This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Zhang, Yige; Xiao, Yu; Zhao, Kai; Rao, Weixiong

DeepLoc: Deep Neural Network-based Telco Localization

Published in:

International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services

Published: 01/01/2019

Document Version

Peer reviewed version

Please cite the original version:

Zhang, Y., Xiao, Y., Zhao, K., & Rao, W. (2019). DeepLoc: Deep Neural Network-based Telco Localization. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (pp. 258-267). ACM. <http://10.1145/3360774.3360779>

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

DeepLoc: Deep Neural Network-based Telco Localization

Yige Zhang

School of Software Engineering
Tongji University, Shanghai, China
yigezhang@tongji.edu.cn

Kai Zhao

Robinson College of Business
Georgia State University, Atlanta, Georgia, United States
kzhao4@gsu.edu

Yu Xiao

Department of Communications and Networking
Aalto University, Espoo, Finland
yu.xiao@aalto.fi

Weixiong Rao

School of Software Engineering
Tongji University, Shanghai, China
wxrao@tongji.edu.cn

ABSTRACT

Recent years have witnessed unprecedented amounts of telecommunication (Telco) data generated by Telco radio and core equipment. For example, measurement records (MRs) are generated to report the connection states, e.g., received signal strength at the mobile device, when mobile devices give phone calls or access data services. Telco historical data (e.g., MRs) have been widely analyzed to understand human mobility and optimize the applications such as urban planning and traffic forecasting. The key of these applications is to precisely localize outdoor mobile devices from these historical MR data. Previous works calculate the location of a mobile device based on each single MR sample, ignoring the sequential and temporal locality hidden in the consecutive MR samples. To address the issue, we propose a deep neural network (DNN)-based localization framework namely DeepLoc to ensemble a recently popular sequence learning model LSTM and a CNN. Without skillful feature design and post-processing steps, DeepLoc can generate a smooth trajectory consisting of accurately predicted locations. Extensive evaluation on 6 datasets collected at three representative areas (core business, urban and suburban areas in Shanghai, China) indicates that DeepLoc greatly outperforms 10 counterparts.

KEYWORDS

Outdoor Localization, Deep Neural Network, Telecommunication Signal Strength Indicator

ACM Reference Format:

Yige Zhang, Yu Xiao, Kai Zhao, and Weixiong Rao. 2019. DeepLoc: Deep Neural Network-based Telco Localization. In *16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, November 12–14, 2019, Houston, TX, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3360774.3360779>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MobiQuitous, November 12–14, 2019, Houston, TX, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-7283-1/19/11...\$15.00

<https://doi.org/10.1145/3360774.3360779>

1 INTRODUCTION

Recent years have witnessed unprecedented amounts of telecommunication (Telco) data generated by Telco radio and core equipment. For example, when mobile devices give phone calls or access data services, measurement records (MRs) are generated to report the connection states, e.g., received signal strength of the mobile device, during each call/session. In a modern urban city, Telco data generated by Telco radio and core equipment account to 2.2TB per day, significantly greater than those accounting and billing data generated by business supporting systems with trivially tens of GB/day [4]. Massive Telco historical data (e.g., MRs) have been widely used to understand human mobility [2, 24] and optimize the applications such as urban planning and traffic forecasting [3], due to the unique advantages: the data can be collected cheaply, frequently and on a global scale.

The key of human mobility and the applications above is to capture the locations of human beings holding mobile devices. To this end, the outdoor localization problem, i.e., how to precisely localize outdoor mobile devices from Telco historical data (e.g., MRs), has attracted intensive research interests. For example, a simple approach adopted by Google MyLocation [1] is to approximate outdoor locations by the positions of cellular towers connected by mobile devices. This method unfortunately suffers from median errors of hundreds and even thousands of meters. More recently, by using the sparse geo-tagged MR data as training data, machine learning approaches such as Random Forest [5, 25, 27], Deep Neural Network [23], and Hidden Markov Model (HMM) [6, 10, 13] are exploited to localize outdoor mobile devices with the median errors of tens of meters.

Unfortunately, the above-mentioned approaches usually calculate location based on a single MR sample, ignoring the sequential or temporal correlations between the locations. When people holding mobile devices are moving around, the associated locations obviously exhibit the temporal and spatial locality. Though several works tried to either manually design temporal and spatial contextual features based on domain expert knowledge [5, 27] or use the first order sequence model such as HMM [6, 10, 13], the challenge still remains: how to automatically discover meaningful spatial and temporal dependencies without applying non-trivial postprocessing steps [19] such as filtering techniques [13] and map-matching algorithms [26]?

To address the issue above, in this paper, we propose a DNN-based localization framework namely DeepLoc to ensemble a

recently popular sequence learning model, long short-term memory networks (LSTM), and a convolutional neural network (CNN). With trivial feature engineering overhead, DeepLoc automatically extracts contextual features and generates the trajectory connecting predicted outdoor locations from input MR sequence data. Specifically, for each MR within the input MR sequence DeepLoc first exploits the CNN model to estimate a score for every potential location based on the extracted spatial features. Next, for the entire input MR sequence, we leverage RecuLSTM (an improved version of LSTM) to generate a smooth trajectory. RecuLSTM infers each location within the trajectory by the scores estimated by CNN. In this way, RecuLSTM and CNN work together to generate a smooth trajectory consisting of precisely estimated locations. As a summary, we make the following contributions in this paper.

- To the best of our knowledge, this is the first to exploit DNN to automatically learn spatiotemporal knowledge from Telco MR data for Telco localization. With trivial overhead for feature engineering and domain expert knowledge (e.g., skillful feature design and post-processing steps), DeepLoc can train a spatiotemporal sequence prediction model for accurate localization.
- As two key components of DeepLoc, CNN learns local spatial features from each individual MR, and the proposed RecuLSTM extracts short and long-term dependencies from a series of sequential MRs. DeepLoc then can generate smooth and accurate trajectories.
- We evaluate DeepLoc against 10 counterparts by using 6 datasets collected from three representative (core, urban and suburban) areas in Shanghai, China. Our evaluation results indicate that DeepLoc significantly outperforms these counterparts.

The rest of this paper is organized as follows. Section 2 first introduces preliminaries. Next, Section 3 gives the problem setting and Section 4 reports the design of DeepLoc. After that, Section 5 evaluates DeepLoc and Section 6 reviews the related works. Section 7 finally concludes the paper.

