OmniCells: Cross-Device Cellular-based Indoor Location Tracking Using Deep Neural Networks

Hamada Rizk
Dept. of Comp. Sci. and Eng.,
E-JUST, Alexandria, Egypt
Tanta University, Tanta, Egypt
Osaka University, Osaka, Japan
hamada.rizk@ejust.edu.eg

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

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

***Abstract*—The demand for a ubiquitous and accurate indoor localization service is continuously growing. Cellular-based systems are a good candidate to provide such ubiquitous service due to their wide availability worldwide. One of the main barriers for accuracy is the large number of models of cell phones, which results in variations of the measured received signal strength (RSS), even at the same location and time. In this paper, we propose *OmniCells*, a deep learning-based system that leverages cellular measurements from one or more training devices to provide a consistent performance across unseen tracking phones. Specifically, *OmniCells* uses a novel approach to multi-task learning based on autoencoders that allows it to learn a rich and device-invariant RSS representation without any assumptions about the source or target devices. *OmniCells* also incorporates different modules to boost the system's accuracy with RSS relative difference-based features and to improve the deep model's generalization and robustness.**

Evaluation of *OmniCells* in two realistic testbeds using different Android phones with different form factors and cellular radio hardware shows that *OmniCells* can achieve a consistent median localization accuracy when tested on different phones. This is better than the state-of-the-art indoor cellular-based systems by at least 101%.

I. Introduction

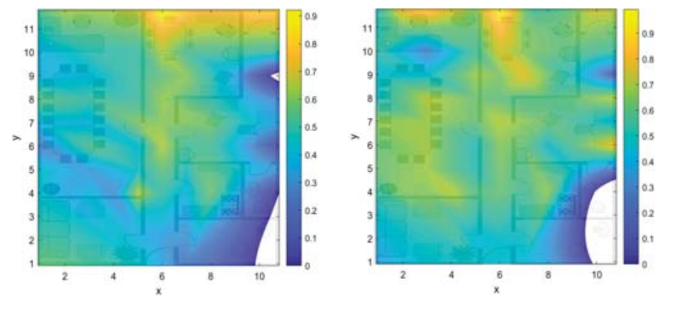
Recent years have witnessed an ever-growing demand for accurate and ubiquitous indoor positioning systems in many applications. Towards achieving this goal, WiFi-based indoor localization is widely adopted. WiFi-based indoor localization systems harness the Received Signal Strength (RSS) captured by the user's phone from WiFi access points as signature for the location identification, building a WiFi fingerprint database. Fingerprinting is a two-phase technique, consisting of an offline and an online phase. During the offline phase, cellular signals are captured at known points (i.e. fingerprints) in the area of interest where their strengths are used to characterize the corresponding locations. Subsequently, the collected fingerprints are used to build a localization model for estimating the user location in the online phase given a signal scan. The major challenge for fingerprint-based approaches is the extraction of robust and **discriminative signatures** across space and across **different devices**.

On the other hand, cellular based techniques have a number advantages that make them more attractive than their WiFi counterparts. First, cellular coverage far exceeds the coverage of WiFi networks as cell towers are dispersed with high density across the inhabited world. Second, all cell phones including low-end ones, by definition support cellular technology. Third, a cellular-based localization system will still work even with a failure in buildings' electrical infrastructure as cellular base stations are better equipped to tolerate power failures. Finally, network configuration changes rarely occur due to the consequent significant expense and complexity of this process when performed frequently. This leads to a ubiquitous and stable operating environment for localization systems, which does not involve tedious re-calibration.

Current cellular-based localization techniques [1]–[3] are designed to capture the fingerprint of the received signal strength from the different cell towers detectable in the area of interest. These fingerprints are then used to train a classifier that differentiates between different reference points in the area of interest. Different types of classifiers are proposed in literature, e.g. Support Vector Machines [2] and K Nearest Neighbors [3]. For better feature representation ability and localization performance, deep learning was adopted in [1].

A common assumption made by these techniques is that the distribution of the collected cell fingerprints is independent of the phone, such that a localization model trained on one device can be leveraged by other devices. In practice, however, different cellphones may in fact have different hardware/software specifications, leading to different distributions even when the user is stationary at the same location. As such, the quality of the localization models drops significantly in scenarios where the system is trained with a specific phone and tested with different types of phones, which is the typical real world use-case. To solve this problem without compromising on accuracy, a localization model should be trained for every possible consumer phone. However, this solution is intrinsically impractical due to the huge number of phones available on the market and the requirement to collect sufficient amounts of data using each individual phone to train a corresponding model.

To tackle the above problems, we propose *OmniCells*: a deep learning-based cellular localization system trained with data collected from only a single or few phones. It is capable of providing stable and accurate positioning even when used by



(a) HTC One X9 phone. (b) Motorola Moto G5 phone.

Fig. 1. Heatmaps of the normalized RSS received by different phones from an arbitrary cell tower in the area of interest.

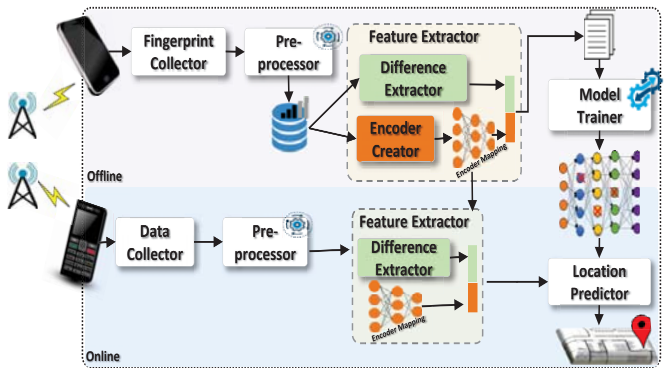


Fig. 2. *OmniCells* system architecture.

other unseen phones without requiring any information about the considered phones. This is made possible by extracting device invariant features, i.e. relative features from the data recorded by the calibration phone(s) during the offline phase. Towards this, we leverage autoencoder networks to transform the input distribution of the raw RSS space to a *latent space* where different phones' data are identically distributed. These features are then harnessed to train a deep neural network to map the transformed RSS data to the user's location. In the tracking phase, the transformed features will be extracted from the cell measurements reported by an unseen phone and then fed to the trained deep model to estimate the user location.

