DMM: Fast Map Matching for Cellular Data

Zhihao Shen
Xi'an Jiaotong University
Xi'an, China
szh1095738849@stu.xjtu.edu.cn

Wan Du
University of California, Merced
Merced, USA
wdu3@ucmerced.edu

Xi Zhao
Xi'an Jiaotong University
Xi'an, China
Zhaoxi1@mail.xjtu.edu.cn

Jianhua Zou
Xi'an Jiaotong University
Xi'an, China
jhzou@sei.xjtu.edu.cn

ABSTRACT

Map matching for cellular data is to transform a sequence of cell tower locations to a trajectory on a road map. It is an essential processing step for many applications, such as traffic optimization and human mobility analysis. However, most current map matching approaches are based on Hidden Markov Models (HMMs) that have heavy computation overhead to consider high-order cell tower information. This paper presents a fast map matching framework for cellular data, named as DMM, which adopts a recurrent neural network (RNN) to identify the most-likely trajectory of roads given a sequence of cell towers. Once the RNN model is trained, it can process cell tower sequences as making RNN inference, resulting in fast map matching speed. To transform DMM into a practical system, several challenges are addressed by developing a set of techniques, including spatial-aware representation of input cell tower sequences, an encoder-decoder framework for map matching model with variable-length input and output, and a reinforcement learning based model for optimizing the matched outputs. Extensive experiments on a large-scale anonymized cellular dataset reveal that DMM provides high map matching accuracy (precision 80.43% and recall 85.42%) and reduces the average inference time of HMM-based approaches by 46.58×.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks*; • **Networks** → **Location based services**; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools*.

KEYWORDS

Map Matching, Cellular Data, Neural Networks

ACM Reference Format:
Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. 2020. DMM: Fast Map Matching for Cellular Data. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3421461

MobiCom '20, September 21–25, 2020, London, United Kingdom

ACM ISBN 978-1-4503-7085-1/20/09…$15.00
https://doi.org/10.1145/3372224.3421461

1 INTRODUCTION

Cellular data is a set of location sequences of cell towers, with which a mobile phone has been associated. It has been processed for many applications [1–9], including transportation analysis [4, 5] and human mobility analysis [6–8, 10, 11]. An essential processing step of all these applications is map matching that transforms a cell tower sequence into the most-likely road trajectory on a road map. Efficient map matching algorithms are necessary for providing fast processing of large-scale cell tower sequences and minimizing computational resource consumption (e.g., power, storage and computation). For example, transportation analysis applications that can estimate road traffic trends using cellular data require to match the cellular data on a road map continuously in a timely manner.

Many map matching approaches [12–19] have been proposed. Most of them use Hidden Markov Models [12–17] as their backbones, relying on Markov assumption to simplify the problem, i.e., the probability distribution of next roads only depends on the current road and not on the past or future road. However, human mobility on a road map is non-Markovian [20], especially when people have a specific destination. Moreover, HMM-based approaches assume to follow the shortest path between the surrounding roads of two consecutive cell towers, which leads to extensive search of the shortest paths during online inference. This incurs high computational overhead, especially for low-sampling-rate cell tower sequences. For a sequence of only 7 cell towers with 68 possible road candidates around each cell tower, HMM takes about 32,368 ($68^2 \times 7$) computations of the shortest paths, corresponding to $\sim$ 82.5 seconds of running time. To provide fast map matching, SnapNet [12] increases the sampling rate of cell tower sequences by interpolating some locations between two adjacent cell towers. SnapNet works well for moving trajectories on highways; whereas it is hard to perform accurate interpolation in urban areas where have a lot of possible routes to connect two locations. As a consequence, simple interpolation degrades the matching accuracy in urban areas.

In this paper, we propose a novel map matching framework for cell tower sequences, named as DMM. It is based on a recurrent neural network (RNN) [21] that takes a sequence of cell tower locations as input and infers a trajectory composed of road segments. The model directly learns the mapping between cell towers and roads based on training data. This avoids the extensive computation of the

shortest paths during online inference, and thus reduces computation overhead. One RNN inference for a cellular data sequence is fast, e.g., ∼1 second in our implementation for a sequence of 12 cell towers. In addition, RNN-based model is expressive of representing the sequence of cell towers by a hidden vector during inference. This allows to consider multiple previous roads for inferring the next road segment, but not just the last road. To transform DMM into a practical system, we tackle a set of challenges.

Deep neural network based models require vector representations for input cell towers. A classic approach is to use a binary vector to represent a cell tower, in which all bits are '0' except one '1', referring to the specific cell tower. However, this approach cannot capture spatial proximity among cell towers. As a consequence, learned map matching patterns of a cell tower cannot be utilized to its adjacent cell towers. To enable accurate map matching, DMM designs a high-quality, low-dimensional representation model. This enables to share a similar representation for spatially-close cell towers, and thus generates similar map matching results.

Intuitively, we design our map matching model based on classic RNN-based models, e.g., Long Short-Term Memory (LSTM) [21] or Gated Recurrent Unit (GRU) [22], which are supposed to transform a given cell tower sequence into a trajectory composed of many connected road segments. However, directly applying these models does not work. First, the RNN outputs are conditionally independent, i.e., the RNN model cannot guarantee that two adjacent output road segments are connected. Second, since a cell tower may cover a large area with hundreds of roads, the number of inferred road segments for each cell tower is large and varies. To tackle the above two challenges, we propose an encoder-decoder model for DMM, which maintains two RNN models to maximize the probability of identifying a true trajectory. One RNN model encodes a variable-length cell tower sequence into a context vector with a fixed size. The other RNN model decodes the vector into a variable-length sequence of road segments. We also plug an alignment component into the basic model to cope with long cell tower sequences.

To enable more accurate map matching for cellular data, DMM leverages a number of heuristics to refine the inference. Besides the heuristics considered in previous works [12] (i.e., taking the major roads and staying on the same road), we also adopt a new global heuristic, i.e., people prefer to choose a road trajectory that has less frequency of turns given a sequence of cell towers. To incorporate these three heuristics into a unified map matching framework, we develop a reinforcement learning scheme. It has a well-defined reward function to encourage the map-matched outputs that follow the above three heuristics.

We implement DMM in PyTorch [23]. In order to train DMM, we use an anonymized city-level cellular dataset provided by mobile carriers in a large city. A GPU card is used to accelerate training the neural networks. We evaluate DMM with real-world cell tower sequences generated by volunteers travelling more than 1,700 km. The experiment results demonstrate that DMM provides precision and recall of 80.43% and 85.42%, respectively, corresponding to performance gains of 19.33% and 15.12% over the state-of-the-art approach [12]. DMM also significantly reduces the inference time of HMM-based approaches by 46.58× while maintaining the accuracy.