2 PRELIMINARIES

2.1 Measurement Record (MR)

A MR indicates the connection states of a mobile device to nearby base stations in a Telco network. Table 1 gives an example 2G GSM MR collected by an Android phone. Each MR contains a unique number (IMSI: International Mobile Subscriber Identity), connection time stamp (MRTime), up to 6 nearby base stations (RNCID and CellID), signal measurements such as AsuLevel, SignalLevel, and RSSI (radio signal strength indicator). AsuLevel, i.e., Arbitrary Strength Unit (ASU), is an integer value proportional to the received signal strength measured by the mobile phone. SignalLevel indicates the power ratio (typically logarithm value) of the output signal of the device to the input signal. Among the 6 base stations, one of them is selected as the primary serving base station to provide communication and data services for the mobile device.

MRTime 2018/4/23 9:20	IMSI xxx	SRNC_ID 6188	BestCellID 26051	LCS_BIT 300
RNCID_1 6188	CellID_1 26051	AsuLevel_1 27	SignalLevel_1 4	RSSI_1 -74.5
RNCID_2 6188	CellID_2 27394	AsuLevel_2 10	SignalLevel_2 3	RSSI_2 -84.88
RNCID_3 6188	CellID_3 27377	AsuLevel_3 18	SignalLevel_3 4	RSSI_3 -85.13
RNCID_4 6188	CellID_4 27378	AsuLevel_4 12	SignalLevel_4 4	RSSI_4 -85.87
RNCID_5 6182	CellID_5 41139	AsuLevel_5 8	SignalLevel_5 3	RSSI_5 -88.88
RNCID_6 6188	CellID_6 27393	AsuLevel_6 9	SignalLevel_6 3	RSSI_6 -90.22

Table 1: An example of a 2G GSM MR

2.2 Telco Localization

The literature works on Telco localization are typically divided into three categories. First, *measurement-based methods* localize mobile devices by estimating absolute point-to-point distance or angles [12, 16, 17]. However, triangulation-based approaches which require signal strength measurements of at least three cells do not work well in 4G LTE networks, since 4G LTE MRs often contain measurements of at most two cells.

Second, *fingerprinting-based methods* [7, 9, 11] usually perform better than the measurement-based methods above by dividing an area of interest into small grids and representing each grid by an associated fingerprint [9]. A recent work NBL [11] assumed that signal strengths of each neighboring cell tower in the grid follow a Gaussian distribution. The online stage next adopts either Maximum Likelihood Estimation (MLE) or Weighted Average (WA) to localize mobile devices.

Lastly, some recent works [5, 27] adopt *machine learning (ML) techniques*, such as Random Forest, to build a training model to describe the correlations between the features extracted from MRs and the associated locations (e.g., GPS coordinates). The predicted locations could be either spatial regions (grid cells) or numeric GPS coordinates. ML techniques train the corresponding multi-classifiers [7, 11] or regression models [5, 27].

ML techniques significantly differ from measurement- and fingerprinting-based methods in terms of their application scenarios. That is, ML techniques work as Telco *backend* approaches on *historical* Telco big data that are maintained by Telco operators. ML techniques usually exhibit better results because ML techniques can leverage the rich information from various Telco data including (geo-tagged) MR data and configuration parameters (e.g., GPS coordinates) of base stations, Web log data and etc. Instead, measurement and fingerprinting-based methods usually work on *frontend* mobile devices as active localization methods, and may not use the rich data as ML techniques do.

3 PROBLEM SETTING

Let us consider that Telco operators have maintained a historical MR dataset. MRs usually do not contain the accurate locations of mobile devices (identified by IMSI in MRs), and we want to

annotate the MRs with accurate locations regarding the mobile devices in MRs. There are various ways to find location labels for such MRs. For example, when mobile users are using vehicle navigation services and switching on GPS receivers in the mobile devices, GPS coordinates of mobile devices are embedded in the URLs of mobile web logs. By extracting GPS coordinates from such URLs, we establish linkage between the extracted GPS coordinates and MRs with help of IMSI and timestamp. In this way, MRs are annotated by the linked GPS coordinates.

However, mobile users frequently switch off GPS receivers in mobile devices e.g., for energy saving. Mobile web logs contain very sparse GPS coordinates. Thus, the majority of Telco MR data cannot be annotated with the associated GPS coordinates. To tackle this issue, we could first train a machine learning model by using the sparse geo-tagged MRs as training data and next localize those MRs without GPS locations (as testing data).

Instead of using a single-point-based localization, we generate a trajectory of the locations predicted from a sequence of consecutive MRs. Using the sequence data, we expect that the localization model can capture the underlying spatial and temporal locality. The MR sequence data can be acquired as follows. Given a database $\mathcal{D}$ of MRs, we first group the MRs by IMSI and then sort the grouped MRs by timestamp. For each IMSI, we have a group of sorted MRs, which become the MR sequence data for this IMSI. Until now we define the following sequence-based localization problem.

PROBLEM 1. *Given a database $\mathcal{D}$ of training MR sequence data and associated GPS location labels, we train a machine learning model to precisely localize the testing MR sequence $\mathcal{T}$ with the objective to minimize the distance between the predicted locations and actual ones.*

Figure 1: Framework Overview

We highlight the general idea of DeepLoc as follows. For each MR within the input MR sequence, we first exploit CNN to estimate the scores for all possible locations as described in Section 4.1. The score indicates the chance for the location as the actual one. Next, for the entire input MR sequence, we leverage ReLuLSTM (an improved version of LSTM) to generate a smooth

trajectory. Each location within the trajectory is specially selected by ReLuLSTM from the list of estimated locations by CNN. In this way, ReLuLSTM and CNN work together to generate a smooth trajectory consisting of precisely estimated locations. In Figure 1, ReLuLSTM is fed with CNN's prediction results to inherit the advantages of CNN in processing spatial data, and to learn the temporal contextual knowledge at the same time. In this way, DeepLoc can capture the underlying spatial and temporal mobility patterns and greatly improve the prediction accuracy. We give the details of DeepLoc in the following section.

4 DESIGN OF DEEPLoc

In this section, we first describe the CNN-based single point localization for each MR, and next the ReLuLSTM-based sequence localization for an entire MR sequence.

4.1 CNN-based Single Point Localization

The CNN model performs a multi-classification prediction. Given numeric GPS coordinates, we divide the area of interest into grid cells (each cell has the fixed width and height $cw \times ch$) and every GPS location is then located inside a certain grid cell. In this way, the localization problem can be formulated as a multi-classification problem that considers grid cells as different classes: given an input MR record, the multi-classification problem identifies the most likely cell among all grid cells, and the centroid of the classified grid is set as the predicted location with respect to (w.r.t) this MR.