Evaluation is held in two different testbeds using different Android phones. We obtained results confirming that *OmniCells* performs equally when **trained and tested using different phones** with a consistent median location error of 1.67m and 2.05m in a small and a large testbed; respectively. This accuracy surpasses the accuracy of the state-of-the-art cellular-based techniques in the considered testbeds by more than 111.9% and 100.98%; respectively.

The rest of this paper is structured as follows. In Section II, we provide a brief description of the problem and an overview of the proposed system. Section III presents in detail the methodology proposed by *OmniCells*. In Section IV, we present the detailed evaluation of *OmniCells*. Sections V and VI discuss related work and conclude the paper respectively.

II. Problem Definition and System Overview

A. Problem Definition

The device heterogeneity problem can be observed from the heatmaps in Fig. 1. Data was collected by two different phones at the same locations from the same cell tower. The figure shows that the two phones has different heatmaps over the area of interest, which negatively affects the localization performance if one of the phones is used to construct the cellular fingerprint while the other is used for testing (as we quantify in the evaluation section IV-C1).

The problem can then be formalized as follows. Consider the training of a localization model with fingerprints collected by a calibration device (denoted as D_c), which will be used with a different tracking device (detonated as D_t). Each cellular scan from each device (x_c and x_t respectively) consists of a RSS vector from surrounding cell towers in the area of interest. The goal is to find a mapping function F that, when applied to scans collected at the same location (l), ensures that they have a similar distribution. More formally:

$$z_t = F(x_t), z_c = F(x_c) \ni P(z_c|l) \approx P(z_t|l) \qquad (1)$$

B. System Overview

The overall *OmniCells* architecture is shown in Fig. 2. *OmniCells* works in two stages: an offline training stage and online tracking stage. The offline stage has four main modules. It starts by recording cell information from the detectable cell towers through the use of the **Fingerprint Collector** module. The recorded data is then forwarded to the **Pre-processor** module which produces a low-level[1] vector of the received signal strengths from the different detectable cell towers. The RSS vectors are then further processed by the **Feature Extractor** module to extract relative features that are loosely-coupled to the cell phone. In addition, this module trains a model (i.e. encoder model) to encode the RSS vector and generate higher level device-invariant features as we describe in Section III-B. This helps the **Localization Model Constructor** module to optimally construct and train a deep model (i.e. localization model) that maps these discriminative features into fingerprint locations in a device-transparent manner. The output of this offline phase is two trained models (i.e. encoder model and localization model) which are saved for later use during the online phase.

During the online phase, the user carrying her phone at unknown location scans for cell towers information, which is then forwarded to the *OmniCells* server. This data is first handled by the **Pre-processor** module. Thereafter, the **Feature Extractor** module will extract the desired device-independent features. Finally, the **Location Predictor** module feeds the data to the localization model constructed in the offline phase to estimate the likelihood of the user being at the different reference points considered during the offline phase. The

[1]RSS vector is called low level vector due to its tight-coupling to the phone that is used in data collection.

system then fuses these different likelihoods to obtain the user location in a continuous manner in the space of interest.

III. The *OmniCells* System

This section presents the details of the different modules of *OmniCells* including the Pre-processor the Features Extractor and Localization Model Constructor modules as well as the processing done in online phase.

A. The Pre-processor Module

This module runs during both the calibration and tracking phases. Given that q cell towers can be heard in the area of interest through the different scans, this module produces a vector $x = (x_1, x_2, .., x_q)$ of length q, where each entry represents the RSS from a certain cell tower. As the number of cell towers that can be detected per scan is limited to seven per the cellular standard [4], non-heard cell towers in an arbitrary scan are assigned a RSS of 0 Arbitrary Strength Unit (ASU). Thereafter, the input RSS values ranges are normalized to be in the range between [0,1] for each cell tower. Cell towers with weak received signals, i.e. close to the sensitivity of the receiving device antenna [2] may be detected only by the calibration phone leading to unstable localization. To handle this behaviour, the *Pre-processor* module switches off (i.e sets to zero) cell towers whose reported RSS value is below a certain threshold, mimicking that this cell tower has not been heard. This can be considered as a filtering operation, which is aimed at removing fluctuating readings that could negatively affect the system performance. In order to mitigate the high overhead of collecting massive amount of data for training deep models, *OmniCells* employs the data augmentation framework proposed in [5]. The framework generates synthetic data from samples collected over a short-term that reflect the typical RSS variation measurements. This has an additional advantage of combating possible overfitting which may occur due to training with a small amount of data.

B. The Features Extractor Module

OmniCells performs features extraction through two different sub-modules: the *Difference Extractor* and the *Encoder Creator*.

1) Difference Extractor: This module calculates the difference of RSSs between pairs of cell towers of each scan to mitigate the effect of device heterogeneity. The intuition behind this method is that different devices lead to a fixed offset value to the readings of all heard towers depending on the receiver sensitivity and gain. Thus, this added value can be eliminated through the use of relative RSS values i.e. a difference operation. This has been confirmed in literature, e.g. [6].

Given a pre-processed cell scan $x = (x_1, x_2, .., x_q)$, *OmniCells* calculates the difference $\triangle x_{ij}$ between the RSS values of every pair of cell towers x_i and x_j such that $\triangle x_{ij} = x_i - x_j$. A total of $\binom{q}{2}$ differences are calculated for each scan, which represent a new feature vector which can be expressed as: $\triangle x = (\triangle x_{12}, ..., \triangle x_{(q-1)q})$.

[2]The receiver sensitivity refers to the minimum level of signal strength can be detected by the receiver's antenna.

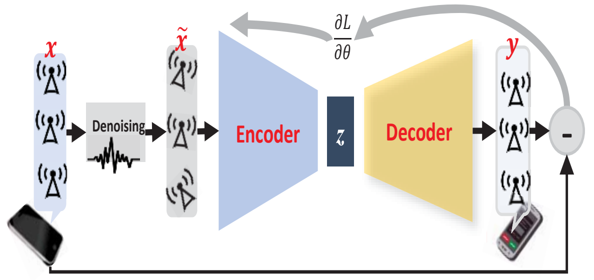


Fig. 3. Architecture of deep autoencoder when only a single device is available in the calibration phase.

2) Encoder Creator: This module is responsible for encoding/mapping the pre-processed (i.e. fixed-length, normalized and filtered) RSS vector to get a consistent representation of the signals across different phones. This can be achieved through the use of an autoencoder neural network (AE) [7]. This network consists of two main parts: an encoder and a decoder. The encoder part is responsible for encoding the input by mapping it into a lower-dimensional latent-space. Formally speaking, the encoder can be considered as an encoding function $z = F(x)$ applied on the RSS input vector x. Conversely, the decoder part is trained to reconstruct the input from that latent representation. Training an autoencoder to reconstruct the original input from such low dimensional space forces the autoencoder to learn the main features of the input.

