CellinDeep: Robust and Accurate Cellular-Based Indoor Localization via Deep Learning

Hamada Rizk, Marwan Torki, *Member, IEEE*, and Moustafa Youssef, *Fellow, IEEE*

***Abstract*— The demand for a ubiquitous and accurate indoor localization service is continuously growing. Current solutions for indoor localization usually depend on using the embedded sensors on high-end phones or provide coarse-grained accuracy. We present *CellinDeep*: a deep learning-based localization system that achieves fine-grained accuracy using the ubiquitous cellular technology. Specifically, *CellinDeep* captures the non-linear relation between the cellular signal heard by a mobile phone and its location. To do that, it leverages a deep network to model the inherent dependency between the signals of the different cell towers in the area of interest, allowing it achieve high localization accuracy. As part of the design of *CellinDeep*, we introduce modules to address a number of practical challenges such as handling the noise in the input wireless signal, reducing the amount of data required for the deep learning model, as avoiding over-training. Implementation of *CellinDeep* on different Android phones shows that it can achieve a median localization accuracy of 0.78m. This accuracy is better than the state-of-the-art indoor cellular-based systems by at least 350%. In addition, *CellinDeep* provides at least 93.45% savings in power compared to the WiFi-based techniques.**

***Index Terms*— Cellular, indoor, localization, deep learning, fingerprinting.**

I. Introduction

OVER the years, cellular phones have gained wide-spread attention, opening the door for many innovations in the area of indoor localization systems that leverage different sensors on the device to achieve its goals. WiFi-based indoor localization systems, e.g. [1]–[8] depend on the installed WiFi infrastructure to estimate the phone location. Nevertheless, not all phones are equipped with WiFi chips, limiting their ubiquitous availability. To increase the localization accuracy, researchers proposed a number of systems that fuse the WiFi signal with the on-board inertial sensors of mobile devices; such as accelerometers, gyroscopes, and compasses [9]–[15]. However, these sensors are only available on the high-end phones.

Given that a large number of people around the world, both in developed and developing countries still use low-end cell phones that do not even support WiFi [16], researchers have proposed a number of systems [17], [18] that leverage the cellular signal for localization. Since, cellular signals are received by all cellphones, this provides a ubiquitous localization service worldwide that can work virtually with any cell phone. Furthermore, this consumes zero extra energy in addition to the standard phone operation. The basic idea of these systems is to capture the signature of the received signal strength (RSS) from the different cell towers at known locations in the area of interest during an offline phase, i.e. construct a “fingerprint”. Then, during the online phase, the cellular signals heard by a user’s device at an unknown location are matched against the constructed fingerprint to find the best match. Nevertheless, current cellular-based indoor localization techniques either try to learn the pattern of the received signal strength using traditional classifiers, e.g. SVM [17] or KNN [18], [19]; or using probabilistic techniques, e.g. [20]–[22]. Traditional machine learning techniques performance cannot handle the large noise inherent in the wireless channel.[1] Probabilistic techniques can counter the inherent wireless signal noise in a better way. However, they usually assume that the signals from different cell towers are independent to avoid the curse of dimensionality problem [25]. This leads to coarse-grained accuracy.

In this paper, we propose *CellinDeep*: a novel cellular indoor localization system. To achieve fine-grained accuracy, *CellinDeep* builds a deep neural network (DNN) model to learn the non-linear correlated relation between the received signal strength from the different cell towers in the area of interest and the user location. Specifically, *CellinDeep* works in two phases: during the offline phase, geo-tagged received signal strength information coming from different cell towers is used to train a deep model. In the online phase, a user freely moves in the area of interest and the RSS measurements from the different cell towers are fed into the deep model to infer her position.

To achieve highly accurate and robust localization, *CellinDeep* needs to make provision for a number of challenges including handling the noisy training data, reducing the data collection overhead typically required for cellular-based localization systems, and avoiding over-fitting to training data. Therefore, we introduce two novel data augmentation algorithms that can automatically increase the size of the

Manuscript received October 24, 2018; revised December 3, 2018; accepted December 3, 2018. Date of publication December 10, 2018; date of current version February 15, 2019. This work was supported by a grant from the Egyptian National Telecommunication Regulatory Authority. The associate editor coordinating the review of this paper and approving it for publication was Dr. Ashish Pandharipande. *(Corresponding author: Hamada Rizk.)*

H. Rizk is with the Department of Computer Science and Engineering, Egypt-Japan University of Science and Technology, Alexandria 21934, Egypt (e-mail: hamada.rizk@ejust.edu.eg).

M. Torki and and M. Youssef are with the Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt (e-mail: mtorki@alexu.edu.eg; moustafa@alexu.edu.eg).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. The Supplementary Material contains an Appendix for the paper in the form of PDF file. This material is 0.0831 MB in size.

Digital Object Identifier 10.1109/JSEN.2018.2885958

[1]SVM can be considered as a special case of a shallow neural network that has limited representational learning capabilities when compared to a deep neural network [23], [24].

input data, different location smoothing and outlier detection approaches, as well as introduce different model regularization techniques to avoid over-fitting.

