$. Using the Bayes theorem, the posterior probability $P(l_i|x)$ is given as:

$$p(l_i|x) = \frac{p(x|l_i)p(l_i)}{p(x)} = \frac{p(x|l_i)p(l_i)}{\sum_{i=1}^{N} p(x|l_i)p(l_i)} \quad (2)$$

Where $p(l_i)$ is the prior probability that the phone is located at a given fingerprint location l_i and N is the number of locations in the fingerprint database (i.e, the training locations). Assuming that all locations are equally probable[1], Equation 2 can be rewritten as:

$$p(l_i|x) = \frac{p(x|l_i)}{\sum_{i=1}^{N} p(x|l_i)} \quad (3)$$

In traditional fingerprinting systems, e.g. [6], $p(x|l_i)$ ***is usually obtained by assuming the independence of the APs using the RSS histograms, which does not capture the rich and correlated relation between the different APs***. On the contrary, to calculate $p(x|l_i)$ *WiDeep* leverages the constructed offline deep learning models. Specifically, reconstructed versions of the input scan x_i are obtained from each deep model along with the associated similarity score to the input signal.

To obtain this similarity score, we use a radial basis kernel as a similarity function since its output is bounded within 0 and 1 and can therefore be interpreted probabilistically. Denoting the output of the similarity function as $p(x|l_i)$ for the i^{th} model, we have that:

$$p(x|l_i) = \frac{1}{n} \sum_{j=1}^{n} e^{-\frac{\|x_{ij} - \hat{x_{ij}}\|}{\lambda\sigma}} \quad (4)$$

Where x_{ij} and $\hat{x_{ij}}$ are the original and the reconstructed input data of the j^{th} scan respectively, σ is the variance of the input scans, λ is selected to be a scaled version of the coefficient of variation (CV) of the input scans, and n is the total number of scans used in location determination. We quantify the effect of these parameters on performance in Section IV.

Till now, we can assign different probabilities to the discrete fingerprinting locations based on the input scan(s). To enable tracking the user in the continuous space, *WiDeep* estimates the user location as the center of mass of all fingerprinting points [32], taking the probability of each reference point $P(l_i|x)$ as its weight. Hence, the user location l is estimated as

$$l = \sum_{i=1}^{N} p(l_i|x) l_i \quad (5)$$

[1]If the user location profile is known, it can be used directly in Equation 2.

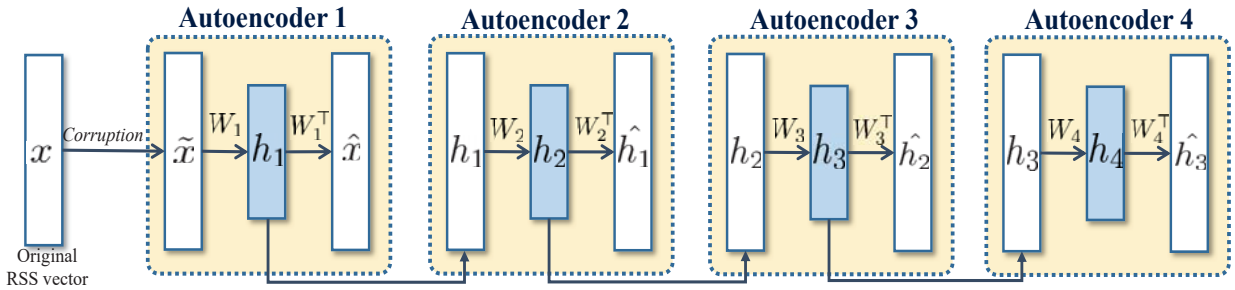


Fig. 4. Greedy layer-wise pre-training process. The latent vector of every trained layer is used to train the subsequent layer.

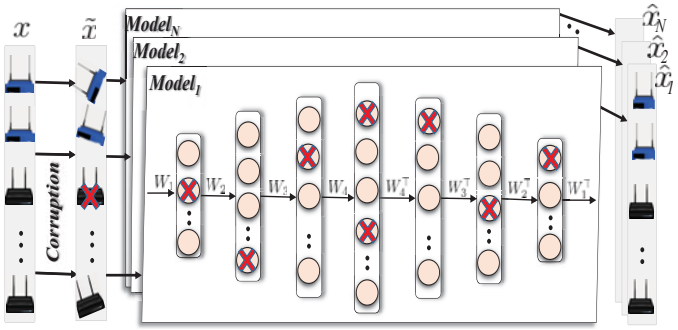


Fig. 5. Deep network architecture. Crossed nodes are example of dropped-out neurons. Note that the output of this network is connected to the probabilistic location inference module.