In the canonical use case, autoencoders are used to reconstruct the original input data. In this work however, we train the autoencoder using data collected by an arbitrary phone to reconstruct data collected at the same time and location by a different phone (or a noisy version of the same phone). The intuition behind this method is that, in the encoding step, the encoder maps the information from the source phone to the latent representation. In the decoding step, the decoder must reconstruct the second phone's data from the latent representation. This forces the network to learn a latent representation which is generic (i.e. phone-invariant) and captures the underlying latent factors common to both RSS vectors. It is these phone-invariant factors which may then be considered to enable device-transparent localization.

Modes of operation: The training process could operate in two modes based on the number of phones available in the offline phase.

Single-phone Mode: Here, a single phone is used for data collection. The effect of phone diversity on the RSS readings can be modeled as an additive random noise [5]. Therefore, *OmniCells* emulates this effect by adding noise (i.e. White Gaussian noise) to the original signal, which leads to a distorted version of the data as:

$$\tilde{x} \sim x + \mathcal{N}(0, s^2) \quad (2)$$

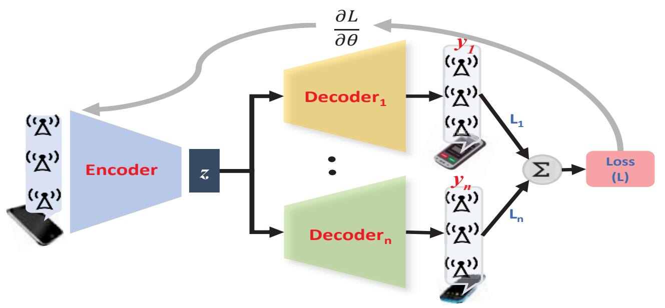


Fig. 4. Architecture of deep autoencoder when multiple devices are available in the calibration phase.

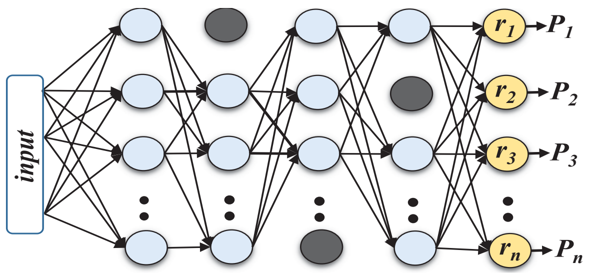


Fig. 5. Neural network structure for the autoencoders. Grey-shaded neurons represent examples of neurons that have been temporary dropped-off to increase the model robustness and avoid over-training.

Where x is the original RSS measurement for each cell tower and s is the standard deviation of the added noise.

The distorted version is then used to train a model to reconstruct the noise-free version. This can be done using a deep denoising autoencoder model.

Specifically, the denoising autoencoder shown in Fig 3 is trained by first corrupting the input RSS vector x to obtain vector $\tilde{x}$. Learning is accomplished by compressing the noisy input to a latent space z from which the the output (y) of the autoencoder is reconstructed and ensured to match the original *uncorrupted* vector x. By using a noisy version of the input, the autoencoder is forced to learn and represent the salient (i.e. device-invariant) features of the input data in the latent space. Training is performed by selecting the weights that minimize the mean square error (i.e. the loss function) between the original input data x and the reconstructed data y.

$$\mathcal{L}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{3}$$

Multiple-phones mode: In this case, a few phones are used for training. Therefore, *OmniCells* employs multi-task learning, which aims to optimize the encoder-decoder model with respect to multiple objectives. More specifically, we have multiple decoders stacked on top of a single encoder; where each target training device has its own decoder as shown in Fig 4. This ensures the network learns a more general encoder (and by extension, a more general latent representation) to accommodate diversity between different phones.

In this mode, we select the reconstruction error as the inverse correlation[3] between the reconstructions of the different phones and the original input vector as:

$$\mathcal{L}(x, y) = 1 - \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{4}$$

where $\overline{x}$ and $\overline{y}$ are the mean of the original RSS vector and the mean of the reconstructed vector across different samples belonging to the same fingerprint point. n is the number of the fingerprint point in the area of interest.

[3]We use inverse correlation loss as we found empirically that it leads to the best results.

3) Feature Aggregator: The trained encoder is used to extract latent features (z), which are concatenated with the differences features $(\triangle)$ calculated in Section III-B1 to get a combined feature vector (c). These features are then leveraged by the *Localization Model Constructor* module to train a deep learning-based localization model in the offline phase.

C. The Localization Model Constructor

This module is responsible for leveraging the aggregated features (c) to train a deep localization model and find its optimal parameters. The trained model is used during the online phase by the *Location Predictor* module to provide an estimate for the user location. A deep fully-connected neural network is adopted here due to its representational ability, which allows learning of complex patterns [1].

1) The network architecture: Fig. 5 shows our deep network structure. We construct a deep fully connected neural network consisting of cascaded hidden layers of nonlinear processing neurons. Specifically, we use the hyperbolic tangent function (tanh) as the activation function for the hidden layers due to its non-linearity, differentiability (i.e. having stronger gradients and avoiding bias in the gradients), and consideration of negative and positive inputs [8]. The input layer of the network is a vector of length k representing the combination of the latent features and the inter-difference values the cellular signal strength received from the q towers in the area of interest. The output layer consists of a number of neurons corresponding to the number of surveyed fingerprint points at the data collection time. This network is trained to operate as a multinomial (multi-class) classifier by leveraging a softmax activation function in the output layer. This leads to a probability distribution over the reference fingerprint locations given an input scan.

More formally, each cell scan $x_i = (x_{i1}, x_{i2}, ..x_{iq})$ before is mapped to a feature vector $c_i = (c_{i1}, c_{i2}, ..c_{ik})$ of length k. The corresponding discrete outputs (i.e logits) c_i is $a_i = (a_{i1}, a_{i2}, .., a_{in})$ capture the score for each reference points from the possible n reference points to be the estimated point. The softmax function converts the logit score a_{ij} (for sample i to be at reference point j) into a probability as:

$$p(a_{ij}) = \frac{e^{a_{ij}}}{\sum_{j=1}^{j=q} e^{a_{ij}}} \tag{5}$$

During the offline phase, the ground-truth probability label vector $P(a_i) = [p(a_{i1}), p(a_{i2})...p(a_{in})]$ is formalized using one-hot-encoding. This encoding has a probability of one for the correct reference point and zeros for others.