Evaluation of *CellinDeep* using different Android phones shows that it can achieve a consistent median localization error of 0.78m. This is better than the state-of-the-art cellular systems by more than 350%. In addition, *CellinDeep* consumes zero extra energy consumption compared to the normal phone operation and is more power efficient than WiFi techniques by 93.45%.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents an overview on how *CellinDeep* works. Section 4 gives the details of *CellinDeep* and how it handles different practical considerations. We evaluate the system performance in Section 5 and compare it to the state-of-the-art. Finally, Section 6 concludes the paper.

II. Related Work

In this section, we focus on four groups of related work for indoor localization classified into: WiFi-, sensor-, cellular- and Deep learning-based techniques.

A. WiFi-Based Techniques

To achieve high accuracy for localization, WiFi-based techniques usually depend on building a WiFi fingerprint of the overheard WiFi access points (APs) in the offline phase. While, in the online phase, the received signals are used find the best matching location from the fingerprint. This matching can be either deterministic [1], [2] or probabilistic [3]. Many systems have been proposed to address several WiFi-based challenges over the years. For instance, [13] fuses WiFi with other sensors that are available only on smartphones to combat the noise of WiFi channels. Instead of using RSS readings, other techniques [6]–[8] leverage the detailed channel state information (CSI) obtained from some WiFi chips. Despite the fact that WiFi fingerprinting techniques have high accuracy due to the limited propagation range of WiFi APs, [26], they are not as ubiquitous as cellular-based techniques due to their dependency on WiFi coverage and high-end phones.

In contrast, CellinDeep presents a technique that relies on the widely-spread cellular technology, which is supported by all phones around the globe. In addition, CellinDeep provides accuracy comparable to WiFi-based techniques by leveraging deep learning models with novel data augmentation techniques.

B. Sensor-Based Techniques

Many indoor localization systems have been proposed drawing on the available sensors of modern smartphones [9]–[14]. The systems in [9], [12], [13], and [15] harness the smartphones' inertial sensors (e.g. magnetometer, accelerometer and gyroscope) to track the user location based on dead-reckoning. The main problem with these techniques is the noise in sensor measurements, which accumulates quickly over time. To handle error accumulation; Zee [9], e.g., resets the error by employing map matching to the floor-plan. Unloc [10], [13] and SemanticSLAM [11] opportunistically reset the error on encounters of some sensor-based landmarks in the area of interest. On the other hand, Headio [14] tries to correct the sensor distortion by leveraging the camera of the smartphone.

All sensor-based techniques require high-end phones to work, limiting their ubiquity.

C. Cellular-Based Techniques

To provide ubiquitous localization, a number of systems have been proposed leveraging the cellular network signals, mainly for outdoor environments [20]–[22], [27], where high accurate localization is not required. On the other hand, research regarding cellular-based indoor localization is limited [17]–[19], [28]. This is due to the challenges of indoor localization that requires highly accuracy despite high variability and noise of cellular signals over time and the large coverage area of cell towers. The basic idea of these techniques is to leverage the RSS measurements collected in the offline phase to build a model to discriminate between different locations in the area of interest. SkyLoc [18], [19] adopted the K-nearest neighbor classifier (KNN) to match the received signals and estimate the user location. In order to enhance the accuracy of their system, they collect GSM data with special high-quality cellular modems. Similarly, [17], [28] introduce a cellular indoor localization systems based on a one-vs-all SVM and carry out Map-matching using a Bayesian filter. Traditional machine learning techniques performance cannot handle the large noise inherent in the wireless channel (as we quantify in Section V-D) as well as are susceptible to over-fitting. Probabilistic techniques can counter the inherent wireless signal noise in a better way. However, they usually assume that the signals from different cell towers are independent to avoid the curse of dimensionality problem [25]. This leads to coarse-grained accuracy.

CellinDeep, on the contrary, utilizes a deep neural network that is able to learn complex relations in the data data automatically. In addition, it generates synthetic data to reduce the calibration overhead and increases its generalization ability. Moreover, it has provisions to handle overfitting.

D. Deep Learning-Based Techniques

Recently, different deep learning techniques have been proposed for indoor localization, mainly based on the WiFi signal. For example, DeepFi [4] uses Restricted Boltzman Machines to pre-train a deep neural network that takes as input the channel state information (CSI). Similarly, ResLoc [29] applies deep residual sharing learning on CSI information. CiFi [30], on the other hand, proposes a deep convolutional network to address the same problem based on CSI while [24] uses features extracted from the magnetic field. In [31] and [32], a feed-forward neural network is adopted to detect the building, floor and location based on RSS signals. To enhance location estimation, the centroid method is used in [31] while [32] adopts a hidden Markov model. *Different from these techniques, CellinDeep leverages cellular signals due to its ubiquity. In addition, it introduces a number of modules to address the noisy wireless channel as well as*