To make the CNN model work, we first process an individual MR. Recall that CNN is widely used for image recognition, and we borrow the idea from image recognition as follows. When an area of interest is divided into $M \times N$ grid cells (where M and N are the rows and columns of divided cells, respectively), we intuitively treat the area as an image, where each cell is treated as a pixel. Since a pixel is with image features such as color and brightness, each cell in our case is with various MR features such as *ASULevel*. In this way, for each MR, we build a three-dimensional (3D) feature tensor $M \times N \times F$, where F is the number of MR features.

By taking the example MR sample in Table 1 for illustration, we have $F = 3$ features, i.e., *ASULevel*, *SignalLevel* and *RSSI*, and Figure 2 shows the 3D feature tensor. First, for each BS_i in the MR, we use the GPS coordinates of BS_i to find the grid cell where BS_i is located. Given at most 6 base stations in a MR, we can identify the associated grid cells and generate a BS grid map for each MR. Note that it is highly possible for multiple base stations to be located inside the same cell (see that BS_3 and BS_4 belong to the same grid cell). Suppose that $B(m, n)$ base stations in a given MR are located inside the same cell w.r.t the m -th row and n -column, we compute the average of the f -th Telco signal feature values V_i^f as follows.

$$I_{m,n,f} = \frac{\sum_{i=1}^{B(m,n)} V_i^f}{B(m,n)} \quad (1)$$

Unfortunately, the 3D feature tensor could be rather sparse: each MR contains at most 6 base stations and the majority of the tensor is empty. A sparse tensor w.r.t a certain MR sample makes it hard to differentiate from others, leading to low prediction accuracy. To overcome this issue, for each MR inside an input sequence, we propose to build a 3D feature tensor from the consecutive MRs.

Figure 2: 3D input tensor of CNN

Specifically, within an input MR sequence, we use k consecutive MRs preceding the given MR. In this way, the 3D feature tensor w.r.t the given MR is actually built from the $(k + 1)$ MRs.

Until now for each input MR, we have built an associated 3D feature tensor. Meanwhile, for the GPS location w.r.t this MR, we have a grid cell ID where the GPS location is located. Based on the 3D feature tensor and the corresponding grid cell ID, we train a CNN-based multi-classifier that maps from a 3D feature tensor to its corresponding grid cell. During the prediction phase, when given a testing MR (together with its k preceding MRs in an input sequence), the multi-classifier computes a score (probability) for each class (i.e., grid cell). Suppose that the input h_0 of the CNN-based classifier is a tensor I , and the output of the m -th hidden layer in CNN is $h_m = f(W_m h_{m-1} + b_m)$, where W_m and b_m are the learned parameters by CNN and $f(\cdot)$ denotes the activation function. The output of the final hidden layer can be written as $f(I : (W, b))$. We train the CNN classifier to compute the conditional probability $p(c|I)$ by the *Softmax* function in Equation 2, where C is the total number of grid cells.

$$p(c|I) = \frac{e^{f_c(I:(W,b))}}{\sum_{i=1}^C e^{f_i(I:(W,b))}} \quad c \in \{1, 2, \dots, C\} \quad (2)$$

For a testing MR, the CNN-based multi-classifier generates a $C \times 1$ output vector. Each vector element is the score of a certain class (grid cell). A greater score indicates a higher chance for the grid cell to be classified as the actual one.

4.2 RecuLSTM-based Sequence Localization

4.2.1 Baseline Approach. For an input sequence with S MR samples, the CNN classifier generates a $C \times 1$ score vector for each MR sample, leading to $S \times C \times 1$ vectors for the sequence. Given a sequence of these S vectors $(x_1, \dots, x_n)$, a baseline solution is to exploit the basic LSTM structure to generate a trajectory of n

predicted locations by maximizing the probability of these n locations when given the S vectors:

$$\tilde{l}_1 \dots \tilde{l}_L = \arg \max_{l_1 \dots l_L} P(l_1, \dots, l_L | x_1 \dots x_L) \quad (3)$$

Nevertheless, the sampling rate of various MR sequences could differ between mobile devices. Let us consider two mobile devices with different sampling rates (for simplicity we assume that the two mobile devices follow the same mobility pattern). The MR sequence with a lower (resp. higher) sampling rate might indicate a larger (resp. smaller) spatial distance between the locations corresponding to two consecutive MR samples. How can a localization model adapt to various sampling rates?

4.2.2 RecuLSTM-based Approach. To answer this question, we propose a recursive LSTM structure, namely RecuLSTM. It consists of 1) multiple LSTM layers to recursively learn sequential features in terms of short- and long-term temporal dependencies and 2) a prediction layer that can infer a trajectory of predicted locations with help of the extracted short- and long-term temporal features.

Figure 3: RecuLSTM Network Structure

As shown in Figure 3, we give an example RecuLSTM structure that consists of two recursive layers and one prediction layer. Given a sequence of nine input MR samples (i.e., nine 3D feature tensors) $x_1, \dots, x_9$, in the 1st recursive layer, the CNN-based approach generates a $C \times 1$ score vector for each input tensor and thus totally nine score vectors. After receiving a window of $w_1 (= 3)$ vectors, the concatenation function (see the concat-1 in Figure 4) of the 1st layer generates a $C \times w_1$ 2D feature tensor as the output sequence.

As shown in Figure 4, the concat-1 function concatenates the window size w_1 of $C \times 1$ score vectors to generate a $C \times w_1$ 2D tensor. The output 2D tensor of the concatenation function can be intuitively interpreted as a short-term dependence of three input MR samples. The purpose of the concatenation operation (instead of using an average of these w_1 input vectors) is to ensure that those features (extracted CNN or LSTM in the underlying recursive layers) are not discarded and can still preserve the extracted knowledge

even when the number of recursive layer increases. Thus, with help of this concatenation operation, the temporal correlations of the input sequence can be comfortably captured, regardless of MR sampling rate or the underlying mobility patterns.