The model is trained using the Adaptive Moment Estimation (Adam optimizer [9]) to minimize the average cross-entropy between the estimated output probability distribution $P(a_i)$ and the one-hot-encoded vector g_i. The loss function is defined as follows:

$$\mathcal{L} = \frac{1}{N_s} \sum_{i=1}^{n} D(P(a_i), g_i) \tag{6}$$

where $P(a_i)$ is obtained using the softmax function, g_i is the one-hot encoded vector of the i^{th} sample, N_s is the number of samples available for training, and $D(P(a_i), g_i)$ is the cross-entropy distance function defined as:

$$D(P(a_i), g_i) = -\sum_{j=1}^{n} g_{ij} log(P(a_{ij})) \tag{7}$$

2) Preventing over-fitting: To increase the model robustness and further reduce over-fitting, *OmniCells* employs two regularization techniques: First, we use **dropout regularization** [10] which has been shown to be useful for the training of deep networks. During training, this method can be seen to sample from a large number of neural networks with different architectures in parallel. This is achieved by stochastically excluding (i.e. dropping out) some neuronal units from each layer in the network as well as their corresponding connections (Fig. 5). In effect, each epoch in training, each layer is updated with a different "view" of the configured layer. Dropout has the effect of making the training process noisy, forcing units within every layer to stochastically take on more or less responsibility for the inputs. Therefore, it prevents the neuronal units from co-relying on each other during training, in turn making the model more robust to unseen data and less likely to overfit the training data. We provide an experiment to show the effect of dropout on the results in Section IV-B2.

OmniCells further adopts **early stopping** as a second regularization method so that the training process would automatically stop at the point when the performance improvements are no longer gained [11].

D. Online Phase

The goal of this phase is to locate the user in real-time using the received cell signals from the nearby towers in the area of interest. This can be done by processing the scanned cells information and extracting the corresponding feature vector as described previously. Thereafter, this vector is then fed to the trained localization model to get a location estimate as one of the fingerprint points, defined at the calibration phase. The point r^* with the maximum probability given the feature vector (c) is selected as the estimated location. That is, we want to find:

$$r^* = \underset{r}{\operatorname{argmax}}[P(r|c)] \tag{8}$$

However, the main challenge here is that the built model by the *Localization Model* can predict the user locations and only at few discrete locations. As such, the estimated locations, even with a very accurate localization model, will be spaced out leading to a bad user experience. Therefore, this phase aims to track the user in the continuous spatial space (i.e. anywhere even on locations different from the reference points). To do so, *OmniCells* reports the center of mass of all reference points, i.e. by applying a spatial weighted average over the reference points, where the weights of each point are chosen as their corresponding likelihood as output from the classifier network [12]. More formally:

$$l_x = \frac{\sum_{i=1}^{n} P_i r_{ix}}{\sum_{i=1}^{n} P_i}, l_y = \frac{\sum_{i=1}^{n} P_i r_{iy}}{\sum_{i=1}^{n} P_i} \tag{9}$$

where r_{ix} and r_{iy} are the spatial coordinates of reference point i, and P_i is its corresponding softmax likelihood.

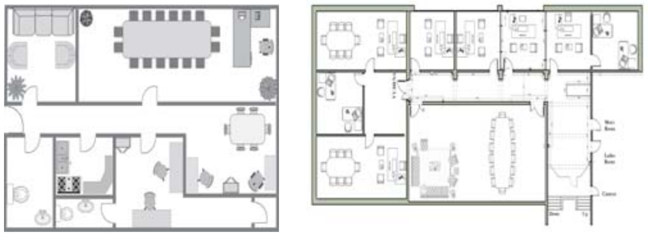

(a) Apartment layout. (b) Campus Building Floor layout.

Fig. 6. Layout of the considered testbeds.

TABLE I
SUMMARY OF THE TESTBEDS CONSIDERED IN EVALUATING *OmniCells*.

Criteria	Apartment	Floor
Area (m^2)	11×12	17×37
Number of fingerprint points	55	310
Spacing of seed points (m)	1	1.5
Number of cell towers	15	37

IV. Evaluation

In this section, we evaluate the *OmniCells* system in two real-world indoor testbeds as summarized in Table I. The first one (denoted as Apartment) is an apartment of 132m^2 area (Fig. 6(a)). The second one (denoted as Campus), shown in Fig. 6(b), is a floor in our university campus (in a different city) with a 629m^2 area containing several labs with different sizes and furniture placements, meeting room, offices as well as corridors.

We start by describing the data collection setup and software used. Next, we show how the system performs by varying the different system parameters. Finally, we compare the performance of *OmniCells* to five state-of-the-art localization systems.

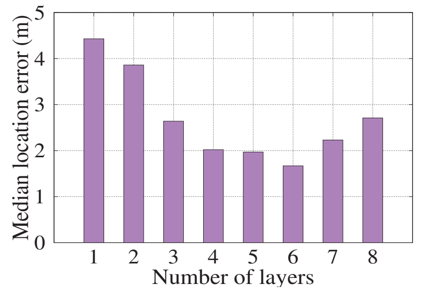


Fig. 7. Effect of changing the number of layers on *OmniCells* accuracy.

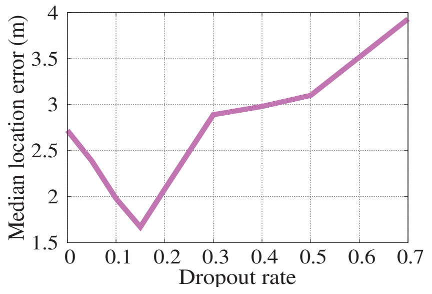


Fig. 8. Effect of the dropout percentage on *OmniCells* accuracy.

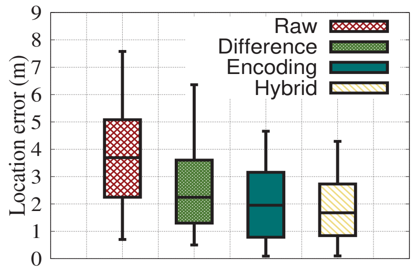


Fig. 9. Effect of feature extraction methods on *OmniCells* accuracy.