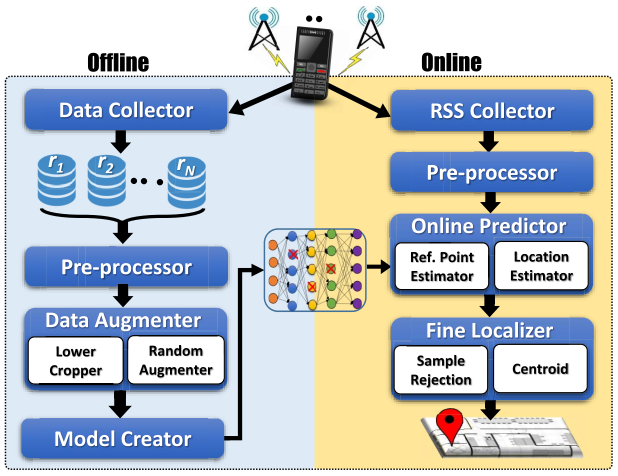

Fig. 1. The CellinDeep system architecture.

deep learning requirements including novel data augmentation schemes, model regularization and anomalous estimation avoidance.

III. The CellinDeep System

In this section, we present the details of the different modules of *CellinDeep*.

IV. Overview and System Model

A. Overview

Figure 1 shows the *CellinDeep* system architecture. *CellinDeep* works in two phases: an offline training phase and online tracking phase. During the offline phase, the Data Collector module running on a cell phone stores the time-stamped cell towers signal measurements at different selected reference points in the area of interest. These collected training measurements are sent to a *CellinDeep* running service in the cloud for further processing. The Pre-processor module is used to handle the noise in the input data and to extract and normalize the required features. The Data Augmenter module is used to extend the collected training data by generating synthetic training samples. This helps not only in reducing the training overhead but also in reducing the noise effect and avoiding over-fitting as we describe in Section IV-D. Next, the Model Creator module constructs and trains a deep neural network while also selecting the optimal parameters for the model. Finally, the trained model is saved for later use by the Online Predictor module.

During the online phase, users are tracked in real time by forwarding the RSS information from the cellular towers at the current unknown user location to the *CellinDeep* server. This data is first filtered by the *Pre-processor* module. Then, the Online Predictor module feeds the data to the deep model constructed in the offline phase to estimate the likelihood of the user being at the different reference points trained during the offline phase. The system then fuses these different likelihood to obtain the user location in the continuous space. Finally, the Fine Localizer module is used to further reduce outliers and provide a more smoothed user location.

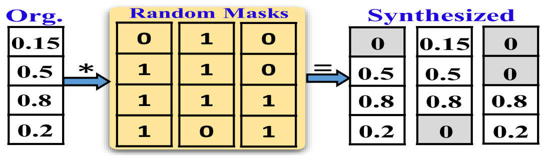

Fig. 2. Example of the Random Augmenter operation on normalized RSS values.

B. System Model

We assume a 2D area of interest where a total of q cell towers can be heard over the entire area. During the offline phase, in order to construct the fingerprints, we collect training samples at n reference points spaced at an equal distance (d_r) apart. The designer should select a reference point spacing to balance between accuracy and computational overhead as we quantify in Section V-B.

To collect the training data, a user uses her cell phone to scan for the nearby cell towers at different reference locations. According to the standard [22], up to seven towers of the total q available cell towers in the area can be heard at any scan. For each heard cell tower, a pair $c =$<CID, RSS> is recorded where CID represents the cell tower unique ID and RSS is the received signal strength from that tower measured in Arbitrary Strength Unit (ASU).[2] Different RSS scans are performed at each reference point to collect the training data.

The goal during the online phase is to find the most probable reference point the user can be standing at given the currently heard cellular RSS in a scan. This is further processed to provide a smooth location in the continuous space.

C. The Pre-Processor Module

This module runs during both the offline and online phases. It processes the input scans to normalize the data for the next modules. Specifically, due to the noise in the wireless channel, not all cell towers can be heard in all scans. To have a consistent set of cell towers input to the deep model, the Pre-processor extends the input scan vector to a vector $s = (s_1, s_2, \ldots, s_q)$ of length q, where each entry represents a cell tower that can be heard in the entire area of interest. Note that only a subset of these q element have been heard in the current scan. Non-heard cell towers are assigned an RSS of 0 ASU. Moreover, we also observed that some scans have incorrect cell tower IDs (such as 65636 which is the maximum number of available cell towers for any location area (LAC) [33]). This anomalous event may occur during the brief duration of a Radio Access Technology (RAT) change [34] and can be discovered with the absence of a LAC ID which should be included with any scan. Thus, the spurious cell tower IDs are dropped. Finally, to improve the convergence time of the model, all RSS values are normalized to the range of 0 to 1.

D. The Data Augmenter Module

Deep learning models have the advantage of capturing the non-linear and correlated relation between the RSS received

[2]ASU is an integer value ranging from 0 to 31 and is proportional to the RSS measured in dBM based on the relation: $dBm = 2 \times ASU - 113$.

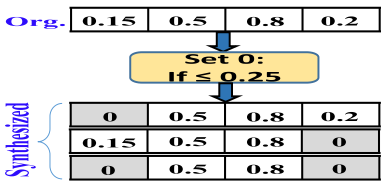

Fig. 3. Example of the Lower Cropper on normalized RSS values.