In summary, this paper makes the following contributions.

- We develop DMM, an RNN-based map matching scheme.
- We customize DMM to tackle a set of challenges, including an encoder-decoder model for variable length of input and output sequences, a spatial-aware representation model for cell towers, and a reinforcement learning scheme to refine the output results.
- We conduct extensive experiments and demonstrate the effectiveness and efficiency of DMM based on a large cellular dataset.

2 MOTIVATION

In this section, we investigate the necessity of a novel map matching scheme for cellular data and the limitations of existing solutions.

2.1 Map matching

We first define some key concepts in map matching.

Definition 1 - Cell tower sample. Every time, a mobile phone communicates with a cell tower, including network service requests (call, SMS and application usage) and the location updates (cell handover and periodic location update), a cell tower sample is passively recorded by the cellular network infrastructure. The cell tower sample includes several fields, i.e., anonymized user identifier, timestamp and the associated cell tower IDs. The anonymized identifier is uniquely associated with each mobile phone. Based on the cell tower map provided by the carriers, we also know the GPS location of each cell tower.

Definition 2 - Cell tower sequence. A cell tower sequence is the input of map matching, composing of a sequence of cell towers accessed by a mobile phone, i.e., $X = x_1, x_2, ..., x_{|X|}$, where $|X|$ is the number of cell towers. In our dataset, we have 887,116 pieces of cell tower sequences from two mobile carriers of a large city.

Definition 3 - Road map. A road map can be described as a directed graph $G(V, E)$, where V is a set of nodes on the road map, representing intersections or terminal points, and E is a set of road segments connecting these nodes. In this study, the road map is obtained from a public open-source website (OpenStreetMap [24]). All road information used in DMM is provided in the OpenStreetMap road map (e.g., the length and speed limit of road segments).

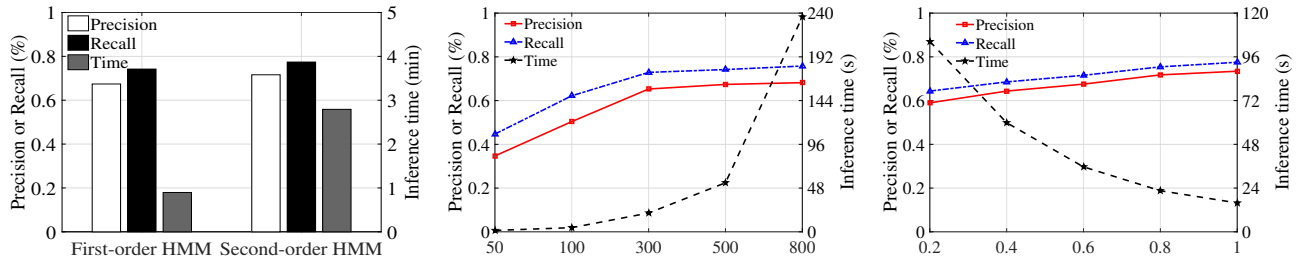
Definition 4 - Candidate road segments. The candidate road segments of a cell tower is a set of roads within a radius R_C near a cell tower. The setting of R_C is related to location error of different location sensors. For the sensor data with low location error (e.g. GPS sensor), we select a smaller value (e.g. 100). In cellular environment, due to the different densities of cell towers in different areas, the choice of R_C varies, e.g., a small value 200 in urban areas and a large value 500 in rural areas.

Definition 5 - Route or trajectory. A route Y is the output of map matching, connecting a sequence of road segments on the road map G, i.e., $Y = y_1, y_2, ..., y_{|Y|}$, where y_i is a road segment in the route Y, $|Y|$ is the number of road segments, and the end point of y_i is the start point of y_{i+1}.

Definition 6 - Map matching. Given a cell tower sequence X and a road map $G(V, E)$, a map matching model finds a most-likely route Y on G.

2.2 Existing map matching solutions

Most recent map matching approaches are based on Hidden Markov Models (HMMs) [12–17]. They define a hidden state (road segment) and an observable state (cell tower) for the map matching process.



(a) Orders of HMM model (b) Search radius R_C (m) (c) Sampling rate (/min)

Figure 1: Performance of the HMM-based model under different settings.

Each road segment maintains two probabilities, i.e., emission probability and transition probability. The emission probability evaluates the probability of a cell tower is localized at this road segment. Transition probability evaluates the probability that transits from the last road segment (first-order HMM) or the last two road segments (second-order HMM) to the current road segment. Empirically, HMMs assume that closer roads have larger emission probabilities. For transition probability, HMMs assume to follow the shortest path between the surrounding roads of two consecutive cell towers.

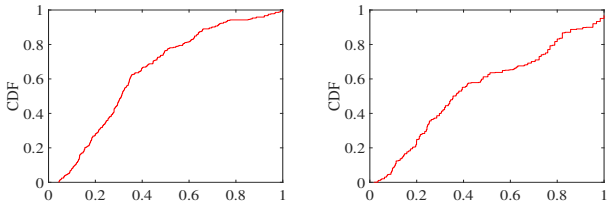
For an online inference, HMM first searches for the candidate road segments within the search radius R_C of each cell tower. As the HMM process proceeds, the product of emission probabilities and transition probabilities of some routes that are composed of a sequence of road segments increases faster than others. In the end, an optimal route with the highest product value can be identified using the dynamic programming technique [25], which leads to $O(n^2)$ computation complexity.

We find three factors that may influence the performance of HMM-based map matching, i.e., the order of HMM model, the location error of cell towers and the sampling rate of cell tower sequences. We use a state-of-the-art HMM-based approach [12] and conduct a series of empirical studies to illustrate why the HMM-based methods are not efficient for cellular sequences. For each experiment, we measure precision, recall and inference time on the same hardware. The specific experiment settings are introduced in Sec. 5.1.

Impact of the order of HMM model. A higher-order HMM model considers last several cell towers in the HMM process. It has an important influence on the accuracy and inference time. Fig. 1(a) depicts the performance of HMM models on different orders (first-order HMM and second-order HMM). We discover that the accuracy of second-order HMM is higher than that of first-order HMM, but the inference time significantly increases.

Impact of location error of cell towers. The location error of cell towers determines the setting of search radius R_C. This leads to different number of candidate road segments in the HMM process. Less road segments indicates fast inference, but it may lead to local optimal results. We investigate the performance of an HMM-based algorithm with respect to different R_C in Fig. 1(b). When R_C is small, the accuracy decreases sharply despite the fast inference time. As R_C increases, the inference time increases at an exponential rate. This is because more candidate road segments are considered into the HMM process, leading to the exponential growth of search space.