IV. Evaluation

In this section, we evaluate the performance of *WiDeep* in two typical indoor environments: a university building floor and an apartment. We start by describing the data collection methodology. Next, we analyze the effect of different parameters on the *WiDeep* system performance. Finally, we compare our system to the state-of-the-art WiFi fingerprinting techniques that are Horus [6] and DeepFi [8].

A. Data Collection

To collect the necessary data for evaluation, we deployed our system in two buildings with different layouts and APs densities (Table I). The first one is a floor of our university building with a $37m \times 17m$ area containing offices, labs, meeting rooms as well as corridors (Fig. 6). The second one, shown in Fig. 7, is an L-shaped private studio apartment with a $14.5m \times 4.5m$ area. In both datasets, we leverage the RSSs of pre-installed WiFi APs in the building or overheard from nearby floors/buildings (university floor has an overall of 122 APs whereas the apartment has 59 APs). 7200 samples in total are collected at 48 different locations in the university dataset. For the apartment dataset,2000 samples are collected at each point of 139 different locations. The data is collected by five participants using different Android phones (e.g., Samsung Galaxy Note 3, Samsung Galaxy S4, Huawei P9 lite, among others) over different days. This captures the time-variant nature of the WiFi fingerprints as well as the heterogeneity of users and devices.

TABLE I
Summary of used testbeds parameters.

Testbed	University	Apartment
Area	$37m \times 17m$	$14.5m \times 4.5m$
Number of APs	*122*	*59*
Density of APs (AP/m^2)	*0.19*	*1.05*
Training points	*29*	*81*
Testing points	*19*	*58*

We implemented a WiFi collector App using the Android SDK to scan APs. The program records the (MAC address, RSS, timestamp) for each heard WiFi AP. The scanning rate was set to one per second. We implemented our deep learning based training using the Google TensorFlow [33] framework on the Google Collaboratory Cloud[2]. 40% of the data points are held out for testing. We experimented with different deep learning architectures and the one of $200\times300\times400\times500$ obtains the best performance.

B. Effect of Changing WiDeep Parameters

In this section, we study the effect of the different parameters on the system performance including different techniques used to add noise to the input, dropout regularization, different values of radial basis function parameters used for probabilistic online localization, number of input scans used in estimation, and number of deep learning model layers. Table II shows the default parameters values used throughout the evaluation section.

1) Effect of different input noise corruption techniques: We experimented with two different input noising methods (Section III-B1): the masking method and additive Gaussian noise method. Fig. 8 shows the mean location error of *WiDeep* when trained using different masking corruption fraction values. Similarly, Fig. 9 shows the mean error of using different standard deviation values to train the models using the additive Gaussian method. The figures show that adding more variations, by injecting artificial noise, of the data input to the model enhances performance compared to the case when when the noise level is zero. However, adding too much noise distort the signal and increase the ambiguity between adjacent locations, hence increases the localization error. An optimal

[2]https://colab.research.google.com

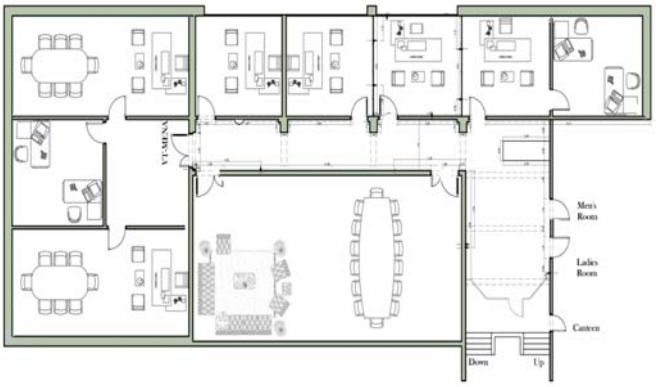


Fig. 6. University floorplan.