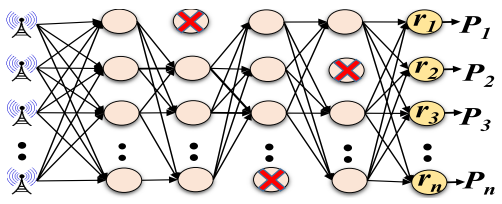

Fig. 4. Deep network structure. Crossed-out neurons represent dropped-out neurons.

from the different cell towers in the area of interest, enabling higher accuracy compared to traditional machine learning models. However, in order to achieve that, they require a lot of training data, which may significantly increase the training overhead. To reduce the training overhead, the Data Augmenter module automatically generates synthetic training samples from a small seed collected data, increasing the training data size and reducing the data collection overhead. *CellinDeep* has two novel data augmenters: Random Augmenter and Lower-bound Cropper. Both techniques, are described next.

1) The Random Augmenter: The number of cell towers detected at a fixed location varies with time due to the noisy wireless channel. Based on our observations, the probability of hearing the maximum number of cell towers (seven as per the GSM specification) is around 6% as shown in Figure 1 in the supplementary material.

The Random Augmenter technique leverages this fact to increase the training data size. The idea is to randomly generate a binary mask that can be multiplied by the RSS vector to selectively drop certain cell towers in a scan (Figure 2). This is analogous to the case of not hearing some cell towers at a given location in the different samples. The module also ensures that the cell tower the device is associated with is always present to capture reality better.

2) The Lower-Bound Cropper: The intuition behind this technique is that cell towers with weak received signals, i.e. close to the receiver sensitivity,[3] appear and disappear between the different scans with a high rate at the same location. The Lower-bound Cropper augmentation technique leverages this observation to increase the training data size. In particular, for a given collected RSS vector, any entry s_i whose value is less than a certain threshold τ is a candidate to be removed (set to zero), mimicking that this cell tower has not been heard in the generated synthetic sample (Figure 3). All combinations of these entries can then be added to the training dataset.

3) Discussion: Note that for both augmentation techniques, any generated duplicates are removed. We show the effect of both modules on system performance in Section V-B.

Furthermore, in addition to reducing the data collection overhead, the data augmentation techniques also increase the robustness of *CellinDeep* to noisy data as well as reduce the possibility of over-fitting training data due to the generation of additional training samples that do not exactly match the collected data.

[3]The receiver sensitivity refers to the minimum signal strength the wireless chip is able to detect a signal at.

E. The Model Creator

One of the main advantages of the DNNs is their ability to represent the model's input data in a hierarchical distributed form [23]. This is desirable when dealing with such challenging cellular data, where the signal from the different cell towers in the area of interest is correlated and non-linear in the user location.

1) DNN Architecture: Figure 4 presents our deep network structure. *CellinDeep* adopts a fully connected neural network [35] . The input layer of the network is a vector of length q, where each element represents the cellular signal strength detected from the q towers in the area of interest (zero for cell towers not heard in the current scan). The hierarchical representation of *CellinDeep* is obtained by four hidden layers of nonlinear processing units. We use rectified linear units (ReLUs) as the activation functions for the hidden layers since they help reduce the vanishing gradient problem [36].

The output layer consists of a number of units corresponding to the number of reference points in the area of interest surveyed during the offline phase. The problem is formulated as a classification problem since this simplifies data collection (i.e. permits collection at low density reference points) unlike equivalent regression models. Since our reference points constitute a multinomial (multi-class) classification problem, the softmax function is used in the output layer. This outputs the probability that the input sample/scan comes from a specific reference point. More formally, assume that we have t training samples. Each record s_i where $1 \leq i \leq t$ contains q features $(s_{i1}, s_{i2}, \ldots s_{iq})$ that present the RSS information coming from q cell towers covering the entire area of interest. The corresponding discrete outcomes (i.e logits) for the input s_i is $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})$ 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:

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

2) Preventing Over-Fitting: To increase the model robustness and further reduce over-fitting, *CellinDeep* employs two regularization techniques: First, we use dropout regularization [37] which has been shown to be useful for large networks. It is used to prevent the neurons from developing co-dependencies among each other during training. In this technique, neurons are randomly removed from the deep neural network during the training phase at different epochs

(Figure 4). Hence, the removed neurons do not contribute to the forward nor the back-propagation passes. It has the effect that the neural network samples different architectures during training, but all these architectures share weights. Therefore, the DNN is forced to learn more robust features that allow the network to generalize rather than memorize. We provide an experiment to show the effect of dropout on the results in section V-B.

Second, we leveraged early stopping so that training would halt once the performance improvements are no longer obtained after some number of epochs - this number is called the patience value [38].