We study the location error of the cellular data. We depict the Cumulative Distribution Function (CDF) distribution of the location



(a) Location error (km) (b) Sampling rate (/min)

Figure 2: Properties of cellular data.

error of collected cellular data (Sec. 5.1), Location error is measured as the distance between the user's GPS position and the cell tower position. As shown in Fig. 2(a), about one third of the location errors of cell towers are larger than 0.4 km, corresponding to a large search radius R_C, which implies a long inference time.

Impact of sampling rate of cell tower sequences. The sampling rate determines the distance between two consecutive cell towers, affecting the running time of calculating shortest paths in the HMM process. We depict the inference time of map matching with respect to different sampling rates in Fig. 1(c). As the sampling rate decreases, the inference time increases at an exponential rate.

We exploit the sampling rates of cell tower sequences in Fig. 2(b). Since the cell tower can only receive the signal when a mobile phone requests the location updates or an application requests the network services, nearly all cell tower sequences have average sampling rate less than 1 sample per minute, leading to an infeasible inference time for map matching.

Summary. From the above empirical experiments, we conclude that the three factors impact the performance of HMM-based approaches. It is difficult to determine the appropriate order of HMM model, the search radius of candidate road segments and the sampling rate of cell tower sequence to achieve the best performance on both accuracy and inference time in the cellular environment.

3 DESIGN OF DMM

In this section, we introduce an overview of DMM and the design of key components in DMM.

3.1 DMM Overview

Fig. 3 depicts the architecture of DMM, consisting of two stages, i.e., the offline training and the online inference.

Offline Training. Given the cell tower sequences in the cellular dataset, we first learn a location representer to capture high-quality

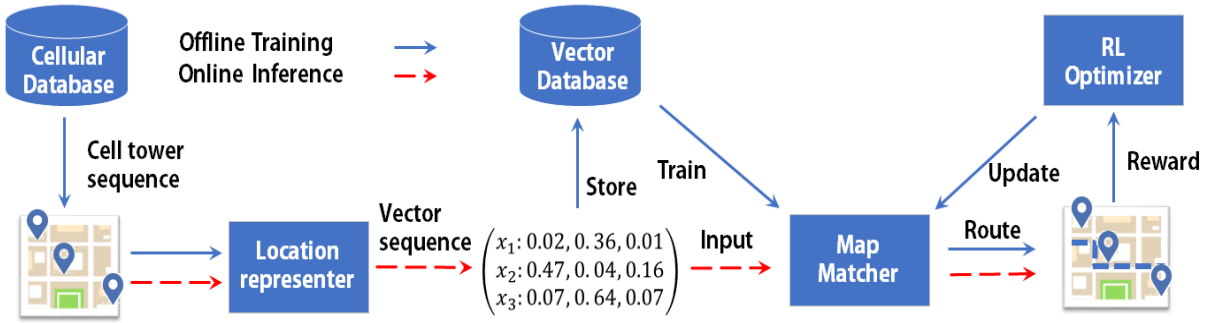


Figure 3: The architecture of DMM.

representations for cell towers (Sec. 3.2). Based on the location representer, we transform all the cell tower sequences into vector sequences and store them in a vector sequence dataset, which will be used for training the map matching model. Then, we learn an RNN-based map matching model to generate the most-likely route on the road map given a vector sequence (Sec. 3.3). The vector sequences as well as the estimated ground truth labels generated from an HMM-based method [12] are used to train the model. Moreover, we customize the map matching model into a reinforcement learning framework to refine the map matching results (Sec. 3.4). By the reward mechanism of reinforcement learning that automates to explore the space of possible results, the initial map matching model is further optimized by incorporating heuristics. Note that the training of the models can be conducted offline, without impacting the speed of online inference.

Online inference. In this stage, cell tower sequences are continuously fed into DMM for route inference. For a cell tower sequence, DMM first transforms it to a vector sequence by the location representer and passes the vector sequence into the final map matching model to identify the most-likely route on the road map.

3.2 Location representer

Intuitively, DMM can quantify an input cell tower using two approaches, i.e., one-hot representation and GPS coordinates of the cell tower. For the one-hot representation, we represent the cell tower as a high dimensional binary vector, in which all bits are '0' except one '1', referring to as the specific cell tower. However, the binary based cell tower representation suffers from two drawbacks. First, the redundant representation reduces the training efficiency of the map matching model, especially in the environment with a large number of cell towers. Second, the learned matching patterns cannot be effectively utilized for unobserved cell tower sequences. For the other representation, it restricts the representation of a cell tower into a two-dimensional GPS coordinates, which essentially encodes the spatial proximity among cell towers. However, it is difficult to derive a high-quality representation of an input cell tower sequence from a sequence of two-dimensional coordinates. Towards this end, we propose to leverage the auto-encoder model [26] to automatically learn high-level and low-dimensional cell tower representations.

The auto-encoder uses a multi-layer neural network to learn the identity mapping for the same input and output. The middle layer learns high-level representations for cell towers, where the number of hidden neurons is less than that of the input and output layers.

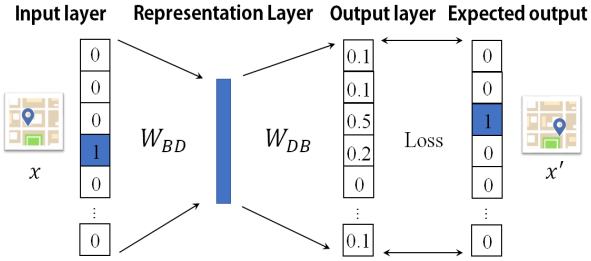


Figure 4: The architecture of the location representer.

However, the basic auto-encoder model is hard to capture the spatial-aware feature among cell towers. Towards this end, we instead use spatially-close cell towers as the expected output of the auto-encoder model. By this way, the spatial characteristic of close cell towers can be easily incorporated into the representations.

Given a cell tower x, we learn the model to maximize the probability that predicts the cell tower x' in the spatially-close cell tower set C_x as Eq. 1. C_x is constructed by the preceding and the following cell towers in a search radius of the present cell tower.

$$\text{maximize} \sum_{x' \in C_x} \log P(x'|x) \tag{1}$$

Fig. 4 depicts the architecture of location representer, consisting of an input layer, a representation layer, and an output layer. The input and output are close cell towers in space and the representation layer plays the role of extracting the high-level features of input cell towers. The input layer simply takes a B-dimension binary cell tower vector as input, where B is the size of cell tower set. We use a fully-connected neural network to transform the input into a D-dimension vector in the representation layer, which can be expressed as a matrix transformation $\boldsymbol{W}_{BD}$. In the output layer, we use a fully-connected neural network as well as a softmax network to classify the D-dimension vector as a spatially-close cell tower x' in C_x. Specifically, the fully-connected neural network (denoted as $\boldsymbol{W}_{DB}$) learns a classification function in the low dimensional vector space and outputs the classification value. The softmax network then normalizes the output values to $[0, 1]$, indicating the probability distribution of all cell towers.