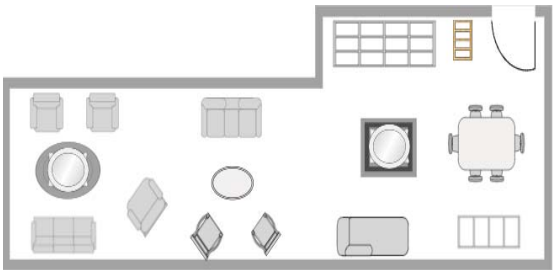


Fig. 7. Appartment floorplan.

noise level for both techniques can be achieved at masking probability $f = 0.1$ and $s = 0.1$.

2) Effect of dropout regularization: Fig. 10 shows the effect of the dropout regularization on the mean localization accuracy. The figure shows that as the dropout rate is increased, the mean localization accuracy is improved. This is aligned with our expectation because the regularization reduces the overfitting of the neural network to the input data. However, using a large dropout rate leads to a decrease in the localization accuracy as the network does not fit the data well (i.e., underfitting). Therefore, this trade-off is balanced at 0.5 dropout rate.

3) Effect of the number of scans per estimate: Fig. 11 shows the effect of increasing the number of the scans (n) used to estimate a location in the online phase. The figure shows that as n increases, the accuracy improves until it reaches an optimal value at $n = 7$ beyond which it begins to deteriorate. This is due to two opposing factors: (1) By increasing n, we get more information which is useful for location estimate, (2) However, as n increases, more time is spent to collect these samples which may involve crossing reference points boundaries. This has a negative effect on performance.

4) Effect of the λ parameter in the Radial Basis Function: Fig. 12 shows the effect of using different values of λ in the radial basis function on the localization error. The figure shows that there is an optimal value for λ at 4.

5) Effect of number of layers in the network: Fig. 13 shows the effect of changing the number of layers (stacked autoencoders). The figure shows that increasing the layer, increases the accuracy until reaching an optimal value at four layers. After that, the accuracy starts to decrease as the network begins to overfit the training data.

TABLE II
DEFAULT PARAMETERS.

Parameter	Range	Default value
Testbed	*University, Apartment*	*University*
Masking fraction (f)	*0 - 0.5*	*0.1*
Additive Gaussian Stdev (s)	*0 - 0.5*	*0.1*
Learning rate	*0.001 - 0.1*	*0.1*
Batch size	*16 - 2048*	*128*
Dropout rate (r)	*0 - 0.9*	*0.5*
Radial Basis Function parameter (λ)	*0.25 - 6*	*4*
Number of scans per estimate (n)	*1 - 30*	*7*
Number of epochs	*1 - 20000*	*10000*
Network architecture	*200×300×400×500*	

C. Robustness Experiments

In this section, we assess the robustness of *WiDeep* under different challenging scenarios including reducing the density of access points and reduced number of fingerprint locations.

1) Density of access points: Fig. 14 shows the effect of reducing the density of the access points on accuracy. For this, we uniformly removed access points from the total access points detected in the area. The figure shows that even with a density as low as 5% of the access points, *WiDeep* can achieve high accuracy of less than 2.8m mean error. This is due to the different noise-handling techniques as well as the used regularization techniques. This highlights the robustness of *WiDeep*.

2) Density of training points: Fig. 15 shows the performance *WiDeep* when the number of training fingerprint locations is reduced. The figure shows that, even though reducing the number of training points/percentage *linearly* leads to increasing the area associated with each training point *quadratically*, the decrease in the accuracy does not grow as fast. *WiDeep* can achieve a mean accuracy of 3.16m even with 50% reduction in training points.

D. Comparative Evaluation

In this section, we compare the location accuracy, robustness to heterogeneous devices, and runtime of the *WiDeep* system against two baseline systems. The first is a popular probabilistic fingerprinting based indoor localization technique (Horus [6]) that assumes the independence of the APs and the second is a recent deep-learning based indoor localization technique (DeepFi [8]) that does not perform noise handling or model over-fitting avoidance.

1) Localization accuracy: Fig. 16 and Fig. 17 show the CDF of distance error for all systems in the university and apartment testbeds. Table III and IV summarize the results. The figures illustrate that our *WiDeep* system can achieve

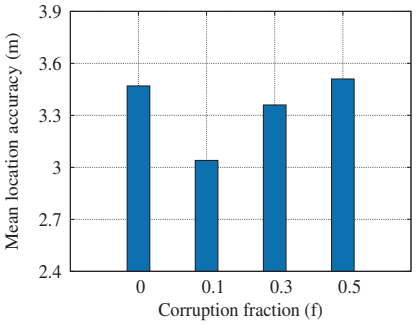


Fig. 8. Effect of the masking probability on accuracy.