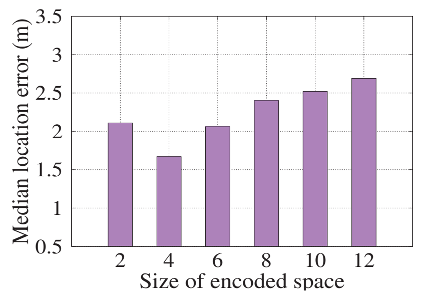


Fig. 10. Effect of changing the latent space size on *OmniCells* accuracy.

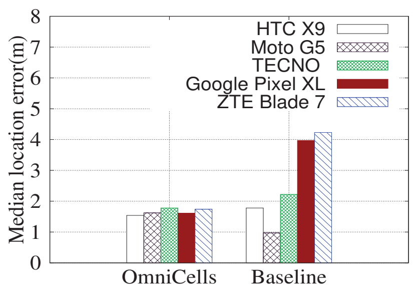


Fig. 11. Effect of varying the testing devices on accuracy while fixing the training devices (HTC X9, Moto G5 and Phantom 6).

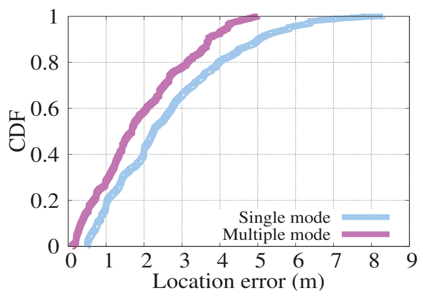


Fig. 12. Effect of the feature extraction modes on accuracy.

A. *Collection Setup and Tools*

Data is collected with an Android application. The application continuously scans for the nearby cell towers in the area of interest and records their different cell information including cell tower identifier (CID), location area code (LAC) and the corresponding timestamped signal strength. To facilitate ground-truth profiling, the same application runs synchronously on all mobile devices with one device dedicated to control ground-truth collection for all devices. The application visual interface is designed to depict the testbed floorplan in the foreground of the master device. The user tags her current location on the displayed testbed as a ground-truth which is triggered by a long tap on the map interface.

The data was collected using different Android phones, with different form factors and cellular chip placement, including HTC One X9, Google Pixel XL, Motorola Moto G5, Tecno Phantom 6, and ZTE Blade 7.

Data was collected at different reference points uniformly covering the area of interest. In the Apartment testbed, 55 different reference locations are annotated with 1m spacing. For the Campus testbed, we considered 310 reference locations with an inter-distance of 1.5m.

We implemented our deep learning based training using the Google TensorFlow framework. Training was carried out on the Google collaboratory cloud platform.

B. *Effect of Different System Parameters*

In this section, we study the effect of the deep models' different hyper-parameters and system parameters on the system performance including the number of layers, the dropout percentage, the effect of the feature extraction methods, and the size of the latent space. These experiments are accomplished when the HTC One X9, Motorola Moto G5 and Tecno Phantom 6 are used in the training for the multi-phone mode (Mode 2 in Section III-B2) of feature extraction while the remaining two cellphones (Google Pixel XL and ZTE Blade 7) are dedicated for testing. We obtained similar results when we changed the sets of training and testing phones.

TABLE II
DEFAULT PARAMETERS VALUES USED IN EVALUATION.

Parameter	**Range**	**Default**
Learning rate	0.0001 - 0.2	0.001
Dropout rate (%)	0 - 90	15
Early stopping patience	1-100	50
Number of hidden Neurons	20 - 1000	570
Number of layers	2 - 30	6
Number of samples per location	20 - 3000	3000
Training devices	HTC X9, Moto G5 and Phantom 6	
Testing devices	Pixel XL and ZTE Blade 7	

In the following subsections, we show the effect of varying these parameters only on the first testbed for clarity of presentation. We report the optimal parameters obtained for both testbeds in Table II. On the other hand, we present how *OmniCells* performs in both testbeds in Subsection IV-D1.

1) Number of Layers: Figure 7 shows the effect of changing the number of layers on *OmniCells* performance. The figure shows that increasing the number of layers increases the accuracy, due to increased model capacity, until it reaches six

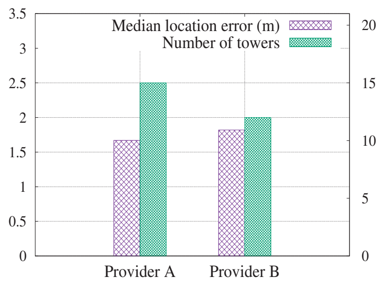


Fig. 13. Effect of changing the provider on *OmniCells* accuracy.

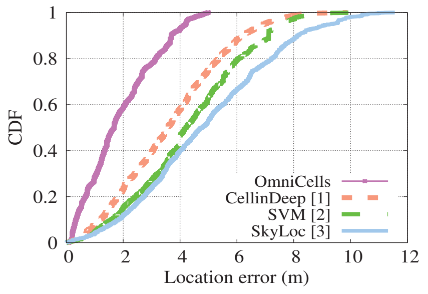


Fig. 14. Comparison of CDFs in the Apartment testbed.

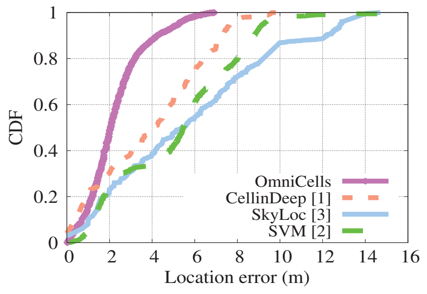


Fig. 15. Comparison of CDFs in the Campus testbed.

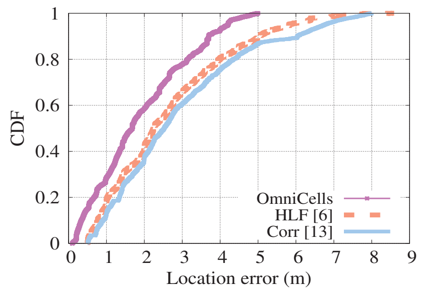


Fig. 16. Comparison of CDFs in the Apartment testbed.

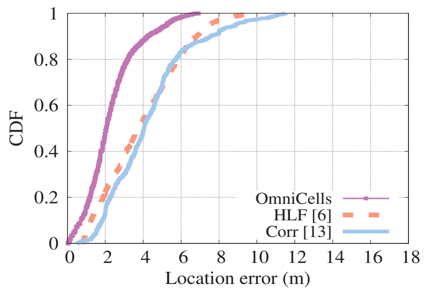


Fig. 17. Comparison of CDFs in the Campus testbed.