To train the location representer, we feed the spatially-close cell tower pairs into the model continuously and calculate the difference between the output probability and the expected output probability as the optimization criterion. After many iterations, the location

information of cell towers as well as spatial proximity among cell towers are learned and represented in the weight matrix of the representation layer.

3.3 Map matcher

Inspired by recent advancement on recurrent neural networks (RNNs) for sequential-based applications [22, 27–33], we design an RNN-based map matcher to learn the mapping between the cell tower sequence and the sequence of roads on the road map. Fig. 5 depicts the architecture for the map matcher, consisting of an RNN encoder-decoder model (blue blocks) and a plug-in alignment component (red block).

3.3.1 Encoder-decoder. The input to the map matcher is a represented cell tower sequence X. The encoder first transforms the input cell tower sequence X into a sequence of hidden state $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{|X|}$. After encoding the input, the context vector $\boldsymbol{c}$ (the last hidden state $\boldsymbol{h}_{|X|}$) is passed to the decoder. Then, the decoder identifies the optimal road segments successively based on the context vector $\boldsymbol{c}$, and finally generates the route Y.

Encoder. The encoder is implemented as one RNN, which encodes the cell tower sequence X successively and embeds it into a context vector $\boldsymbol{c}$. During the encoding process, the hidden state $\boldsymbol{h}_t$ is updated as Eq. 2.

$$\boldsymbol{h}_t = \text{GRU}\left(\boldsymbol{h}_{t-1}, x_t\right) \tag{2}$$

where GRU (Gated Recurrent Unit [22]) is a non-linear function. After encoding the whole cell tower sequence, the continuous vector $\boldsymbol{c}$ (i.e., the hidden state $\boldsymbol{h}_{|X|}$) is served as the input of the decoder network, which conserves the location information of the sequence.

Decoder. The decoder is the other RNN, which generates the map-matched route Y successively given the context vector $\boldsymbol{c}$. At the beginning, we feed the decoder a Start Of Sequence token (*SOS*) to start a map matching process. At step t, given the last predicted road y_{t-1} and the hidden state $\boldsymbol{h}_t$ at step t, the probability can be estimated as Eq. 3.

$$P(y_t|y_1, \cdots, y_{t-1}) = \text{GRU}\left(y_{t-1}, \boldsymbol{h}_t\right) \tag{3}$$

where GRU is the other non-linear function to generate the probability y_t. Until the decoder generates an End Of Sequence (*EOS*) token, we accomplish a map matching process and finally obtain a map-matched route Y.

3.3.2 Alignment model. In the above encoder-decoder model, the encoder compresses the input cell tower sequence into a fixed context vector, which is difficult to memorize the whole information of long sequences. As a result, the basic map matching model faces a performance degradation on accuracy in the case of the long sequences. Towards this end, we plug an alignment component into the encoder-decoder model, which learns to match and align the input cell tower sequence and the map-matched route jointly. Specifically, the alignment component considers all hidden states $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{|X|}$ of the encoding stage instead of the last context vector $\boldsymbol{c}$, as the basic encoder-decoder model does. This avoids to conserve the whole information of cell tower sequence, and thus handling the long cell tower sequences. Next, we present how the alignment component works.

As shown in the red block of Fig. 5, at step $i-1$ of the decoding process, the decoder generates a road segment y_{i-1} and updates the

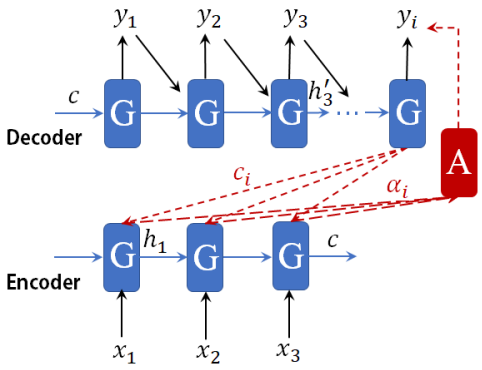


Figure 5: The architecture of map matcher.

hidden state $\boldsymbol{h}_i'$. Then, the alignment component searches for the most relevant context vectors from the hidden states $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{|X|}$ from the encoding process. An adaptive context vector $\boldsymbol{c}_i$ is designed to weight the hidden states $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{|X|}$ to concentrate the relevant parts of the cell tower sequence as Eq. 4.

$$\boldsymbol{c}_i = \sum_{j=0}^{|X|} (\alpha_{ij} \boldsymbol{h}_j) \tag{4}$$

where j represents the jth element in the input cell tower sequence, $|X|$ represents the length of the sequence, and $\boldsymbol{h}_j$ represents j_{th} hidden state of the encoder. α_{ij} measures the importance of $\boldsymbol{h}_j$, which can be calculated by Eq. 5.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{|X|} \exp(e_{ik})} \tag{5}$$

where e_{ij} is a score function, which measures the matching degree between the hidden state $\boldsymbol{h}_j$ of the encoder network and the hidden state $\boldsymbol{h}_{i-1}^{'}$ of the decoder.

3.3.3 Training for the map matcher model. The RNN-based map matcher needs to be trained using a large amount of cell tower sequences with labeled true route, which is difficult to obtain in practical. We take the second best to generate the labels of cell tower sequences using a state-of-the-art HMM-based method [12]. Although this will bring deficiencies to the map matcher model, the model is only for the initialization for the RL optimizer for further optimization.

Specifically, given a cell tower sequence, the encoder-decoder model and the alignment model are jointly trained to maximize the log-likelihood of the output road sequence:

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \log P_{\boldsymbol{\theta}}\left(Y_i|X_i\right) \tag{6}$$

where $\boldsymbol{\theta}$ is the parameters in networks, N is the number of training pairs sampled from the training data and (X_i, Y_i) is a training pair of the cell tower sequence and the output route. To speed the convergence in the training process, we use expected output $\hat{y}_{t-1}$ obtained from the labels as the input of step t, instead of the predicted output y_{t-1} of last step $t-1$.

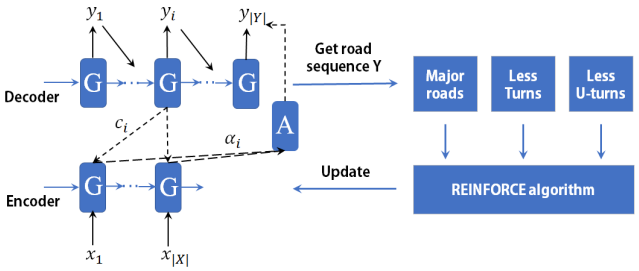


Figure 6: The architecture of RL optimizer.