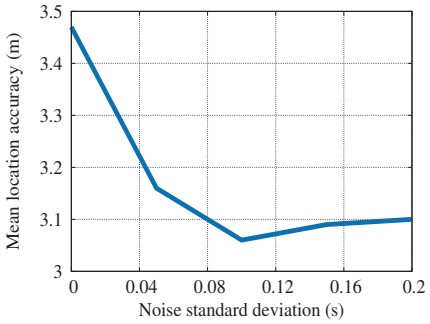


Fig. 9. Effect of the noise standard deviation on accuracy.

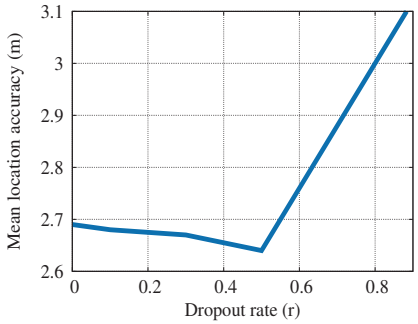


Fig. 10. Effect of the dropout regularization rate on accuracy.

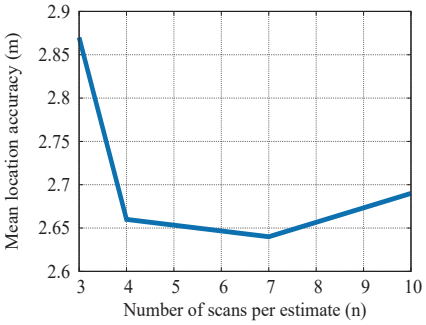


Fig. 11. Effect of the number of scans per estimate on the accuracy.

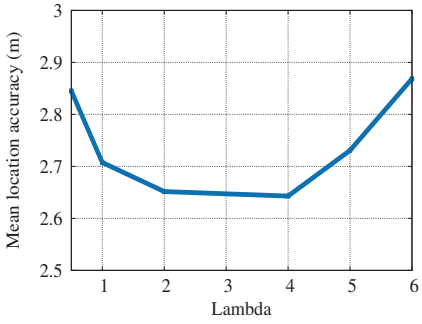


Fig. 12. Localization error vs. RBF parameterλ.

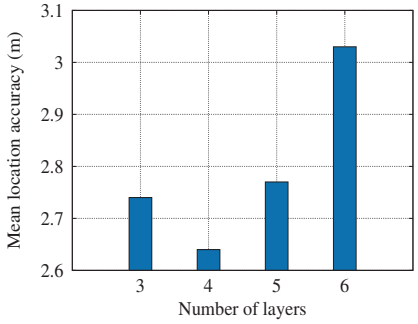


Fig. 13. Effect of the number of layers on accuracy.

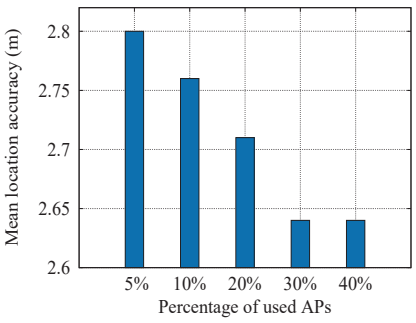


Fig. 14. Effect of density of APs on accuracy.

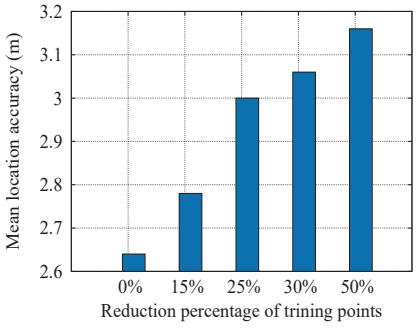


Fig. 15. Effect of reducing the number of training locations on accuracy.