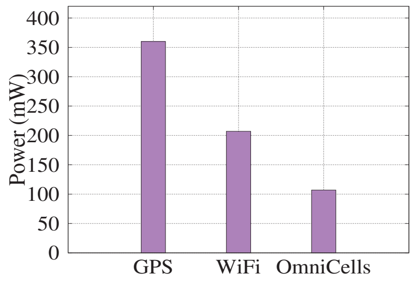


Fig. 18. The power consumption profile of different technologies.

layers. After that, the model tends to overfit the training data, leading to reduced accuracy. Therefore, we choose six layers as the default number of layers in our model.

2) Dropout Percentage: Fig. 8 shows how the dropout percentage affects *OmniCells* performance. The figure shows that an optimal value is achieved at 15% dropout percentage which enhances the system accuracy by 38% compared to the case of no dropout (dropout percentage = 0). This is due to the robustness against overfitting gained by the model when using dropout as discussed in section III-C2. However, beyond 15% dropout percentage, the model tends to under-fit the training data.

3) Performance of the feature extraction module: As discussed in section III-B, the *Feature Extractor* module consists of two sub-modules: *Difference Extractor* and *Encoder*. In this section, we study the influence of each sub-module on the overall system performance. Fig. 9 shows boxplots of the localization error of *OmniCells* when using the raw RSS vectors, latent space features, relative difference features, or both. The figure shows that the combination of the two proposed methods gives an improvement in median error of 120%, compared to the baseline case of using raw RSS. This confirms the importance of the proposed features in capturing device-independent signal characteristics.

4) Effect of latent dimension size: Fig. 10 further shows the median localization accuracy when varying the dimensionality (i.e. size) of the autoencoder's latent space z. The figure shows that a latent space of four dimentions gives the best perfornace. Beyond that, the autoencoder tends to include undesirable (i.e phone-specific) information in the encoding, which leads to a drop in accuracy.

C. Robustness Experiments

1) Performance in diversity : In this section we evaluate the performance of the *OmniCells* system when tested with two different phones individually and compare the results to the devices used in training. For training purposes we leverage a HTC X9 cellphone as a source for the autoencoder while the Motorola Moto G5 and Tecno Phantom 6 are leveraged as the targets for the reconstruction process. We use a Google Pixel XL cellphone and ZTE Blade 7 as unseen test devices and compare the result of each to the result when using the calibration phones seen during the training process as test.

Fig. 11 shows the performance when tested using all the considered phones. As can be seen from the figure, *OmniCells* provides a consistent median localization error for the different phones: it obtains a median error of 1.61m and 1.74m when tested with Pixel XL and ZTE Blade 7 (unseen phones) respectively, while obtaining approximately the same accuracy when tested with the calibration devices (1.54m, 1.62m, and 1.78m when tested with HTC One X9, Motorola Moto G5 and Tecno Phantom 6, respectively). On the other hand, the unseen phones show massive error in the baseline system, that uses the raw, RSS as it is not designed to handle heterogeneity. These results confirm the robustness of the system performance with different unseen devices.

2) Performance in different modes of feature extraction: In this section, we investigate the performance of *OmniCells* in the two modes of the autoencoder: the single-phone and multi-phone mode (Section III-B2). Fig.12 compares the two modes and shows, as expected, that the multiple devices mode obtains better results due to the proposed technique of defining multiple decoders for the same encoder. This leads to more general and device-independent encoding at the bottleneck of the network. Nonetheless, both modes of operation lead to consistent tracking accuracy for unseen modes, with a slight advantage to the mutli-phone mode, if available.

3) Different providers: Here, we quantify the performance of *OmniCells* when tested with two different cellular service providers at the same testbed (i.e Apartment). The providers cover the area of interest with a density of 15 and 12 for providers A and B, respectively. Fig. 13 shows that *OmniCells* performance is correlated with the cell tower density of each provider: the higher the density, the better the localization accuracy.

D. Comparative Evaluation

In this section, we compare the performance of *OmniCells* to localization systems in literature with respect to localization accuracy and power consumption. Specifically, we compare the performance of *OmniCells* to five state-of-the-art systems. Three of them are cellular-based localization systems while the remaining two are well-known device-invariant-based localization systems. All techniques are tested using the default set of calibration and testing devices listed in Table II.

CellinDeep [1] adopts a multinomial deep classification model to estimate the user's location based on raw cell measurements from the detectable towers in the area of interest. Similarly, based on the raw RSSs from the different towers, SkyLoc [3] uses a K-nearest neighbor (KNN) classifier while [2] (denoted as SVM) harnesses a one-vs-all SVM classifier for the signals to location mapping. Note that all techniques are evaluated using cell readings received by two holdout devices (i.e. Google Pixel XL and ZTE Blade 7), which are not considered in the training phase. The technique in [13], denoted as Corr, builds a transformation model between the testing and the calibration devices with correlation-based approximation of the user location. The second technique, called hyperbolic location fingerprinting (HLF) [6] harnesses signal strength ratios between pairs of RSSs as fingerprints for the user location instead of absolute device-dependant values.

1) Localization accuracy: Figures 14 and 15 show the CDF of localization error for the different techniques in the two testbeds. Fig. 14 shows that *OmniCells* gives an improvement in median error obtained in the Apartment testbed of 111.90%, 183.33% and 150.59% compared to the CellinDeep [1], SkyLoc [3] and SVM [2] systems, respectively. Similalrly, for the second testbed, *OmniCells* outperforms CellinDeep [1], SkyLoc [3] and SVM [2] systems by 100.98%, 164.39% and 166.41%, respectively. In summary *OmniCells* improves upon the other techniques in both considered testbeds when tested with unseen devices (real scenario) as it handles the device-heterogeneity by the transformation of the raw RSS values into a more general space in which device-specific components are reduced.

we also compare the accuracy of *OmniCells* to Corr [13] and HLF [6] which are originally designed to tackle the device heterogeneity problem in ***WiFi***. Figures 16 and 17 show the CDF of distance error for the three techniques in the different testbeds with same set of testing devices. The figures show the superiority of *OmniCells* over the Corr [13] and HLF [6] techniques with 50% and 33.33% in the Apartment testbed and with 100% and 84.39% in the Campus testbed, respectively. This can be explained by noting that these compared schemes assume a linear mapping between measurements from different phones, which is not generally justified. *OmniCells* on the other hand, leverages deep learning techniques to learn general device-independent latent features, leading to better accuracy.