3.4 RL optimizer

To further improve performance, we exploit the global hints observed from real driving scenarios, such as preferring the routes with more proportion of major roads, less frequency of turns and U-turns. To incorporate these hints, we customize the basic map matching model into a reinforcement learning framework.

3.4.1 Basics for RL. Reinforcement learning (RL) is a promising machine learning approach, which instructs an agent to accomplish a task by trials [34–37]. A learning process is described by four elements, i.e., state, action, policy and reward. Given a specific state, a policy of the agent learns to map from the state to an action. A reward is then designed to estimate how good or bad of current action or a sequence of actions. Finally, the policy is optimized for better performance with respect to the reward.

To apply RL in DMM, we view the map matcher model as the agent and customize it into a RL framework with specific designs of the key elements. Fig. 6 depicts the architecture of RL optimizer. At every iteration, the map matcher agent reads the cell tower sequence $X = x_1, ..., x_{|X|}$ as state input and generates an action sequence $Y = y_1, y_2, ..., y_{|Y|}$, which is also the map-matched result of our model. A reward r, which measures the satisfactions of global hints of route Y, is then computed to assess the quality of output route. Finally, the REINFORCE algorithm [38] is used to update the policy of map matcher agent based on the reward. Next, we introduce the details of reward function and REINFORCE algorithm.

3.4.2 Reward design. We incorporate a number of global hints into DMM. First, users are more likely to select a sequence of major roads based on the uneven distribution of traffic flows. Second, users prefer the routes with turns as few as possible if exists multiple possible routes between the origin and destination. Third, people normally prefer to follow the same direction, rather than completely changing the moving direction. Based on the above observations, we present the corresponding design of the reward $r(Y)$ to evaluate the output routes, as shown in Eq. 7.

$$r(Y) = \lambda_P \cdot r_P + \lambda_T \cdot r_T + \lambda_U \cdot r_U \tag{7}$$

where $\lambda_P, \lambda_T, \lambda_U \in [0, 1]$. $r(Y)$ is a shorthand for $r(X, Y)$ where X is the input cell tower sequence, Y is the map-matched route. r_P, r_T, r_U represent for the goal of the output route, namely, spatial proximity to the input cell tower sequence, less frequency of turns, less U-turns. In the following, we present detail designs for the reward.

Spatial proximity. The reward of spatial proximity r_P needs to ensure that the generated routes are spatially-closest to the input cell tower sequence, which is in line with the intuition of map matching task. However, due to the large location error of the cellular data, a cell tower may cover an area with many roads, leading to the basic intuition incorrect. Inspired by the first observation, we propose to use the negative weighted projection distance between the input cell tower sequence and the map-matched route as the design of r_P. The projection distance is calculated by the geodesic distance between the GPS location of cell tower and its projected location on the corresponding road segment.

Specifically, we assign a road weight w_s to different types of roads. The road with a higher speed limit is assigned with a smaller weight, making the projection distance to the major roads smaller. In this setting, the map matching results are more likely to move on the major roads. We use a linear function to calculate the weight as $w_s = 1 - q \cdot r_l$, where q is a constant and r_l is the speed limit of the road. If the user does move on the side road, using this trick may lead to incorrect matching results. However, the probability is lower than that of driving on the main road in most cases.

Less frequency of turns. To avoid the unnecessary turns of output routes, we design a reward r_T, which rewards the route with similar number of turns between the sequence and the output route. Based on the second observation, we define the reward r_T as Eq. 8.

$$r_T = \begin{cases} 1 - \frac{|T_X - T_Y|}{T_X} & \text{if } T_X \geq T_Y \text{ and } T_X \neq 0 \\ 1 - \frac{|T_X - T_Y|}{T_Y} & \text{if } T_X \leq T_Y \text{ and } T_Y \neq 0 \end{cases} \tag{8}$$

where T_X and T_Y are the estimated numbers of the turns of input and output sequences. We measure the number of turns based on the sum of angles of every adjacent cell towers.

Less U-turns. We design a reward r_U to avoid the occurrence of U-turns in the output routes. Different from the design of r_T, we estimate the difference of the number of U-turns between the cell tower sequence U_X and the output route U_Y as the reward r_U. We measure the number of U-turns by the number of the completely change of the moving direction in the sequence. Specifically, we replace the T_X and T_Y in the reward r_T with U_X and U_Y in Eq. 8 to calculate r_U.

3.4.3 REINFORCE algorithm. In terms of the characteristics of encoder-decoder based policy in the map matcher agent, we adopt the REINFORCE algorithm [38] to refine the policy of map matcher agent. It optimizes the policy in an episodic way, i.e., optimizing the policies using the final reward obtained at the end of an episode, such as playing chess (win/lose in the end). In DMM, the reward of map-matched route cannot be computed until the end of map matching process.

The training process of RL optimizer is outlined in Algo. 1. We first initialize the parameters of policy π_θ with a pre-trained map matcher agent. Given a cell tower sequence X, we generate a route Y

Algorithm 1 Training process of the RL optimizer

1: Initialize the parameters θ of policy π_θ using the pre-trained map matcher;
2: **for** iteration = $1, \cdots, I$ **do**
3: Sample M routes from the distribution $\pi_\theta(\cdot|X)$;
4: Estimate an expected reward $J(\theta)$ as Eq. 9;
5: Calculate the gradient $\nabla J(\theta)$ as Eq. 10;
6: Update the parameters θ of policy π_θ as Eq. 11.
7: **end for**

based on the policy π_θ, consisting of an action sequence (a sequence of road segments). Then, according to the reward $r(Y)$, the expected reward can be obtained as Eq. 9.

$$J(\theta) = E_{Y\sim\pi_\theta(\cdot|X)}\left[r\left(Y\right)\right] \tag{9}$$

There may be infinite map-matched routes for a cell tower sequence X. As a result, the expectation of reward $E_{Y\sim\pi_\theta(\cdot|X)}$ from the distribution $\pi_\theta(\cdot|X)$ cannot be estimated directly. We approximate this expectation by sampling M routes from the distribution $\pi_\theta(\cdot|X)$ [34]. To reduce the variance that leads to inaccurate estimation of expected reward, we subtract the reward $r(Y)$ from a baseline b [39]. b is defined as an average reward of sampled M routes. Then, the gradient can be approximated as Eq. 10.

$$\nabla J(\theta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{|Y|}\nabla \log \pi\left(y_i|y_{1:i-1}, X\right)\left[r\left(Y\right) - b\right] \tag{10}$$

Finally, we update the parameters of map matcher agent using gradient descent as Eq. 11. η is the learning rate.

$$\theta \leftarrow \theta + \eta\nabla J(\theta) \tag{11}$$

4 IMPLEMENTATION

In this section, we introduce implementation details on three models and online inference process of DMM in Fig. 7. We implement DMM on a server with 2 CPUs. Both CPUs have dual Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70 GHz with 8 cores. A graphics processing unit card (NVIDIA Titan X) is used to accelerate the training process. We develop DMM in Python. The code is implemented in PyTorch [23], an open-source machine learning framework.