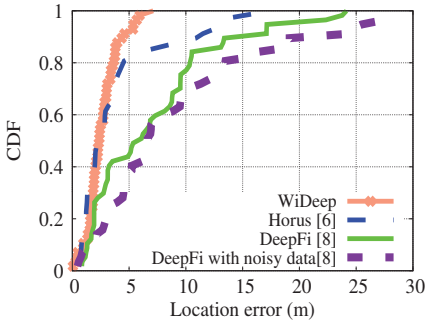


Fig. 16. Comparison of CDFs in the University testbed.

significantly better mean localization accuracy than the other systems by at least 29.8% and up to 169% in university and apartment testbeds, respectively. Moreover, *WiDeep* enhances all the other quantiles. This can be explained by noting that traditional probabilistic fingerprinting techniques such as Horus [6] cannot capture the correlation between the different APs and the fingerprinting locations. Similarly, traditional deep learning techniques such as DeepFi [8] do not take the inherent noise of the wireless signals into consideration nor avoid over-training. Therefore, their performance drops noticeably when trained with noisy data. This can be seen in figures 16 and 17, where the accuracy of DeepFi degrades in such scenarios while WiDeep maintains its accuracy.

Note also that *WiDeep* performance is consistent in the two testbeds, contrary to the other two techniques: DeepFi performs better in the testbed with more available data (i.e higher density of APs and training locations) while Horus can tolerate better the lower APs density and lower in the other testbed.

2) Device heterogeneity: Here, we evaluate the different techniques robustness to devices heterogeneity. Initially, *WiDeep* is trained and tested with the same device (i.e. Samsung Galaxy Note 3). We then carry out experiments by training the different systems with the Samsung Galaxy Note 3 tablet and testing with a Samsung S4 mini smartphone. The two devices have completely different form factors and WiFi

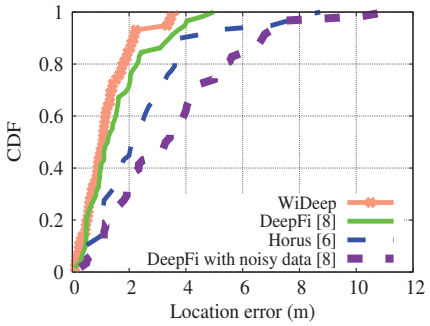


Fig. 17. Comparison of CDFs in the Apartment testbed.

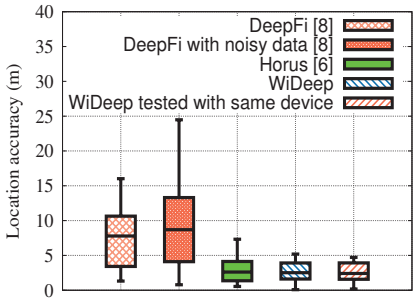


Fig. 18. Comparison based on device heterogeneity.

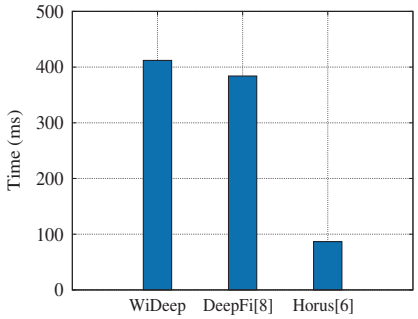


Fig. 19. Time per location estimate.

chips.

Fig. 18 shows that *WiDeep* provides approximately the same accuracy when testing with different device as when testing with the same device. It can also be seen from the figure that *WiDeep* has the best performance in handling device heterogeneity compared to the other two systems across all percentiles. This is due to the combination of additive noise in the training data and the adoption of denoising autoencoders which gives *WiDeep* greater flexibility than the other systems. In particular, this is true since device heterogeneity can be considered to be a form of noise, which the *WiDeep* network and training process are designed specifically to combat. Horus also shows better adaptability than DeepFi. This can be attributed to the fact that it utilizes probabilistic techniques, which are known to perform well in the presence of uncertainty or noise. On the other hand, DeepFi shows poor performance to noisy data because of the lack of specific provisions to handle such phenomena in its design.

3) Time per location estimate: Fig. 19 compares the running time per location estimate for the three techniques. The machine used for running the algorithms is an HP Omen laptop with an i7 2.6 GHz processor, 16 GB RAM, and a Nividia GTX965 GPU. The figure shows that the running time of *WiDeep* and DeepFi is comparable. Horus has the lowest running time per location estimate as its prediction is based on only Bayesian inference. Deep learning techniques, on the other hand, need to pass the data through all the layers of the network. Nonetheless, all techniques can estimate the user location in less than 412ms, which allows realtime tracking of the user. This can be further enhanced if needed through parallelization.