2) Power consumption: In this section, we quantify the power consumption of *OmniCells* as compared to other localization systems that use other sensors on the phone such as WiFi and GPS. For this, we utilize the PowerTutor [14] application. The measurements were taken over the course of an hour using the X9 phone. Fig. 18 shows that *OmniCells* has a remarkably lower power consumption profile with 93.45% and 236.4% savings in power compared to that of the WiFi- and GPS-based solutions respectively. Note that unlike WiFi and GPS, cellular service is running by default during the normal phone operation. Therefore, *OmniCells* practically consumes zero extra sensing power in addition to the standard phone operation.

V. Related Work

In this section, we discuss the most relevant literature to our *OmniCells* system.

A. Cellular-based Techniques

Cellular-based localization systems have been adopted for both outdoor and indoor use cases. This is due to the fact that cellular technology has the most wide-spread infrastructure around the world and is supported by the vast majority of mobile devices. Cellular-based systems have two different approaches in operation. The first approach, *Cell-ID based* [15], considers the user location as the location of the strongest heard cell tower. Despite its simplicity, this approach is infeasible for indoor scenarios due to its relatively high error.

The most commonly used cellular-based approach is called the **signal strength-based** approach. Its basic idea is to capture the relationship between the cell signals received by the user's end-phones and their distance from the transmitting cell towers [1]–[3], [7], [16]–[23]. Signal strength-based systems can be classified based on the solution into probabilistic or machine learning systems.

Probabilistic solutions [16], [17] are usually vary capable in tackling the noise associated with the received signals. For simplicity, these techniques estimate the location that maximize the probability of the received signals from different cell towers. However, this solution usually ignores dependency between the received measurements from different cell towers, i.e. doesn't consider the rich and correlated relation between the different towers, to avoid the curse of dimensionality [24].

This leads to remarkable information loss. To address this issue, different machine learning-based systems were proposed [1]–[3], [7], [18], [19], [25] to learn such dependency. These systems train models for estimating the user location based on using raw received signals. More specifically, these models can be designed to work as classifiers to discriminate between different points or regressors to interpolate or extrapolate the user location. This trained model can be then queried to obtain user locations as desired. SkyLoc [3] builds a K-nearest neighbor classifier, while the system in [2] leverages a Bayesian filter on top of one-vs-all SVM classifier. [1] learn such dependency through the efficient use of deep neural networks to discriminate between different locations and provide smoothed location estimates through the use of different modules. The commonality between these systems is their reliance on the raw features (i.e. received signal strength of information from the towers) and ignorance of the fact that different devices have different distributions over their sensed signals. Therefore, location services offered by these systems have severely degraded performance in real scenarios involving phones unseen during the training of the localization model (as we quantified in Section IV-D).

OmniCells, on the contrary, extracts general features that represent inter-tower relative differences with an encoded latent representation to train a localization model. Additionally, the localization model is ensured to generalize and avoid over-fitting through the use of different regularization methods.

B. Deep Learning-based Techniques

Recently, different deep-learning based fingerprinting techniques have been proposed [12], [26]–[34] in indoor environments. Several systems were proposed based on channel state information (CSI) received from WiFi transmitters. The CSI amplitude values are leveraged to train a deep autoencoder in DeepFi [28]. PhaseFi [35] harnesses the CSI phase to train autoencoders. While in BiLoc [36] same network architecture is considered with a bimodal tensor of amplitude and phase of CSI. On the other hand, the received signal strength (RSS) data from different access points is used to train WiDeep [26] that improves the localization performance by combining stacked denoising autoencoders (i.e. a model for each fingerprint point) and a probabilistic framework. The above motioned systems build a model for each reference point which affect the system's scalability in large environments and the performance in real-time applications. AutLoc [26] proposed a system that combines a autoencoder with ensemble learning of different techniques including random forest regression, multi-layer perceptron, regression and multi-layer perceptron classification. In [12], a multi-label classifier is used with auto-encoder to locate the user floor and location. Although autoencoders have been frequently adopted in this domain, the commonality between the different approaches is the fact that they use the same input data (i.e. having same distribution) as a source and destination for training the autoencoder models, without making allowances for device heterogeneity.

Different from these techniques, OmniCells introduces a encoder-decoder model that is designed to extract general latent features along with relative differences to train a localization model.

C. Heterogeneity Handling Techniques

Several techniques have been proposed to handle the device heterogeneity problem in WiFi-based localization. They can be classified into two main classes: transformation-based methods [13], [37] and feature extraction methods [6], [17], [38]. The transformation-based methods [13], [37] aim to construct a transformation function between every possible user device and the calibration device. This transformation function can be precisely built offline, but this approach also faces a scalability problem due to the enormous number of consumer devices. Building this transformation online doesn't compete favorably neither in localization accuracy nor in time per location estimate [39].

On the other hand, feature extraction methods aim to extract some robust features instead of the absolute RSS features. For instance, the method in [38] stores the inter-transmitter (e.g. access point) difference as the fingerprint to train a traditional machine learning model (e.g. KNN). Similarly in [17], the method harnesses signal strength ratios as fingerprints for probabilistic model. These methods are much more scalable than transformation methods as they don't require any samples from the target unseen devices to function. However, they generate high dimensional feature space with many misleading features that may negatively impact the adopted model.

In contrast, the proposed method aims to encode the cellular data to obtain a robust representation for the cellular data in a lower dimensional space. This can be achieved without jeopardizing the localization accuracy.

VI. Conclusions

We proposed *OmniCells*, a deep learning system for indoor localization designed to be combat the hardware diversity problem in the clients' devices. We presented the details of the system and its ability to extract the core features, in different calibration scenarios (e.g. multiple phone and single phone), to mitigate the device heterogeneity effect. To achieve that, *OmniCells* leverages a combination of relative differences and a learned, device-invariant representation using stacked encoder-decoder layers. These features are further harnessed to train a deep model for localizing the user device. Furthermore, we showed how *OmniCells* includes provisions in the model to avoid over-fitting and boost the model generalization ability.

Implementation of our system on different Android-based phones showed that *OmniCells* can obtain a localization accuracy, improving upon five different state-of-the-art indoor positioning systems in two different testbeds. In addition, *OmniCells* provided negligible power consumption compared to the de facto standard technologies utilized in localization (e.g. WiFi and GPS).

REFERENCES

[1] H. Rizk, M. Torki, and M. Youssef, "Cellindeep: Robust and accurate cellular-based indoor localization via deep learning," *IEEE Sensors Journal*, vol. 19, no. 6, pp. 2305–2312, March 2019.