4.1 Offline stage

With the cellular dataset provided by mobile carriers, we conduct offline training of DMM. Three models in DMM are trained. We first train the location representer to obtain the high-quality cell tower representations (Sec. 3.2), and then perform the map matcher model training (Sec. 3.3). Finally, we train the reinforcement learning model to refine map matching results (Sec. 3.4).

Training for the location representer. To train the location representer, we first construct a spatially-close cell tower pair set from cell tower sequences. For any one cell tower in the cell tower sequence, we choose the cell towers within a certain window before and after the cell tower. The window size is set to 2. We pair each cell tower in the window and the existing cell tower together to form a cell tower pair. After traversing all the cell tower sequences, we obtain a spatially-close cell tower pair set.

We implement the location representer as a two-layer neural network. The size of input and output layer is set to the size of cell tower set B. The size of hidden unit is 64. Cross entropy loss [40] is used to calculate the loss between true output and expected output. Once trained, we store the learned representations of cell towers into a hash table. This can speed up the representations of cell towers in the following map matching process.

Training for the map matcher. We train the map matcher model using the represented cell tower sequences as well as the estimated ground truth labels generated from an HMM-based method [12]. The parameters of map matcher are uniformly initialized to $[-0.1, 0.1]$.

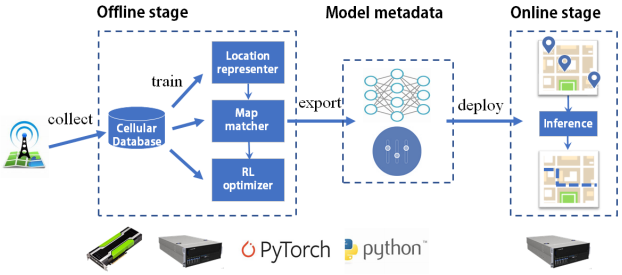


Figure 7: The workflow of DMM.

We use Adam optimizer [41] to update the parameters. Batch size is set as 128. We use Gated Recurrent Unit (GRU) [22] as the RNN units of encoder and decoder networks due to its higher computational efficiency than LSTM [21]. The dimension of hidden state is set as 128. The learning rate is set as 0.001. The GRUs are regularized with a dropout rate of 0.1. We implement the alignment component as a feed-forward neural network, which is jointly trained with the encoder-decoder networks.

During the training of map matcher, mini-batch is a classic technique to accelerate the training speed and model convergence. Cell tower sequences are randomly selected to update the parameters at every iteration. We adopt the padding technique [42] to fill short cell tower sequences with the same length of the longest sequence in a batch. This ensures that the cell tower sequences in a batch are of the same length. We also divide the training cell tower sequences into different buckets according to the number of sampling points [42]. During training, the mini-batches are sampled from the same bucket. This can avoid the training inefficiency caused by padding too many meaningless *PADs* in short cell tower sequences.

Training for the RL optimizer. Since the training of RL optimizer does not need true label to calculate the loss, we use the cell tower sequences as the training data to train the RL optimizer. We use stochastic gradient descent with the learning rate $\eta = 0.01$. We set $\lambda_P = 0.5$, $\lambda_T = 0.25$, $\lambda_U = 0.25$.

4.2 Online stage

Once the DMM models are trained, we export the metadata of DMM for online deployment. The metadata includes the network architecture and the refined DNN parameters, which are used to deploy DMM for online inference. After deploying, DMM takes the cell tower sequences as input, transforms them into vector sequences, and identifies the most-likely routes on the road map.

5 EVALUATION

We first present the experiment results on the overall performance of DMM. Then, we study the performance of the two key models in DMM, i.e., the location representer and the RL optimizer.

5.1 Experiment settings

Data collection. We recruited volunteers and collected their GPS locations as ground truth. All the volunteers gave their consents to participate in the experiments and use their data for study. During the data collection, we asked the volunteers to equip with mobile phones and drive in our city. The volunteers were required to enable GPS on their mobile phones. We also install a data collection application (GPS Toolbox [43]) to record GPS locations at a high sampling rate

Figure 8: Coverage map of our collected dataset.

up to 1 sample per second. The mobile carrier also provides the corresponding anonymous cell tower sequences of the volunteers for evaluation. We map-match all GPS-based location sequences to obtain the true routes as the ground truth [14].

Statistics of the collected data. We collected 198 car driving traces, 167 of which are in urban areas (average distance less than 9 km to the city center). The total length of dataset is 1,701 km with 2,848 distinct cell towers. The traces cover various road types, such as main roads and side roads, varying from 2.5 km to 23.6 km. The red lines in Fig. 8 show a coverage map of the collected dataset. Since most of the traces are collected in urban areas with varieties of traffic conditions, 76% of average moving speed is below 18 km/h. About 99% sampling rates of the traces are less than 1 sample per minute.

Performance criteria. We assess the accuracy of all map matching approaches by comparing the map-matched route to the ground truth route. Given the testing cell tower sequences, we use average precision and recall as accuracy criteria. Precision is defined as the ratio of the total length of the correctly-matched route to the total length of the route. Recall is the ratio of the total length of the correctly-matched route to the total length of the ground truth route. Meanwhile, average inference time is used to evaluate the efficiency, which is defined as the average running time required to transform cell tower sequences into routes.

Benchmarks. We compare DMM with following baselines. All the baselines are implemented in Java. By default, we set the search radius $R_C = 500$ in our experiments.

- *ST-Matching.* ST-Matching [14] is a widely used HMM-based approach for mapping low-sampling-rate GPS-based cell tower sequences, which takes the spatial topological structure of road maps and the temporal constraints of moving speed into account simultaneously.
- *SnapNet.* SnapNet [12] designs an HMM-based map matching approach for cellular data collected from mobile phone side. It incorporates several digital map hints and heuristics to handle the issues of larger location error and low sampling rate, e.g., preferring major roads and staying on the same road.
- *SnapNet w/o I.* SnapNet [12] adopts a linear interpolation technique to improve the sampling rates of cell tower sequences, but it severely harms the accuracy of map matching, as we have discussed in Sec. 2.2. Towards this end, we implement a variant of SnapNet, denoted as SnapNet w/o I, to compare with other methods. In particular, SnapNet w/o I gets rid of the linear interpolation from pre-processing model of SnapNet.