E. Discussion

WiDeep is designed to operate with heterogenous devices without compromising on localization accuracy. The use of a deep model alone (e.g as in DeepFi) cannot lead to this design goal. *WiDeep* is able to achieve this as a combination of the particular choice of deep network used and the associated design considerations. Specifically, the stacked denoising autoencoder network used in *WiDeep* is, by definition, capable of reconstructing the underlying input in the presence of noise or distortion. Therefore for the best results, the training process of this network necessitates the use of noisy data so that the network truly learns to extract the underlying information from the data as obtained from users' heterogeneous devices [26]. This enhances the generalization ability of the network in challenging scenarios, e.g. in the presence of device heterogeneity. At the same time, the use of noisy data alone with DeepFi (which does not use this type of network model) leads to a degradation of the obtained localization accuracy. Additionally, the dropout regularization of *WiDeep* ensures the quality of the final model by eliminating co-dependencies between the constituent neurons [28], [34].

It can be seen from Fig. 9 that training *WiDeep* without considering the noise injection process leads to a significant drop of the localization performance to 3.47m. Similarly, Fig. 10 shows that the accuracy degrades to 2.69m without penalizing the training process with such dropout regularization. Therefore, the combination of the network used and the regularization techniques are able to yield significant improvements over traditional deep learning models.

V. Related Work

In this section, we discuss the most relevant literature to our *WiDeep* system. In particular, we cover two categories: fingerprinting systems and crowd-sourcing systems.

A. Fingerprinting Systems

Fingerprinting systems present the most popular WiFi-based indoor localization technique due to their high accuracy. Those can be categorized into traditional and deep learning-based fingerprinting systems.

*1) **Traditional Fingerprinting Systems**:* Radar [13] fingerprint captures the average RSS of the heard APs at the different fingerprint locations. During the online phase, matching is based on using the k-nearest neighbors algorithm [35]. Deterministic approaches though cannot deal well with he noise and variations of the RF signal. To tackle the noisy nature of RSS, probabilistic techniques have been proposed, e.g. Horus [6], [36], [37]. In this case, the fingerprint reflects the RSS histogram for each AP at each reference location, assuming APs are *independent*. The most probable location

is estimated based on Bayesian inference. Many variants of probabilistic techniques have been proposed over the years to further enhance the localization performance [5], [7], [38]. For instance, [7] uses radial basis networks to predict the unknown location. Despite probabilistic techniques being able to handle the inherently noisy wireless signals in a better way than deterministic techniques, they usually assume that the signals from different APs are independent to avoid the curse of dimensionality problem [20]. This leads to coarse-grained accuracy.

WiDeep, on the contrary, harnesses a deep neural network that is able to learn dependencies between signals from different APs. In addition, it is designed to address the inherent noise in the RF signals. Moreover, it has provisions to handle over-fitting, leading to better robustness.

*2) **Deep Learning Systems**:* Recently, different deep learning techniques have been proposed in order to train models to provide a localization service. In DeepFi [8], [9], Restricted Boltzman Machines are used to pre-train a deep learning system. The localization service of DeepFi depends on the magnitudes of the channel state information (CSI) data, as compared to the standard received signal strength. Later, deep convolution networks based on CSI data also have been proposed to estimate the unknown locations [39], [40]. All these techniques use CSI data, which needs *special hardware* for collection, reducing the system ubiquity. In addition, they do not have provisions to reduce over-fitting or handle the inherent noise in the input data, reducing their robustness.

In contrast, the operation of *WiDeep* depends on standard RSS readings, which can be received by the common on-board WiFi radio present in all mobile devices using standard APIs in the operating system. In addition,*WiDeep* is designed to deal with noisy data and have provisions to avoid model over-fitting, both leading to higher accuracy and more robustness.

B. Crowdsourcing Systems

To reduce the fingerprint construction overhead, a number of systems have been introduced in which the users collaborate to improve the localization system by crowd-sourcing the fingerprint. [41] uses crowd-sourcing to improve the particle filter performance overtime and hence improve the localization accuracy of the system. Other systems, e.g. [16], [17], [21], [42]–[44], use the smartphone inertial sensors to calculate the user location using dead-reckoning and leverage different sensor-based landmarks, including WiFi, to reset the accumulated error. These system, however require additional sensors, which may not be available on all mobile devices, especially in development countries where low-end phones are more common. *WiDeep* can benefit from crowd-sourcing to construct its fingerprint in an automatic manner. In addition, based on deep learning and its different noise and robustness handling modules it can provide robust and high accuracy localization without the need of any additional sensors.