When the $C \times w_1$ 2D tensor (sequence) of the 1st recursive layer are transmitted to the 2nd layer, the LSTM in the 2nd layer generates a $C' \times w_1$ 2D feature tensor, and the concatenation function (see *concat-2* in Figure 4) processes w_2 input $C' \times w_1$ 2D tensors and generates a $C' \times (w_1 \times w_2)$ 2D feature tensor. Note that the window size w_2 in the 2nd layer does not necessarily be equal to the one w_1 in the 1st layer. Following the example above with 9 input MRs, we tune $w_1 \times w_2 = 9$, so that each of the 9 input MR samples is mapped to a 1D $C' \times 1$ feature vector within the $C' \times (w_1 \times w_2)$ 2D feature tensor.

Finally, the prediction layer in RecuLSTM consists of an LSTM layer and a fully connected (FC) layer. The LSTM layer processes the $C' \times (w_1 \times w_2)$ 2D feature tensor and generates a $2 \times (w_1 \times w_2)$ 2D tensor, where 2 corresponds to the number of features in a GPS coordinate (i.e. longitude and latitude). The FC layer trains a regression model that processes the generated $2 \times (w_1 \times w_2)$ 2D feature tensor to predict the MR locations. Using a regression model is mainly because the locations are represented by numeric GPS longitude and latitude coordinates. As show in Figure 3, when given the $2 \times (w_1 \times w_2)$ 2D feature tensor with the associated $(w_1 \times w_2) = 9$ locations, the training model essentially maps the $2 \times (w_1 \times w_2)$ tensor to a trajectory consisting of these 9 locations. After the regression model is trained, it can be used to predict the GPS locations from an input feature tensor.

4.2.3 Optimization. In this section, we give three tricks to optimize DeepLoc. First, the CNN approach could lead to a great size C . For example we could divide an area of interest into hundreds or even thousands of grid cells, e.g., $C = 1000$. Given the window size $w_1 = 3$ in the 1st recursive layer of RecuLSTM, the output 2D tensor of the concatenation function in this layer is a 1000×3 2D tensor (i.e., a matrix). Here, the problem is that the matrix contains only several non-zero elements (i.e., the scores w.r.t grid cells). To this end, to avoid the curse of dimensionality, we add a FC layer to the output of CNN for dimensionality reduction [18], and the processing result of this FC layer then becomes the input of the *concat-1* function.

Second, we tune the window sizes such as w_1 and w_2 as follows. For an input MR sequence, we count those MR samples sharing the same serving BS (denoted by C_{1_bs}) and those MRs sharing both the same serving BS and at least one neighbouring BS (denoted by C_{2_bs}), and empirically set $w_1 = C_{2_bs}$ and $w_1 \times w_2 = C_{1_bs}$. In this way, setting such window sizes as C_{1_bs} and C_{2_bs} can roughly capture the underlying mobility pattern with help of base stations.

Finally, in Figure 5, we tune the parameters of the fully connected network in the prediction layer of RecuLSTM. 1) In terms of the number of hidden layers in this FC layer, we set it as the number of grid cells where the GPS locations of the training data set are located. In this way, we can greatly reduce the number of hidden layers. 2) We initiate the weight $W^{(1)}$ between the input layers and the hidden layers by the prediction score of CNN. Specifically, the output of this FC layer is computed by $o = f_c(XW + b)$, where $W = W^{(1)}W^{(2)}$ and b are the weight and bias, respectively. Since

we have tuned the number of hidden layers, the weights $W^{(1)}$ are initiated as follows.

$$W_{i,j}^{(1)} = \text{score}(x_i^j) = p(\text{gird}_j|x_i) \quad (4)$$

where $p(\text{gird}_j|x_i)$, i.e., the j -th item in the output vector of CNN for the i -th input MR sample x_i , is computed by Equation 2. The weight initialization above offers the following benefit: the FC layer can re-use the prediction result of CNN to speed up the convergence of FC loss and to improve the FC prediction accuracy.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Data Sets. We evaluate the performance of DeepLoc using the data sets collected at three representative areas in Shanghai, China: a core business area *Xuhui*, an urban area *Siping* and a suburban area *Jiading*. The physical distance of Jiading $\sim$ Siping, Jiading $\sim$ Xuhui and Siping $\sim$ Xuhui is around 31 km, 37 km and 15 km, respectively. The data sets (including both MR samples and GPS location labels) at Xuhui were provided by one of the largest Telco operators in China, and the data sets at Jiading and Siping were collected by own-developed Android mobile app. When mobile users are moving around in outdoor road networks to collect MR samples, we meanwhile switch on GPS receivers on Android mobile phones to collect current GPS coordinates. Since the GPS coordinates collected by mobile phones may contain noises, we thus employ the map-matching technique [26] to mitigate the effect of noises.

	Jiading	Siping	Xuhui
Num. of samples (2G/4G)	15954/10372	6723/4953	13404/7755
Route Len (2G/4G) in km	94.1/52.1	24.6/15.5	26.4/12.7
Sampling Period (sec)	1	1	2
Num. of BSs per km ² (2G/4G)	5.67/8.42	7.96/10.35	8.32/12.44

Table 2: Statistics of Used Data Sets

Table 2 summarizes the used data sets. In each area, we have two data sets containing MRs collected from both 2G GSM and 4G LTE networks, respectively. Each sample in the data sets includes one MR and the associated GPS coordinates. Due to the limitations of the Android APIs in the 4G networks, each 4G MR sample collected in Jiading and Siping contains only one (serving) base station without the information about other neighboring base stations. Yet the 4G samples in Xuhui still contain up to 6 base stations.

Besides, we acquire the GPS longitude and latitude coordinates of every BS included in the our datasets from a base station database, and use them as additional features of MR samples. However, those parameters such as antenna height and location angle are unavailable due to the limited information provided by the database. We believe that these parameters, if available, could improve DeepLoc.

5.1.2 Counterparts. In Table 3, we compare DeepLoc against 4 literature works given by [11, 13, 27] and 6 DNN-based approaches. First, regarding the localization result, the counterparts are either single-point-based or sequence-based approaches. In addition, in terms of used models, the counterparts

Figure 4: Concatenation Function

can be either the fingerprinting-based, the traditional machine learning-based (e.g., Random Forests and HMM) or deep neural network-based models (e.g., CNN, LSTM, ConvLSTM [14], stacked LSTM, Seq2Seq [15], RecuLSTM and DeepLoc). Here, RecuLSTM is a simplified version of DeepLoc by removing the used CNN model. For the literature work [27], the localization model can be either only a single layer random forest, namely RaF, or a two-layer random forests (we use the name CCR to be consistent with [27]). Here, the 2nd layer random forest in CCR uses the contextual features e.g., moving speed, acquired from the predicted result of the 1st layer. We thus say that CCR is an implicit sequence-based approach due to the contextual features.