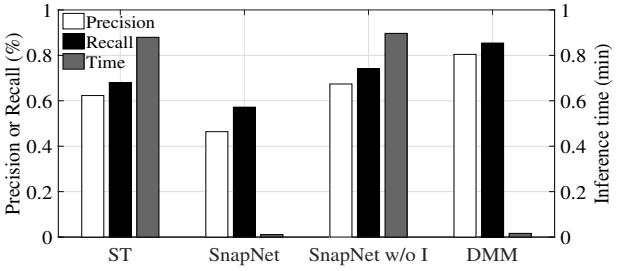


Figure 9: Overall performance of DMM.

Table 1: Inference time (s) of different approaches w.r.t. the sampling rate of cell tower sequences (/min).

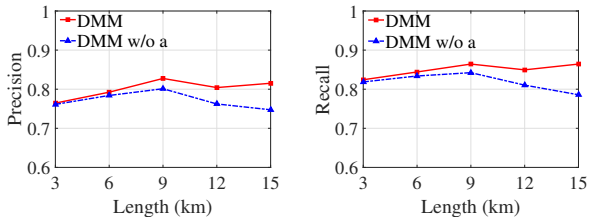
Sampling rate	0.2	0.4	0.6	0.8	1
ST-Matching	111.65	64.58	39.91	26.37	21.35
SnapNet w/o I	104.46	59.84	35.63	22.55	15.84
SnapNet	0.10	0.10	0.15	0.14	0.13
DMM	0.94	0.77	0.84	1.25	1.08

5.2 Overall performance of DMM

We first compare DMM with the baselines on the cell tower sequences collected by our volunteers. Fig. 9 depicts the overall performance of all approaches.

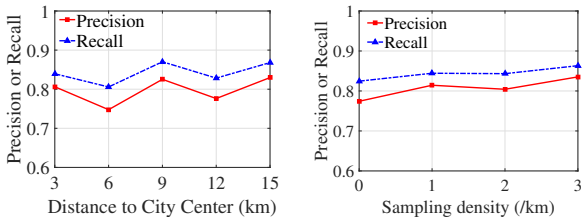
5.2.1 Accuracy. We use the map matcher in DMM to transform the cell tower sequences of our volunteers into the routes on the road map and compare the generated results with the corresponding GPS ground truth. All the 1701-km traces are used in the test. As depicted in Fig. 9, we discover that DMM provides the best accuracy. The reasons are as follows. First, DMM adopts an RNN-based model to transform the cell tower sequence into context vectors, which conserves the historical location information for map matching. For HMM-based approaches, they can only take the last road segment into account to make inference, leading to the loss of historical cell tower information. Second, the location representer enables high quality cell tower representations, which allows to make inference for unobserved cell tower sequences. Third, we also leverage a reinforcement learning based framework to incorporate the global information of the cell tower sequences. We will further decompose the performance of location representer and RL optimizer in Sec. 5.4 and Sec. 5.5 respectively.

5.2.2 Running efficiency. We also use the collected dataset to evaluate the running efficiency of different map matching approaches in Fig. 9. DMM runs much faster than the other HMM-based approaches, except SnapNet. This is because DMM only needs to make a forward computation of neural networks to identify an optimal route during the inference stage, which only requires $O(n)$ computation complexity. In contrast, the HMM-based approaches rely on heavy computations of dynamic programming to identify the optimal matching, with a time complexity of $O(n^2)$ computation complexity. Although SnapNet has less inference time than DMM by using a linear interpolation of raw cell tower sequences, the precision and recall of SnapNet decrease sharply. SnapNet is more capable of handling the trajectories on highways. In urban areas, the linear



(a) Precision. (b) Recall.

Figure 10: Alignment component in the map matcher.



(a) Area. (b) Sampling density.

Figure 11: Different input cell tower sequences.

interpolation of low-sampling-rate cell tower sequences introduces large noise between two adjacent cell towers.

To exploit the inference time of different approaches as the sampling rate varies, we discretize the sampling rate into five levels, i.e., $\{< 0.2/min\}$, $\{\geq 0.2/min \ \& \ < 0.4/min\}$, $\{\geq 0.4/min \ \& \ < 0.6/min\}$, $\{\geq 0.6/min \ \& \ < 0.8/min\}$ and $\{\geq 0.8/min \ \& \ < 1/min\}$ and obtain results in Tab. 1. When the sampling rate is low, DMM can still maintain lightweight inference. In contrast, the inference time of HMM model increases exponentially to maintain high accuracy.

5.2.3 Effect of the alignment component. To enable more accurate map matching for long cell tower sequences, we plug an alignment component into the basic map matching model. We explore the benefit of the alignment component under different length of cell tower sequences, varying from 3 km to 15 km in Fig. 10. We discover that both the precision and recall of the basic encoder-decoder model deteriorate rapidly as the length of cell tower sequences increases. By incorporating the alignment component, the results are better than the basic model, especially for the long input sequences. This is due to the fact that the alignment component only needs to memorize relevant location information in the cell tower sequence, instead of memorizing the whole cell tower sequence.

5.3 DMM Robustness

We evaluate system robustness according to different attributes of input cell tower sequences.

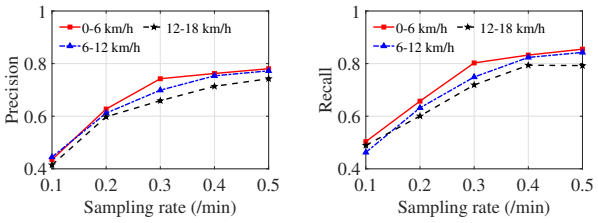
5.3.1 Different input cell tower sequences. We first exploit the system robustness in Fig. 11 according to different categories, i.e., area of the cell tower sequences, sampling density of the cell tower sequences. The sampling density is defined as the average number of sampling points per kilometer, which is determined by the moving speed and sampling rate of the cell tower sequence.

Impact of area of the cell tower sequences. We evaluate the impact of the area of cell tower sequences on system performance. We divide the collected sequences into 5 levels according to the distance to the center of city. Fig. 11(a) depicts the accuracy in the different areas. As shown, DMM achieves comparable accuracy in both urban areas and remote areas. The reasons are as follows. First, in the remote area (larger than 9 km), the driving speed is high and the cell tower density is low. Both will cause low sampling rates that may impact the system performance; however, road density in remote areas is much lower than that that in urban areas, which makes the map matching model easily determine the true route, which a user is moving along with. Second, in the urban area, although road conditions are more complex, cell tower density is higher too; therefore, the sequences in urban areas have high sampling density, and thus more information can be used for map matching.