VI. Conclusion

We presented *WiDeep*, an accurate and robust WiFi fingerprinting indoor localization technique based on a deep neural network. The system leverages stacked denoising auto-encoders in a probabilistic framework to mitigate the noise in the RSS measurements. Additionally, it employs model regularization to enable the network to generalize and avoid over-fitting, leading to a more robust and stable models.

We evaluated *WiDeep* in two different challenging environments that represent a university building and a domestic apartment using different Android devices. The results show the *WiDeep* comes with a localization accuracy better than the state-of-the-art systems by at least 53% and 29.8% in the large and small environments respectively. Moreover, its performance is robust to different devices and different densities of APs in different environments.

TABLE III
Accuracy percentiles of different systems in the University floorplan

Technique	Average	50th Percentile	75th Percentile	100th Percentile
WiDeep	**2.64m**	**2.38m**	**3.38m**	**7.12m**
Horus [6]	4.04m (-53.03%)	2.25m (5.46%)	4.03m (-19.23%)	17.50m (-145.78%)
DeepFi [8]	7.10m (-168.93%)	6.09m (-155.88%)	9.54m (-182.24%)	24.14m (-239.04%)

TABLE IV
Accuracy percentiles of different systems in the apartment floorplan

Technique	Average	50th Percentile	75th Percentile	100th Percentile
WiDeep	**1.21m**	**1.07m**	**1.62m**	**3.74m**
DeepFi [8]	1.57m (-29.75%)	1.25m (-16.82%)	2.04m (-25.92%)	4.97 (-32.88%)
Horus [6]	2.57m (-112.39%)	2.17m (-102.08%)	3.35m (-106.79%)	8.73 (-133.42%)

Acknowledgment

This work has been supported in part by a grant from the Egyptian National Telecommunication Regulatory Authority (NTRA).

References

[1] P. Davidson and R. Piché, "A survey of selected indoor positioning methods for smartphones," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1347–1370, 2017.

[2] D. Lymberopoulos and J. Liu, "The microsoft indoor localization competition: Experiences and lessons learned," *IEEE Signal Processing Magazine*, vol. 34, no. 5, pp. 125–140, 2017.

[3] J.-P. Sheu, P.-C. Chen, and C.-S. Hsu, "A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement," *IEEE transactions on mobile computing*, vol. 7, no. 9, pp. 1110–1123, 2008.

[4] N. Lasla, M. F. Younis, A. Ouadjaout, and N. Badache, "An effective area-based localization algorithm for wireless networks," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2103–2118, 2015.

[5] M. Youssef and A. Agrawala, "Handling samples correlation in the Horus system," in *Proceedings of the International Conference on Computer Communications*, vol. 2. IEEE, 2004, pp. 1023–1031.

[6] ——, "The Horus WLAN location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 205–218.

[7] C. Laoudias, P. Kemppi, and C. G. Panayiotou, "Localization using radial basis function networks and signal strength fingerprints in WLAN," in *Proceedings of the International Conference on Global Communications*. IEEE, 2009, pp. 1–6.

[8] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," in *Proceedings of the International Conference on Wireless Communications and Networking*. IEEE, 2015, pp. 1666–1671.

[9] ——, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2017.

[10] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu, "Deep neural networks for wireless localization in indoor and outdoor environments," *Neurocomputing*, vol. 194, pp. 279–287, 2016.

[11] R. Elbakly and M. Youssef, "A calibration-free RF localization system," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015, p. 63.

[12] K. El-Kafrawy, M. Youssef, A. El-Keyi, and A. Naguib, "Propagation modeling for accurate indoor WLAN RSS-based localization," in *Proceedings of the Vehicular Technology Conference Fall (VTC)*. IEEE, 2010, pp. 1–5.

[13] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proceedings of the Nineteenth International Conference on Computer and Communications Societies*, vol. 2. IEEE, 2000, pp. 775–784.

[14] H. Abdel-Nasser, R. Samir, I. Sabek, and M. Youssef, "MonoPHY: Mono-stream-based device-free WLAN localization via physical layer information," in *Proceedings of the IEEE International Conference on Wireless communications and networking*, 2013, pp. 4546–4551.

[15] K. Habak, K. A. Harras, and M. Youssef, "Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey," *Computer Networks*, vol. 92, pp. 168–188, 2015.

[16] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 197–210.