For evaluation, we select 80% and 20% MR samples as the training and testing datasets, respectively. During the training phase, we adopt the 10-fold cross validation to tune parameters. After the models are trained, we use them to calculate the locations of testing MR samples. By computing the Euclidian distance between the predicted locations and the corresponding ground truth (i.e., the GPS coordinates of MR samples) as the prediction error, we choose median error, mean error and top 90% error (by sorting errors of validation data in an ascending order) as evaluation metrics.

Counterpart	Description	Approach
NBL	Recent fingerprinting method [11]	Single Point
RaF	1-layer random forest regression [27]	Single Point
CCR	2-layer random forest regression [27]	Single Point
HMM	HMM + particle filtering [13]	Sequence
CNN	CNN using 4 convolution layers	Single Point
LSTM	basic LSTM	Sequence
StackLSTM	two-layer stacked LSTM network	sequence
Seq2Seq	sequence to sequence model [15] by LSTM	sequence
ConvLSTM	convolutional LSTM [14]	Sequence
RecuLSTM	Proposed Recursive LSTM	Sequence
DeepLoc	CNN + RecuLSTM	Sequence

Table 3: DeepLoc’s Counterparts

Figure 5: FC Net in Prediction Layer

5.2 Baseline Study

In Table 4, we compare the prediction errors (median, mean and 90% errors) of the 11 approaches in three areas (Jiading, Siping, and Xuhui). For each area, we have a 2G data set AND a 4G data set, and thus totally 6 datasets. In addition, for two single-point-based approaches such as CCR and NBL, we particularly perform post-processing steps on their predicted locations by using the particle filtering (PF) technique, namely CCR_F and NBL_F, respectively. Following [13], we set the parameters used for PF as follows: 1) 500 particles, 2) 100 times of recurrent processes and 3) Gaussian distribution for system and observation noises. Moreover, with help of extra road network information, we exploit the map-matching technique [26] to project the predicted locations by CCR, NBL and HMM and DeepLoc onto the road networks, and name them by CCR_M, etc. From this table, we have the following findings.

First, DeepLoc offers the least error among these 11 approaches on 6 MR datasets only except those 4G MR samples in Siping and Jiading areas which contain only one BS information in each sample. For example, in case of 2G data sets from Jiading, DeepLoc reduces the median error by 55.4% compared with CCR. In addition, DeepLoc slightly outperforms the 6 other DNN-based approaches (i.e. CNN, LSTM, StackLSTM, and SeqtoSeq, ConvLSTM, and RecuLSTM). Such results indicate that DeepLoc outperforms both the state-of-the-art approaches and alternative deep-learning approaches.

Second, we are interested in the result of two post-processing techniques. At first, the particle filters-based post-processing CCR_F and NBL_F lead to slightly better results than the original ones. However, DeepLoc still outperforms CCR_F and NBL_F, mainly because particle filtering can correct individual outliers but not multiple continuous outliers which could entirely shift away from ground truth locations. Secondly, the map matching-based approaches CCR_M, NBL_M and HMM_M leverage road network constraints and lead to much lower errors than particle filters. Nevertheless, the majority of those map matching-based results still cannot compete with DeepLoc, let alone DeepLoc_M. These numbers verify the power of DeepLoc to generate smooth and accurate location trajectories.

Dataset	Jiading(2G)			Jiading(4G)			Siping(2G)			Siping(4G)			Xuhui(2G)			Xuhui(4G)		
	Median	Mean	90%	Median	Mean	90%	Median	Mean	90%	Median	Mean	90%	Median	Mean	90%	Median	Mean	90%
NBL	57.4	67.2	300.9	53.7	66.8	318.6	63.0	298.3	43.2	64.9	256.7	45.9	59.0	240.7	32.2	52.7	191.2	
NBL_F	40.8	59.2	214.7	38.6	55.7	207.8	37.4	56.1	209.3	36.7	57.2	209.3	37.2	51.4	186.5	29.3	48.5	171.3
NBL_M	38.9	50.7	165.8	36.1	49.2	149.5	30.5	45.1	141.8	31.1	46.6	140.6	32.3	45.6	155.1	30.6	44.8	142.9
RaF	38.3	48.3	168.9	38.5	47.2	158.9	41.1	47.3	158.8	35.1	44.4	105.8	35.6	43.4	137.7	32.2	41.5	104.4
CCR	34.1	43.2	142.3	30.2	44.5	145.9	37.5	42.8	139.5	29.5	40.6	101.2	30.0	40.2	125.2	20.0	34.1	98.3
CCR_F	29.3	40.8	130.7	27.7	41.2	136.3	30.1	39.6	128.4	24.5	37.4	95.6	25.3	34.2	117.3	18.6	32.1	91.9
CCR_M	22.1	36.6	99.1	17.8	32.9	105.5	23.8	33.5	93.9	16.8	32.1	79.9	18.4	34.5	92.6	13.6	28.4	86.7
HMM	38.9	59.9	172.8	42.1	53.6	188.4	37.2	51.4	160.3	30.7	48.2	145.6	35.3	42.7	148.0	28.6	39.1	129.5
HMM_M	22.3	41.7	109.2	24.6	40.3	118.3	25.0	36.6	121.0	19.7	35.1	99.3	21.2	34.8	113.4	18.3	31.9	103.5
CNN	18.2	42.2	82.7	25.4	47.2	92.3	18.9	45.4	79.1	24.1	44.3	96.4	17.6	43.1	73.0	15.9	39.9	67.4
LSTM	26.5	51.3	89.2	24.3	50.6	88.9	25.2	49.6	86.3	24.3	48.2	88.5	22.6	48.2	90.1	21.4	44.1	82.3
StackLSTM	26.7	56.1	93.1	25.4	54.2	92.3	25.4	54.2	89.9	24.5	50.8	84.4	23.7	44.8	86.3	21.9	43.3	84.9
Seq2Seq(LSTM)	25.4	50.6	85.3	24.2	50.1	81.7	23.4	48.9	84.9	22.7	47.3	83.5	22.5	43.2	84.7	21.9	42.9	81.8
ConvLSTM	28.5	59.3	129.3	27.3	58.1	124.5	27.7	57.3	120.4	25.6	55.7	117.2	25.3	52.5	113.2	24.4	50.6	107.8
RecurLSTM	17.9	42.6	67.7	17.0	38.7	63.5	18.5	43.8	70.4	16.9	37.4	62.5	18.8	37.6	64.5	14.6	34.6	62.1
DeepLoc	15.2	37.8	63.2	16.4	39.6	66.4	16.7	33.2	62.17	17.4	35.7	60.2	15.7	34.4	62.5	13.7	31.2	59.3
DeepLoc_M	13.9	33.5	60.1	13.7	34.8	61.4	13.7	31.5	60.4	14.8	33.7	58.6	14.2	32.8	60.3	13.1	29.9	57.5

Table 4: Baseline Experiment: Localization Errors of 9 Approaches

Thirdly, we compare the top-90% errors among the 11 approaches. Still on Jiading 2G dataset, DeepLoc outperforms CCR with 60.3% lower top-90% error and 55.4% lower median error. Similar results apply to the other literature works. These numbers indicate that DeepLoc, due to the used sequence model, can effectively mitigate the outliers among the predicted locations.