[2] Ye Tian, Bruce Denby, Iness Ahriz, Pierre Roussel, and Gérard Dreyfus, "Robust indoor localization and tracking using GSM fingerprints," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, pp. 157, 2015.

[3] Alex Varshavsky, Eyal De Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason, "GSM indoor localization," *Pervasive and Mobile Computing*, vol. 3, no. 6, pp. 698–720, 2007.

[4] James Kurose and Keith Ross, "Computer networks: A top down approach featuring the internet," *Peorsoim Addison Wesley*, 2010.

[5] Hamada Rizk, Ahmed Shokry, and Moustafa Youssef, "Effectiveness of data augmentation in cellular-based localization using deep learning," pp. 1–6, April 2019.

[6] Mikkel Baun Kjærgaard and Carsten Valdemar Munk, "Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution)," in *Proceedings of the Sixth Annual International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2008, pp. 110–116.

[7] Hamada Rizk, "Device-invariant cellular-based indoor localization system using deep learning," in *The ACM MobiSys 2019 on Rising Stars Forum*. 2019, RisingStarsForum'19, pp. 19–23, ACM.

[8] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

[9] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] Yoshua Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*, pp. 437–478. Springer, 2012.

[12] Kyeong Soo Kim, Sanghyuk Lee, and Kaizhu Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting," *Big Data Analytics*, vol. 3, no. 1, pp. 4, 2018.

[13] Sadiq Jafar Sadiq and Shahrokh Valaee, "Automatic device-transparent rss-based indoor localization," in *Proceedings of the international conference on Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[14] Z Yang, "Powertutor-a power monitor for android-based mobile platforms," *EECS, University of Michigan, retrieved September*, vol. 2, pp. 19, 2012.

[15] Jeongyeup Paek, Kyu-Han Kim, Jatinder P Singh, and Ramesh Govindan, "Energy-efficient positioning for smartphones using cell-id sequence matching," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 293–306.

[16] Mohamed Ibrahim and Moustafa Youssef, "CellSense: An accurate energy-efficient GSM positioning system," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 286–296, 2012.

[17] Mohamed Ibrahim and Moustafa Youssef, "Enabling wide deployment of GSM localization over heterogeneous phones," in *International Conference on Communications (ICC)*. IEEE, 2013, pp. 6396–6400.

[18] Hamada Rizk and Moustafa Youssef, "Monodcell: A ubiquitous and low-overhead deep learning-based indoor localization with limited cellular information," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 109–118.

[19] Hamada Rizk, "Solocell: Efficient indoor localization based on limited cell network information and minimal fingerprinting," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 604–605.

[20] Hamada Rizk, Sherin Elgokhy, and Amany 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.

[21] Yige Zhang, Aaron Yi Ding, Jorg Ott, Mingxuan Yuan, Jia Zeng, Kun Zhang, and Weixiong Rao, "Transfer learning-based outdoor position recovery with telco data," *arXiv preprint arXiv:1912.04521*, 2019.

[22] Rizanne Elbakly and Youssef Moustafa, "Crescendo: An infrastructure-free ubiquitous cellular network-based localization system," in *Proceedings of Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019.

[23] Mohamed Ibrahim and Moustafa Youssef, "A hidden markov model for localization using low-end GSM cell phones," in *Proceedings of the international conference on Communications (ICC)*. IEEE, 2011, pp. 1–5.

[24] Christopher M Bishop, "Information science and statistics," *Pattern Recognition and Machine Learning. Springer*, 2006.

[25] Ahmed Shokry, Marwan Torki, and Moustafa 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.

[26] Moustafa Abbas, Moustafa Elhamshary, Hamada Rizk, Marwan Torki, and Moustafa Youssef, "WiDeep: WiFi-based accurate and robust indoor localization system using deep learning," in *Proceedings of the International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2019.

[27] Xuyu Wang, Lingjun Gao, Shiwen Mao, and Santosh Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2017.

[28] Xuyu Wang, Lingjun Gao, Shiwen Mao, and Santosh 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.

[29] Xuyu Wang, Xiangyu Wang, and Shiwen Mao, "Resloc: Deep residual sharing learning for indoor localization with csi tensors," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017 IEEE 28th Annual International Symposium on*. IEEE, 2017, pp. 1–6.

[30] Xuyu Wang, Xiangyu Wang, and Shiwen Mao, "Cifi: Deep convolutional neural networks for indoor localization with 5 ghz wi-fi," in *International Conference on Communications (ICC)*. IEEE, 2017.

[31] Wei Zhang, Kan Liu, Weidong Zhang, Youmei Zhang, and Jason Gu, "Deep neural networks for wireless localization in indoor and outdoor environments," *Neurocomputing*, vol. 194, pp. 279–287, 2016.

[32] Christos Laoudias, Paul Kemppi, and Christos G Panayiotou, "Localization using radial basis function networks and signal strength fingerprints in wlan," in *Global telecommunications conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.

[33] Gibrán Félix, Mario Siller, and Ernesto Navarro Alvarez, "A fingerprinting indoor localization algorithm based deep learning," in *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on*. IEEE, 2016, pp. 1006–1011.

[34] Omar Hashem, Moustafa Youssef, and A. Harras Harras, "WiNar: Rtt-based sub-meter indoor localization using commercial devices," in *Proceedings of the International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2020.

[35] Xuyu Wang, Lingjun Gao, and Shiwen Mao, "Phasefi: Phase fingerprinting for indoor localization with a deep learning approach," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[36] Xuyu Wang, Lingjun Gao, and Shiwen Mao, "Biloc: Bi-modal deep learning for indoor localization with commodity 5ghz wifi," *IEEE Access*, vol. 5, pp. 4209–4220, 2017.

[37] Liye Zhang, Lin Ma, Yubin Xu, and Cheng Li, "Linear regression algorithm against device diversity for indoor wlan localization system," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[38] Fangfang Dong, Yiqiang Chen, Junfa Liu, Qiong Ning, and Songmei Piao, "A calibration-free localization solution for handling signal strength variance," in *International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*. Springer, 2009, pp. 79–90.

[39] Shih-Hau Fang, Chu-Hsuan Wang, Sheng-Min Chiou, and Pochiang Lin, "Calibration-free approaches for robust wi-fi positioning against device diversity: A performance comparison," in *Proceedings of the 75th vehicular technology conference (VTC Spring)*. IEEE, 2012, pp. 1–5.