[17] H. Abdelnasser, R. Mohamed, A. Elgohary, M. F. Alzantot, H. Wang, S. Sen, R. R. Choudhury, and M. Youssef, "SemanticSLAM: Using environment landmarks for unsupervised indoor localization," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1770–1782, 2016.

[18] M. Ibrahim, M. Torki, and M. ElNainay, "CNN based indoor localization using RSS time-series," in *Proceedings of the Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 01 044–01 049.

[19] M. Youssef, M. Abdallah, and A. Agrawala, "Multivariate analysis for probabilistic WLAN location determination systems," in *Proceedings of The Second Annual IEEE International Conference on Mobile and Ubiquitous Systems: Networking and Services*. IEEE, 2005, pp. 353–362.

[20] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[21] H. Aly and M. Youssef, "Dejavu: an accurate energy-efficient outdoor localization system," in *Proceedings of the 21st SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 154–163.

[22] M. Elhamshary and M. Youssef, "CheckInside: a fine-grained indoor location-based social network," in *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 607–618.

[23] M. Elhamshary, M. Youssef, A. Uchiyama, H. Yamaguchi, and T. Higashino, "TransitLabel: A crowd-sensing system for automatic labeling of transit stations semantics," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2016, pp. 193–206.

[24] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You are facing the Mona Lisa: spot localization using PHY layer information," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 183–196.

[25] Y. Kang, K.-T. Lee, J. Eun, S. E. Park, and S. Choi, "Stacked denoising autoencoders for face pose normalization," in *Proceedings of the International Conference on Neural Information Processing*. Springer, 2013, pp. 241–248.

[26] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[27] M. Youssef and A. K. Agrawala, "Small-scale compensation for WLAN location determination systems." in *Proceedings of the International Conference on Wireless Communications and Networking*. IEEE, 2003, pp. 1974–1978.

[28] H. Rizk, M. Torki, and M. Youssef, "CellinDeep: Robust and Accurate Cellular-based Indoor Localization via Deep Learning," *IEEE Sensors Journal*, 2018.

[29] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of Advances in neural information processing systems*, 2007, pp. 153–160.

[30] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[31] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[32] M. Youssef and A. Agrawala, "Continuous space estimation for WLAN location determination systems," in *Proceedings of 13th International Conference on Computer Communications and Networks, (ICCCN)*. IEEE, 2004, pp. 161–166.

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[34] A. Shokry, M. Torki, and M. Youssef, "DeepLoc: a ubiquitous accurate and low-overhead outdoor cellular localization system," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2018, pp. 339–348.

[35] H. Rizk, S. Elgokhy, and A. Sarhan, "A hybrid outlier detection algorithm based on partitioning clustering and density measures," in *Proceedings of the Tenth International Conference on Computer Engineering & Systems (ICCES)*. IEEE, 2015, pp. 175–181.

[36] M. A. Youssef, A. Agrawala, and A. U. Shankar, "Wlan location determination via clustering and probability distributions," in *Proceedings of the First International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2003, pp. 143–150.

[37] K. El-Kafrawy, M. Youssef, and A. El-Keyi, "Impact of the human motion on the variance of the received signal strength of wireless links," in *Proceedings of the 22nd Personal Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2011, pp. 1208–1212.

[38] M. Youssef and A. Agrawala, "Location-clustering techniques for wlan location determination systems," *International Journal of Computers and Applications*, vol. 28, no. 3, pp. 278–284, 2006.

[39] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional Neural Networks Based Indoor Wi-Fi Localization Using Channel State Information," *IEEE Access*, vol. 5, pp. 18 066–18 074, 2017.

[40] X. Wang, X. Wang, and S. Mao, "CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," in *Proceedings of the International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[41] Z. Liu, L. Zhang, Q. Liu, Y. Yin, L. Cheng, and R. Zimmermann, "Fusion of magnetic and visual sensors for indoor localization: Infrastructure-free and more effective," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 874–888, 2017.

[42] H. Aly, A. Basalamah, and M. Youssef, "Map++: A crowd-sensing system for automatic map semantics identification," in *Proceedings of the Eleventh Annual International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2014, pp. 546–554.

[43] N. Mohssen, R. Momtaz, H. Aly, and M. Youssef, "It's the human that matters: accurate user orientation estimation for mobile computing applications," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2014, pp. 70–79.

[44] H. Aly, A. Basalamah, and M. Youssef, "Lanequest: An accurate and energy-efficient lane detection system," in *Proceedings of the International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2015, pp. 163–171.