F. Online Predictor

During the online phase, a user is at an unknown location u receiving cellular signals from the nearby cell towers. The goal of the online phase is to find the reference point r^* that has the maximum probability given the RSS vector s . That is, we want to find:

$$r^* = \underset{r}{\text{argmax}}[P(r|s)] \tag{2}$$

CellinDeep leverages the probabilistic output from the softmax layer of the DNN to estimate the closest reference point r^* to the user. Note that, compared to traditional probabilistic techniques in literature, e.g. [20]–[22]; *CellinDeep* does not assume independence of the cell towers but rather depends on the DNN model to capture the correlation between the different cell towers. The probabilistic output (obtained as described previously) is denoted as $P = [P_1, P_2, \ldots, P_n]$. Where n is the number of reference points in the area of interest and $P_i (1 \leq i \leq n)$, represents the possibility that the signal strength vector s came from the i^{th} reference point. P_i is formulated as follows:

$$P_i = P(r_i|s) \tag{3}$$

The reference point with the highest probability is returned as the predicted location. However, using this technique, the user location can only be one of the discrete reference points. Therefore, to allow tracking the user in the continuous space, we employ a smoothing step (spatial weighted average) over the reference points by estimating the user location as the center of mass of most probable k reference points, weighted by their corresponding softmax likelihood [31]. More formally:

$$l = \frac{\sum_{i=1}^{k} P_i r_i}{\sum_{i=1}^{k} P_i} \tag{4}$$

where r_i is the coordinate of reference point i after arranging them in descending order according to the softmax likelihood.

G. The Fine Localizer

The method described in the previous section considers only one sample (i.e a single RSS vector) for each location estimate. However, in general, the scan rate of the cellular data by the cellular chip is much higher than the movement rate. Hence, the *CellinDeep* system leverages multiple successive samples in its operation to further enhance accuracy. In particular, we introduce two techniques to obtain a fine-grained location estimate in the continuous spatial space based on the notion of multiple sampling. Both techniques estimates the location from each of υ input RSS scans and then combine the different estimated locations differently:

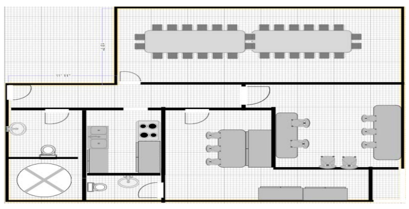

Fig. 5. Layout of the testbed.

1) The Sample Rejection Approach: This approach removes the anomalous location estimates that may occur due to the noisy wireless channel. To do that, it calculates the distance between each estimated location and the remaining υ reported locations, which we call inter-location distance, defined as:

$$d_i = \sum_{j \neq i} \frac{d(l_i, l_j)}{\upsilon - 1} \tag{5}$$

Where $d(l_i, l_j)$ is the Euclidean distance between location l_i and l_j.

Then, it calculates the average inter-location distance d_{avg} and rejects samples that do not satisfy $d_i \leq d_{avg}$ [39]. Finally, the remaining locations are averaged in order to report the a fine-grained estimate of the user location.

2) The Centroid Approach: The module performs a temporal averaging step over the υ estimated locations. The weight of each location estimate in this averaging step is taken to be proportional to the highest softmax likelihood of that location estimate as:

$$l = \frac{\sum_{i=1}^{\upsilon} P_{\max_i} l_i}{\sum_{i=1}^{\upsilon} P_{\max_i}} \tag{6}$$

Where l_i is the location of the i^{th} sample reported by online predictor module and the $P_{\max_i}$ is the softmax highest likelihood for that sample.

Both techniques can be applied together to further enhance localization accuracy.

V. Evaluation

In this section, we evaluate the performance of the *CellinDeep* system in a typical indoor environment. We start by analyzing the effect of different parameters on performance. Next, we investigate the robustness of the system under different scenarios. Finally, we compare our system to the state-of-the-art techniques.

More experiments and their results can be found in the supplementary material.

A. Data Collection

We deployed our system on a floor at our university campus building with an area of 11×12m containing offices, a meeting

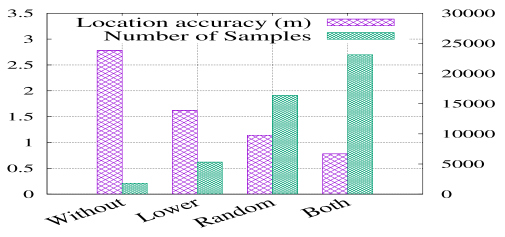

Fig. 6. Effect of data aug. methods on accuracy.

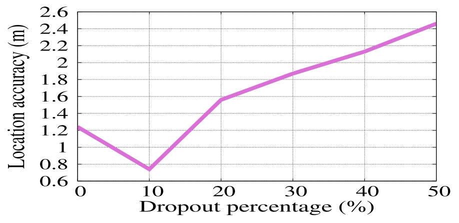

Fig. 7. Effect of the dropout on accuracy.

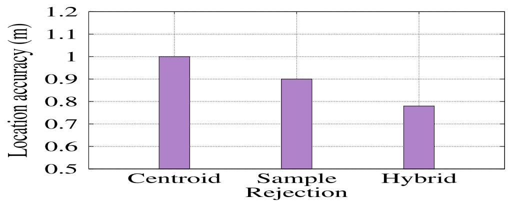

Fig. 8. The effect of the Fine-Localizer methods.