Fourthly, we compare the results of the 8 sequence-based approaches against 3 single-point-based ones. For example, DeepLoc outperforms CNN by 16.8% smaller median error on Jiading 2G data set. Note that CNN works better than all other models except DeepLoc among most datasets except Jiading and Siping 4G datasets. It is mainly because CNN can comfortably capture the spatial features in case that the MR samples contain enough BSs. In case there is only one BS in a MR sample of the Jiading and Siping 4G datasets, CNN cannot consistently lead to a good result. In addition, though CCR and RaF are both Random Forest-based approaches, CCR, due to the used contextual features such as moving speed, leads to better result than RaF by 10.9% smaller median error. Such results verify that the sequence models, no matter implicit or explicit use of contextual features, could lead to better results than the original single-point-based approaches.

Fifthly, among the 7 deep neural network-based approaches, ConvLSTM (see Section 6 for an introduction of ConvLSTM) suffers from the highest errors. It might be mainly because ConvLSTM is typically used to solve flow prediction (e.g., precipitation nowcasting [14] and traffic accident [21]) for the sequence data involving explicit spatiotemporal information. However, our MR records do not contain accurate locations, which are just the objective of our model (Note that the GPS locations of BSs are used as additional features, but not the locations of MR records). Thus, ConvLSTM may not properly solve our problem though ConvLSTM also utilizes the power of both CNN and LSTM.

Sixthly, we compare DeepLoc against the three LSTM variants including StackLSTM, Seq2Seq and RecuLSTM. Simply stacking the LSTM layers for deeper neural networks might not greatly decrease prediction errors, instead incurring the worst result. Though slightly better than StackLSTM, Seq2Seq cannot process the MR sequence in a hierarchical manner and still cannot compete with DeepLoc. As a simplified version of DeepLoc without using the CNN model, RecuLSTM does not capture the spatial features offered by the CNN model and thus leads to a worse result than DeepLoc.

Finally, Table 4 is consistent with the previous works CCR and NBL: for the same area e.g., Xuhui, 4G MR samples typically lead to better result than 2G samples; and for the same Telco networks, the

results of the Xuhui 2G dataset are better than those of Siping and Jiading 2G datasets. Such a result is consistent with the statistics in Table 2: higher BS deployment density in 4G networks and core areas can achieve better result than those in 2G and rural areas. We also note that due to only one BS per MR in Jiading and Siping 4G datasets, their errors are even higher than those 2G datasets. Such result verifies the importance of having measurements of multiple base stations for precise localization.

5.3 Sensitivity Study

Due to space limit, we choose Jiading 2G dataset (with the most data samples and largest area) for sensitivity study.

(1) **CNN Configuration:** Recall that the CNN component in DeepLoc is important to capture spatial features and even can be used as a single-point model. Thus, we are interested in the performance of DeepLoc by varying the configurations of the CNN components. Table 5 gives the prediction errors of 5 configurations, in which the prediction layers follow the same configuration “FC256-FC256-Softmax”, i.e., a fully connected network with 2 hidden layers and the softmax for regression. In this table, the 1st configuration means a $100 \times 100 \times 3$ input MR 3D tensor and 4 convolution layers (three of them contain the associated maximal pooling) plus the aforementioned prediction layer.

We first find that the $100 \times 100 \times 3$ input MR 3D tensor in the 1st configuration performs better than the $50 \times 50 \times 3$ tensor in other configurations, due to the finer-grained grid mapping. Secondly, the four-convolution-layer models in the 2nd and 3rd configurations are generally better than the three-convolution-layer model in the 5th configuration. However, having more than four convolutional layers in the 4th configuration does not significantly reduce the errors. Thirdly, a greater filter kernel (9×9 in the 3rd configuration) leads to higher errors than the one with a smaller filter kernel such as 5×5 in the 2nd configuration. It is mainly because a greater kernel leads to more loss of BS spatial information especially when those BSs in a MR sample are not very far away. As a summary, when compared with the 5th configuration, the 2nd one even with one less convolution layer can achieve acceptable errors.

(2) **Number of Recursive Layers:** We study how the number of recursive layers in the RecuLSTM component affects prediction errors by varying the number of recursive layers from 0 to 4. Here, DeepLoc with no recursive layer contains a fully connected layer alone, leading to the highest error. A greater number of recursive layers give better results, though after 2 recursive layers, deeper models do not greatly improve the errors. Therefore, using 2

Configuration	Input Tensor	Conv	Median	Mean	Top-90%
1	$50 \times 50 \times 3$	Conv1 5*256-Maxpool-Conv2 5*128-maxpool-Conv3 3*64-Maxpool-Conv4 3*32	25.7	49.9	97.8
2	$100 \times 100 \times 3$	Conv1 5*256-Maxpool-Conv2 5*128-Maxpool-Conv3 3*64-Maxpool-Conv4 3*32	18.2	42.2	82.7
3	$100 \times 100 \times 3$	Conv1 9*256-Maxpool-Conv2 5*128-Maxpool-Conv3 3*64-Maxpool-Conv4 3*32	23.1	47.5	89.5
4	$100 \times 100 \times 3$	Conv1 5*512-Maxpool-Conv2 5*256-Maxpool-Conv3 5*128-Maxpool-Conv4 3*64-Maxpool-Conv5 3*32	18.0	41.7	83.1
5	$100 \times 100 \times 3$	Conv1 5*256-Maxpool-Conv2 5*128-Maxpool-Conv3 3*64-Maxpool	24.3	48.6	99.2

Table 5: Prediction Errors of various network configurations

Figure 6: Number of Recursive Layers

Figure 7: Effect of Prediction Layer

recursive layers can achieve comfortable results with a short execution duration.