TABLE I
TESTBED PARAMETERS

Criteria	Value
Area (m^2)	11×12
Grid cell size (m^2)	1
Total number of cell towers	17
Building Material	Brick
Sampling rate (scan/sec)	3.33

TABLE II
DEFAULT PARAMETERS VALUES USED IN EVALUATION

Parameter	Range	Default
Provider	A, B, C	Provider A
Data Augmentation	Random Augmenter, Lower Cropper, Both	Both
Learning rate	0-1	0.001
Batch size	1-Dataset size	256
Dropout rate (%)	0-50	10
Number of patience epochs	1-10	10
Number of hidden Neurons	20-1000	280
Number of classes	1-51	51
Number of Top probable loc. (k)	1-51	7
Fine localizer method	Sample Rejection, Centroid, Hybrid	Hybrid
Number of samples (υ)	1-10	5

room, and corridors (Figure 5). We collected the training data from 51 reference points that span the whole testbed area. The training reference points were collected on a uniform grid with a 1m spacing. The data was collected by two participants using different Android phones (e.g. HTC One X9, Google Pixel XL, Tecno Phantom 6, HTC One E9, Motorola Moto G5 and ZTE Blade 7). The dataset is collected over two days to capture the time-variant nature of the cellular signal strength. We implemented a scanning application using the Android SDK to scan cell towers in the area of interest. The default scanning rate was set to 3Hz. To test the robustness of *CellinDeep* to temporal changes in the environment, we collected an independent test set on different days at arbitrary locations in the testbed area (different from the reference points). Table I summarizes the testbed parameters.

We used the Adam optimizer [40] and categorical cross-entropy as our loss function.

B. Effect of Changing CellinDeep Parameters on Accuracy

1) Data Augmentation Techniques: Figure 6 shows the effect of data augmentation methods (Section IV-D) on the *CellinDeep* performance. The figure shows that both the Lower Cropping and Random Cropping methods can improve the system performance compared to augmentation-free training. The figure also confirms that the more training samples generated by the augmentation technique the better the performance that can be achieved without extra fingerprinting effort. Applying both augmentation techniques together significantly improves the performance, not only by increasing the number of training samples, but also by implicitly simulating the inherent variation of the wireless channel.

2) Dropout Percentage: Figure 7 shows the effect of increasing the dropout percentage on performance. The figure shows that an optimal value is achieved at 10% dropout. This can be explained by noting that as the network size grows, the role of the dropout regularization becomes significant in avoiding over-fitting. However, at larger dropout percentage, the model tends to under-fit the training data.

3) Fine-Localizer Techniques:

a) Algorithm: Figure 8 compares the three Fine-localizer methods (Section IV-G). The figure shows the hybrid method provides an improvement of around 22% and 13% over the centroid method and the sample rejection method respectively. **Number of samples:** Figure 9 shows the effect of changing the number of samples (υ) used by the Fine-localizer module on performance. The figure shows that as υ increases, the accuracy improves until it reaches an optimal value at $\upsilon = 5$ beyond which it begins to deteriorate. This is due to two opposing factors: (1) As we increase υ we have more information to predict the user location, (2) However, as υ increases, more time is spent to collect these samples which may involve crossing reference points boundaries. This has a negative influence on accuracy.

C. Robustness Experiments

In this section, we evaluate the robustness of *CellinDeep* under different scenarios.

1) Different Providers: Figure 10 shows the performance of *CellinDeep* when tested with three different cellular providers. Each provider covers the area of interest with a cell tower

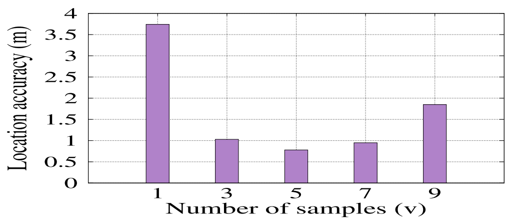

Fig. 9. Effect of no. of samples (v).

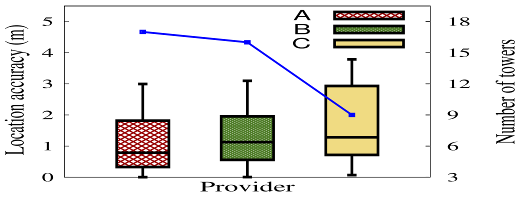

Fig. 10. Effect of cell providers on accuracy.

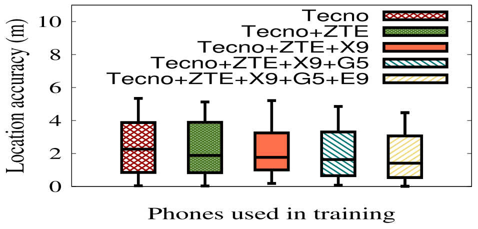

Fig. 11. Effect of device heterogeneity on accuracy.

density of 17, 16, and 9 for providers A, B, and C respectively. The figure shows that the accuracy of the different providers is proportional to the their cell tower densities: the higher the density, the higher the accuracy. Nevertheless, even with the challenging case of Provider C, *CellinDeep* still works well due to the generation of more data using the different data augmenters discussed in Section IV-D.

2) Device Heterogeneity: We investigate the effect of training *CellinDeep* with phones different than the one used in testing. We considered six different phones: Tecno Phantom 6, ZTE Blade 7, HTC X9, Moto G5, HTC E9 and Google pixel XL. The Google Pixel XL phone is fixed as the testing device. Then, we combine the other phones incrementally for training. Figure 11 shows that the more phones considered in training, the better the accuracy obtained. Combining different phones serves to induce variations in the training samples collected at the same reference points. This is beneficial for the model's generalization ability as it learns to capture the underlying relationship between the heard signal and the location in a device-independent way. Even when using a single phone during the training, *CellinDeep* still provides a median localization accuracy of less than 2.5m. This is due to the different regularization techniques employed by the system as well as the use of data augmenters.

D. Comparative Evaluation

In this section we compare *CellinDeep* performance to two state-of-the-art cellular localization techniques: SkyLoc [18] and the system in [17]). SkyLoc [18] uses a KNN classifier to provide discrete estimate of the user locations based on the RSS vector. On the other hand, [17] (denoted as SVM) uses a one-vs-all SVM classifier for multiclass (location) estimation as well as employing a Map-matching through the use of

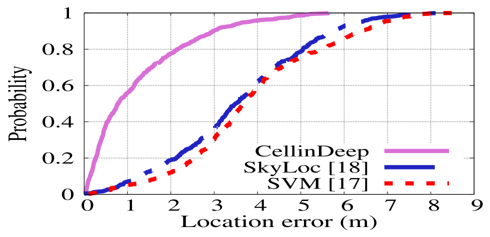

Fig. 12. CDF of the localization error.

Bayesian filter for further enhancement. The construction of both techniques takes long period in order to get adequate amount of data for training. Figure 12 shows the CDF of location error for the different techniques. All techniques are trained using data collected by the HTC X9 and the Moto G5 cell phones. Testing was carried out using samples from Google Pixel XL phone and the two phones used in training. The figure illustrates that *CellinDeep* outperforms the other techniques with an enhancement in the median error by 350% and 371% than the SVM [17] and the SkyLoc [18] systems respectively. This confirms that the *CellinDeep* deep model and the associated modules for data augmentation and over-fitting avoidance can capture the interrelated signatures of the different cell towers at different reference points better than traditional methods.

VI. Conclusions

We proposed *CellinDeep*, a deep learning system for cellular-based indoor localization. We presented the details of the system and how it augments the training data to increase the system robustness and reduce the calibration effort. Furthermore, we showed how *CellinDeep* includes provisions in the model to avoid over-training and increase the model robustness. Implementation of our system on Android-based phones showed that it can achieve consistent median localization accuracy of 0.78m. This accuracy is better than other indoor cellular-based systems by at least 350%. In addition, *CellinDeep* provided at least 93.45% savings in power compared to the WiFi-based techniques.

References

[1] P. Bahl and V. N. Padmanabhan, "Radar: An in-building RF-based user location and tracking system," in *Proc. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, Mar. 2000, pp. 775–784.

[2] P. Kontkanen, P. Myllymaki, T. Roos, H. Tirri, K. Valtonen, and H. Wettig, "Topics in probabilistic location estimation in wireless networks," in *Proc. IEEE 15th Int. Symp. Pers. Indoor Mobile Radio Commun.*, vol. 2, Sep. 2004, pp. 1052–1056.

[3] M. Youssef and A. Agrawala, "The horus WLAN location determination system," in *Proc. 3rd Int. Conf. Mobile Syst. Appl. Services*, Jun. 2005, pp. 205–218.

[4] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2015, pp. 1666–1671.

[5] Y. Jiang *et al.*, "Ariel: Automatic Wi-Fi based room fingerprinting for indoor localization," in *Proc. ACM Conf. Ubiquitous Comput.*, Sep. 2012, pp. 441–450.

[6] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. M. Ni, "CSI-based indoor localization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1300–1309, Jul. 2013.

[7] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *Proc. ACM SIGCOM Comput. Commun. Rev.*, vol. 40, Oct. 2010, pp. 159–170.

[8] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "Precise indoor localization using PHY information," in *Proc. 9th Int. Conf. Mobile Syst. Appl. Services*, 2011, pp. 413–414.

[9] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2012, pp. 293–304.

[10] M. Alzantot and M. Youssef, "Crowdinside: Automatic construction of indoor floorplans," in *Proc. 20th Int. Conf. Adv. Geograph. Inf. Syst.*, Nov. 2012, pp. 99–108.

[11] H. Abdelnasser *et al.*, "SemanticSLAM: Using environment landmarks for unsupervised indoor localization," *IEEE Trans. Mobile Comput.*, vol. 15, no. 7, pp. 1770–1782, Jul. 2016.

[12] M. Alzantot and M. Youssef, "Uptime: Ubiquitous pedestrian tracking using mobile phones," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 3204–3209.

[13] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, Jun. 2012, pp. 197–210.

[14] Z. Sun, S. Pan, Y. C. Su, and P. Zhang, "Headio: Zero-configured heading acquisition for indoor mobile devices through multimodal context sensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2013, pp. 33–42.

[15] Y. Shu, C. Bo, G. Shen, C. Zhao, L. Li, and F. Zhao, "Magicol: Indoor localization using pervasive magnetic field and opportunistic WiFi sensing," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1443–1457, Jul. 2015.