(3) Optimization of Prediction Layer: Recall that in Section 4.2, we have applied two optimization techniques (tuning the number of hidden layers and initiating the weight $W^{(1)}$ by the processing result of CNN). To study the effect of the optimization techniques, we evaluate DeepLoc (after optimization) against the one without optimization. For the two versions of DeepLoc, we manually vary the number of used epochs (or iterations) to train RecuLSTM in DeepLoc and see when DeepLoc becomes converged. As shown in Figure 7, after 200 epoches, the prediction error of DeepLoc with optimization remains rather stable meanwhile with much smaller prediction error than the one without optimization. It verifies that the optimization techniques do speed up convergence and also lead to smaller errors.

(4) Concatenation Function: Recall that we perform a concatenation operation on the input sequence vectors (or tensors). Alternatively, such operations as *mean*, *max* and *last* can be applied to these vectors/tensors. Here, the *last* function indicates that among the input sequence consisting of multiple vector/tensors, the last vector/tensor is directly taken as the output. As shown in this figure, the concatenation operation leads to the best result. It is mainly because this operation can preserve the exactly entire information from the input sequence. In addition, the *last* operation leads to a slightly better result than the *max* and *mean* Operations. It might be because the final prediction result is highly possible related to the last hidden state of each LSTM.

(5) Sampling Interval: For the raw Jiading 2G dataset, we randomly select some samples by varying the sampling interval from every 2 seconds to every 30 seconds. Figure 9 gives the prediction errors of 4 deep neural network-based approaches (we do not choose ConvLSTM, StackLSTM, and Seq2Seq because

ConvLSTM incurs the worst results as mentioned above, and both StackLSTM and Seq2Seq are the variants of LSTM and they have the similar prediction errors compared with LSTM). Given a higher sampling interval (and thus more sparse data samples), all approaches incur higher prediction errors. Secondly, the (single point-based) CNN approach performs better than the (sequence-based) LSTM approach for a low sampling interval, and yet worse for a high sampling interval. It is mainly because the CNN model can comfortably capture the spatial features locally from dense samples and instead the LSTM approach can extract better temporal dependencies even from sparse samples. Finally, RecuLSTM can capture short/long-term dependency from either sparse or dense MR sequences, and DeepLoc ensembles the power of CNN and RecuLSTM, leading to the least errors.

(6) Sliding Window Size: Recall that DeepLoc with 2 recursive layers leads to the best prediction results. Thus, we study how the window sizes in these 2 layers affect the prediction errors of DeepLoc. By enumerating the various window size from 1 to 10, our experiments empirically find that $w_1 = 8$ and $w_2 = 3$ can achieve the best result. These numbers are roughly consistent with the empirical setting in Section 4.2.2: we set $w_1 = C_{2_bs}$ and $w_1 \times w_2 = C_{1_bs}$. Based on Jiading 2G dataset, we compute the average $C_{2_bs} = 8.33$ and $C_{1_bs} = 27.53$, leading to $w_1 = 8.33$ and $w_2 = 3.3$.

5.4 Visualization

Finally Figure 10 visualizes the prediction results of DeepLoc and three existing Telco localization approaches (HMM, CCR and NBL) on a randomly selected small area from Jiading 2G data set. In addition, with help of the extra road network, we perform the map-matching technique [26] on the trajectories and plot the map-matching trajectories in this figure.

Figure 8: Concatenation Function

In case of no map-matching, the trajectory predicted by DeepLoc is the closest to the ground truth in terms of horizontal moving directions and vertical shift out of road segments. Yet NBL leads to the most significant shift in both horizontal and vertical directions. HMM and CCR, though with better results than NBL, cannot compete with DeepLoc still due to some minor outliers.

In terms of the map-matching results of these approaches, this post-processing technique can greatly improve CCR and NBL to generate smooth trajectories. However, the map-matching technique cannot fully overcome the overall shift along the horizontal moving direction, which has been clearly visualized in this figure. Consistent with Section 5.2, DeepLoc can still lead to the acceptable result, even without those post-processing steps such as map-matching.

6 RELATED WORK

Trajectory Recovery: Some literature works focus on calculating single-point locations, and others on estimating a sequence (or trajectory) of multiple locations. For example, AT&T researchers [13] proposed to apply HMM for trajectory generation and next apply particle filters to smooth the trajectory. CCR [27] designed temporal and spatial contextual features (e.g., moving speeds, time gap, etc.) based on domain expert knowledge and feature engineering expertise to greatly improve prediction accuracy. In addition, when given a set of (GPS) locations, some works proposed to recover an entire trajectory from such locations. For example, with help of road network constraints, map-matching techniques and their variants [26] can project the locations onto road networks to correct outlier locations (for example those noisy GPS points). Also some works use various filtering techniques (e.g., particle filters) [13, 19] to smooth the trajectories.

Mobile Data Processing by Deep Learning: Deep neural networks have been widely used in various scenarios for processing mobile data in recent years [22]. First, for the time-series data generated by mobile sensors (e.g., accelerometers and gyroscopes), some works were proposed to address data noise issues [8, 20]. In [8], the authors proposed an outlier detection framework that incorporates several deep neural network based autoencoders. DeepSense [20] targets to improve the accuracy of

Figure 9: Sampling Interval

estimating signals from noisy measurements in mobile sensing applications by integrating convolutional and recurrent neural networks.

ConvLSTM [14] and its variants such as Hetero-ConvLSTM [21] have been used in flow predictions when using spatiotemporal sequence data. ConvLSTM is a variation of LSTM to handle spatiotemporal prediction problem for precipitation nowcasting. Each input feature of a ConvLSTM is a 3D spatio-temporal tensor, where the first two dimensions (row and column in the 3D tensor) are the spatial dimensions. Comparing with the original LSTM, the input-to-state and state-to-state transitions of the ConvLSTM cell involve the convolutional operations that output 3D tensors. To study the traffic accident prediction problem using ConvLSTM, the authors in [21] proposed a Hetero-ConvLSTM framework to address the spatial heterogeneity challenge in the data over the state of Iowa. Our work DeepLoc differs from ConvLSTM [14] because MR data do not explicitly contain accurate spatial information which is just our prediction objective (note that the GPS locations of base stations are not the locations of the mobile devices in MRs). Moreover, in terms of network structure, we utilize the prediction results of CNN, but not a convolution operator in input-to-state and state-to-state transitions.