[16] *Type of Mobile One Owned by Adults in the United States 2017*, 2018.

[17] Y. Tian, B. Denby, I. Ahriz, P. Roussel, and G. Dreyfus, "Robust indoor localization and tracking using GSM fingerprints," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, no. 1, p. 157, Dec. 2015.

[18] A. Varshavsky, E. de Lara, J. Hightower, A. Lamarca, and V. Otsason, "GSM indoor localization," *Pervasive Mobile Comput.*, vol. 3, no. 6, pp. 698–720, Dec. 2007.

[19] V. Otsason, A. Varshavsky, A. Lamarca, and E. de Lara, "Accurate GSM indoor localization," in *Proc. Int. Conf. Ubiquitous Comput.* Springer, Sep. 2005, pp. 141–158.

[20] M. Ibrahim and M. Youssef, "CellSense: A probabilistic RSSI-based GSM positioning system," in *Proc. IEEE Global Telecommun. Conf. GLOBECOM*, Dec. 2010, pp. 1–5.

[21] M. Ibrahim and M. Youssef, "A hidden Markov model for localization using low-end GSM cell phones," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.

[22] M. Ibrahim and M. Youssef, "Cellsense: An accurate energy-efficient GSM positioning system," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 286–296, Jan. 2012.

[23] C. C. Aggarwal, *Neural Network and Deep Learning*. Springer, 2018.

[24] N. Lee and D. Han, "magnetic indoor positioning system using deep neural network," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Sep. 2017, pp. 1–8.

[25] C. Bishop, "Information science and statistics," in *Pattern Recognition and Machine Learning*. Springer, 2006.

[26] P. Davidson and R. Piché, "A survey of selected indoor positioning methods for smartphones," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1347–1370, 2nd Quart., 2017.

[27] M. Ibrahim and M. Youssef, "Enabling wide deployment of GSM localization over heterogeneous phones," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 6396–6400.

[28] Y. Oussar, I. Ahriz, B. Denby, and G. Dreyfus, "Indoor localization based on cellular telephony RSSI fingerprints containing very large numbers of carriers," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, no. 1, p. 81, Dec. 2011.

[29] X. Wang, X. Wang, and S. Mao, "Resloc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.

[30] X. Wang, X. Wang, and S. Mao, "CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[31] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting," *Big Data Anal.*, vol. 3, no. 1, p. 4, Dec. 2018.

[32] 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, Jun. 2016.

[33] A. Parmar and K. M. Pattani, "Sniffing GSM traffic using RTL-SDR and kali linux OS," *Int. Res. J. Eng. Technol.*, vol. 4, no. 1, pp. 1637–1642, 2017.

[34] A. Tudzarov and T. Janevski, "M-RATS: Mobile-based radio access technology selector for heterogeneous wireless environment," in *Proc. 18th Telecommun. Forum (TELFOR)*, Nov. 2010, pp. 336–339.

[35] A. Shokry, M. Torki, and M. Youssef, "Deeploc: A ubiquitous accurate and low-overhead outdoor cellular localization system," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, Nov. 2018, pp. 339–348.

[36] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Jun. 2011, pp. 315–323.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[38] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 437–478.

[39] H. Rizk, S. Elgokhy, and A. Sarhan, "A hybrid outlier detection algorithm based on partitioning clustering and density measures," in *Proc. 10th Int. Conf. Comput. Eng. Syst. (ICCES)*, Dec. 2015, pp. 175–181.

[40] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

Hamada Rizk is currently pursuing the Ph.D. degree with the Computer Science and Engineering Department, Egypt-Japan University of Science and Technology. He is also a Teaching Assistant at Tanta University, Egypt. His research interests include indoor localization and machine learning. He was a recipient of a Scholarship from the Egyptian Ministry of Higher Education. His work was accepted in the third Student Research Competition of the ACM SIGSPATIAL in 2018.

Marwan Torki received the B.Sc. and M.Sc. degrees in computer science from Alexandria University, Egypt, in 2003 and 2006, respectively, and the Ph.D. degree in computer science from Rutgers University, New Brunswick, NJ, USA, in 2011. He is currently an Assistant Professor with the Computer and Systems Engineering Department, Alexandria University.

Moustafa Youssef (F'19) is currently a Professor at Alexandria University and the Founder and Director of the Wireless Research Center of Excellence, Egypt. He has tens of issued and pending patents. His research interests include mobile wireless networks, mobile computing, location determination technologies, pervasive computing, and network security. He was a recipient of the 2003 University of Maryland Invention of the Year Award, the 2010 TWAS-AAS-Microsoft Award for Young Scientists, the 2013 and 2014 COMESA Innovation Award, the 2013 ACM SIGSpatial GIS Conference Best Paper Award, the 2017 Egyptian State Award, multiple Google Research Awards, among many others. He is the Lead Guest Editor of the upcoming IEEE Computer Special Issue on Transformative Technologies, an Associate Editor for IEEE TMC and ACM TSAS, an Area Editor of Elsevier PMC, and served on the organizing and technical committees of numerous prestigious conferences. He is an ACM Distinguished Speaker and an ACM Distinguished Scientist.