7 CONCLUSION

In this paper, to generate a smooth trajectory connecting precisely predicted outdoor locations, the proposed framework DeepLoc ensembles the power of CNN and RecuLSTM. Our extensive evaluation validates that DeepLoc greatly outperforms four state-of-the-art Telco localization approaches and six alternative deep neural network-based approaches on six 2G/4G datasets collected in three representative areas in Shanghai. The promising result of DeepLoc inspires the following future work. For example, we are interested in the use of those generated trajectories from Telco MR data for mobility pattern analysis, and the potentials of applying DeepLoc to forthcoming 5G networks.

ACKNOWLEDGE

This work is partially supported by National Natural Science Foundation of China (Grant No. 61572365, 61772371, and

Figure 10: Visualization of Prediction Results (Map Scale 1.5:10000). Black: ground truth; Red: predicted locations.

61672380), The Fundamental Research Funds for the Central Universities, and Academy of Finland under grant number 317432 and 318937.

REFERENCES

- [1] Google maps for mobile. <http://www.google.com/mobile/maps>.
- [2] R. A. Becker, R. Cáceres, K. Hanson, S. Isaacman, J. M. Loh, M. Martonosi, J. Rowland, S. Urbanek, A. Varshavsky, and C. Volinsky. Human mobility characterization from cellular network data. *Commun. ACM*, 56(1):74–82, 2013.
- [3] R. A. Becker, R. Cáceres, K. Hanson, J. M. Loh, S. Urbanek, A. Varshavsky, and C. Volinsky. A tale of one city: Using cellular network data for urban planning. *IEEE Pervasive Computing*, 10(4):18–26, 2011.
- [4] C. Costa and D. Zeinalipour-Yazti. Telco big data: Current state & future directions. In *IEEE MDM 2018*, pages 11–14, 2018.
- [5] Y. Huang, W. Rao, F. Zhu, N. Liu, M. Yuan, J. Zeng, and H. Yang. Experimental study of telco localization methods. In *IEEE MDM 2017*, pages 299–306, 2017.
- [6] M. Ibrahim and M. Youssef. A hidden markov model for localization using low-end GSM cell phones. In *IEEE ICC 2011*, pages 1–5. IEEE, 2011.
- [7] M. Ibrahim and M. Youssef. Cellsense: An accurate energy-efficient GSM positioning system. *IEEE Trans. Vehicular Technology*, 61(1):286–296, 2012.
- [8] T. Kieu, B. Yang, and C. S. Jensen. Outlier detection for multidimensional time series using deep neural networks. In *IEEE MDM 2018*, pages 125–134. IEEE, 2018.
- [9] James J. Caffery and G. L. Stuber. Overview of radiolocation in cdma cellular systems. *IEEE Communications Magazine*, 36(4):38–45, Apr. 1998.
- [10] I. Leontiadis, A. Lima, H. Kwak, R. Stanojevic, D. Wetherall, and K. Papagiannaki. From cells to streets: Estimating mobile paths with cellular-side data. In A. Seneviratne, C. Diot, J. Kurose, A. Chaintreau, and L. Rizzo, editors, *ACM CoNEXT 2014*, pages 121–132. ACM, 2014.
- [11] R. Margolis, R. A. Becker, S. D. Byers, S. Deb, R. Jana, S. Urbanek, and C. Volinsky. Can you find me now? evaluation of network-based localization in a 4g LTE network. In *IEEE INFOCOM 2017*, pages 1–9, 2017.
- [12] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal. Locating the nodes: cooperative localization in wireless sensor networks. *IEEE Signal Processing Magazine*, 22(4):54–69, 2005.
- [13] A. Ray, S. Deb, and P. Monogioudis. Localization of LTE measurement records with missing information. In *IEEE INFOCOM 2016*, pages 1–9, 2016.
- [14] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *NIPS 2015*, pages 802–810, 2015.
- [15] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS 2014*, pages 3104–3112, 2014.
- [16] S. Swales, J. Maloney, and J. Stevenson. Locating mobile phones and the us wireless e-911 mandate. In *Novel Methods of Location and Tracking of Cellular Mobiles and Their System Applications (Ref. No. 1999/046)*, IEE Colloquium on, pages 2–1. IET, 1999.
- [17] R. M. Vaghefi, M. R. Gholami, and E. G. Ström. Rss-based sensor localization with unknown transmit power. In *IEEE ICASSP 2011*, pages 2480–2483. IEEE, 2011.
- [18] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang. Modeling trajectories with recurrent neural networks. In *IJCAI 2017*, pages 3083–3090, 2017.
- [19] H. Wu, W. Sun, B. Zheng, L. Yang, and W. Zhou. CLSTERS: A general system for reducing errors of trajectories under challenging localization situations. *IMWUT*, 1(3):115:1–115:28, 2017.
- [20] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. F. Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, editors, *WWW 2017*, pages 351–360. ACM, 2017.
- [21] Z. Yuan, X. Zhou, and T. Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In Y. Guo and F. Farooq, editors, *ACM SIGKDD 2018*, pages 984–992. ACM, 2018.
- [22] C. Zhang, P. Patras, and H. Haddadi. Deep learning in mobile and wireless networking: A survey. *CoRR*, abs/1803.04311, 2018.
- [23] Y. Zhang, W. Rao, and Y. Xiao. Deep neural network-based telco outdoor localization. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, SenSys 2018, Shenzhen, China, November 4-7, 2018*, pages 307–308, 2018.
- [24] K. Zhao, M. Musolesi, P. Hui, W. Rao, and S. Tarkoma. Explaining the power-law distribution of human mobility through transportation modality decomposition. *Scientific reports*, 5:9136, 2015.
- [25] P. Zhao, C. Zhang, Y. Huang, W. Rao, F. Zhu, N. Liu, J. Geng, M. Yuan, and J. Zeng. Demo abstract: Telco localization techniques. In *2017 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Atlanta, GA, USA, May 1-4, 2017*, pages 970–971, 2017.
- [26] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: Concepts, methodologies, and applications. *ACM TIST*, 5(3):38:1–38:55, 2014.
- [27] F. Zhu, C. Luo, M. Yuan, Y. Zhu, Z. Zhang, T. Gu, K. Deng, W. Rao, and J. Zeng. City-scale localization with telco big data. In *ACM CIKM 2016*, pages 439–448, 2016.