Impact of sampling density of the cell tower sequences. We also exploit the accuracy as sampling density varies. We discretize the sampling density into four levels, i.e., $\{\geq 0/km \ \& \ < 1/km\}$, $\{\geq 1/km \ \& \ < 2/km\}$, $\{\geq 2/km \ \& \ < 3/km\}$ and $\{\geq 3/km\}$. As shown in Fig. 11(b), the results reveal that DMM also achieves stable accuracy as the sampling density varies. For the cell tower sequences with high sampling density, more information can be used for map matching. For the cell tower sequences with low sampling density, we deeply analyze the map matching results and find that most cell tower sequences with low sampling density are collected in the remote area, which has better map matching performance.

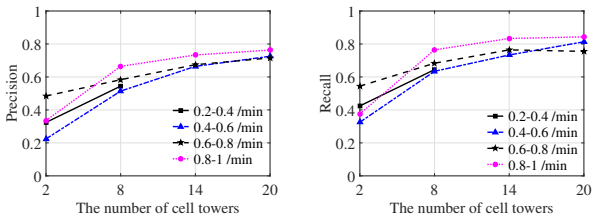
5.3.2 Impact of sampling rate and moving speed. The above experiment results are the average results for our collected cell tower sequences. We further explore the accuracy in the urban area, which is more challenging because of high road density and complex road condition. We calculate the average distance of each cell tower in a cell tower sequence to the city center and conserve the cell tower sequences that the distance to city center is less than $9km$ for evaluation. To test the system robustness on cell tower sequences with lower sampling density, we down-sample each cell tower sequences for lower sampling rate. In this way, with consistent moving speed, the average distance between the cell towers will be increased as the sampling rate decreases. Towards this end, we first describe the procedure of processing the collected dataset into smaller datasets with different levels of sampling rates and moving speeds and then test the performance on each dataset.

We first split the collected dataset into the datasets with different levels of moving speeds. Based on the statistical analysis on our collected dataset, we discretize the moving speeds into the three levels, i.e. $\{\geq 0km/h \ \& \ < 6km/h\}$, $\{\geq 6km/h \ \& \ < 12km/h\}$, $\{\geq 12km/h \ \& \ < 18km/h\}$ and obtain three datasets. Then, we further divide each of the three datasets into five sub-datasets according to the preseted levels of sampling rates (i.e., $0.1/min$, $0.2/min$, $0.3/min$, $0.4/min$, $0.5/min$). Specifically, for a sub-dataset with the same level of moving speed, we first sort the sequences according to the ascending order in their sampling rates. Then, we down-sample each trace to a certain sampling rate one by one until all the sequences have been processed. For example, given a trace with 10 cell tower samples in 10 minutes (corresponding to the sampling rate at 1), if the sampling rate is larger than the current level of sampling rate (e.g. $0.5/min$), we remove 5 cell tower samples ($10 - 0.5/min \times 10min$) to obtain the specific sampling rate. If the number of the sequences of a given level of sampling rate reaches 1/5 of the number of



(a) Precision. (b) Recall.

Figure 12: Impact of sampling rate and moving speed.



(a) Precision. (b) Recall.

Figure 13: Impact of sampling rate and number of cell towers.

the sequences in the sub-dataset, the following sequences will be distributed to the next level.

Based on the processed 15 sub-datasets, we exploit the DMM robustness on different levels of moving speeds and sampling rates. As shown in Fig. 12, we find that DMM provides relatively low accuracy under the circumstances of low sampling rate. For example, for the cell tower sequences with the average moving speed about 15 km/h and sampling rate about 0.1 sample per minute (the average sampling distance is about 2.5 km), DMM achieves the average precision and recall about 41.5% and 48.9%. This is because it is difficult for the map matching model to determine the specific route between the sparse cell towers.

With the increase of the sampling rate or the decrease of the moving speed, DMM provides better precision and recall. This is because slower moving speed and larger sampling rate lead to denser cell tower sequences, thus more location information can be used to localize the true route. For example, as the sampling rate increases from 0.1 to 0.5, both the precision and recall values increase sharply (e.g., 78.0% in precision and 85.5% in recall for the sequences with the moving speed below 0.6 km/h). It also suggests the potential of DMM to be better in the future, where mobile app usages will significantly increase and thus the sampling rate of cell tower sequences will be further increased.

5.3.3 Impact of sampling rate and number of cell towers. We also conduct the system robustness evaluation on different level of sampling rate with different number of cell towers in the urban area. Given a level of sampling rate, different numbers of cell towers correspond to different time duration of the route. Specifically, we first partition the cell tower sequences in the urban area into four datasets with different levels of sampling rate. For each sequence in the four datasets, we generate a set of sequences with different number of cell towers by connecting the sequence between the first cell tower and the remaining cell towers. We keep the cell tower sequences with four levels of numbers of cell towers, i.e., 2, 8, 14, 20. For example, for a cell tower sequence $X = x_1, x_2, x_3, \ldots, x_9$, we could generate two sequences, i.e., $X_1 = x_1, x_2$ and $X_2 = x_1, x_2, \ldots, x_8$.

As shown in Fig. 13, we find that the performance of short sequences performs worse than that of long sequences. This indicates that it is hard for our map matching model to work for the short sequences. For example, the accuracy of the cell tower sequence of two cell towers achieves 22.6% in precision and 32.5% in recall. The reasons for better performance of long sequences are as follows. First, DMM adopts an RNN-based model to transform the input into context vectors, which conserves the location information for map matching. Second, our performance criteria focus on the length of correctly-matched route. For the long sequences, it is more tolerant of partial matching errors than short sequences.

Moreover, with the increase of cell tower number and sampling rate, DMM provides better accuracy. For example, when the number of cell towers in a cell tower sequence is larger than 8 and the sampling rate is larger than 0.6/min (corresponding to average moving time of the trajectory is about 13.33 min and average moving length is about 2km with an average speed about 9 km/h), DMM can achieve 58.3% in precision and 68.4% in recall. This is because longer sequences contain more location information that can be used for map matching.

5.4 Location representer in DMM

We verify the effectiveness of our spatial-aware cell tower representation technique in DMM based on the map matching accuracy. We also visualize learned representations of cell towers to better understand our location representer.

5.4.1 Effectiveness of the location representer. We implement a variant of DMM (DMM w/o LR), which simply uses binary vectors to represent cell towers. As depicted in Fig. 14, the precision and recall of DMM w/o LR are 74.66% and 79.54%, worse than those of DMM. This is because DMM w/o LR cannot learn the spatial proximity relationship so that it is impossible to generalize the learned map matching patterns to unobserved cell tower sequences.

5.4.2 Case study of the location representer. We use a case study to present how the location representer captures spatial proximity among cell towers. We visualize the learned representations of 4 cell towers in the cellular dataset. For each cell tower, we find the closest 10 cell towers and lookup their vectors represented by the location representer. Finally, we use Principal Component Analysis (PCA) technique [44] (one of the widely-used data dimension reduction method) to visualize the cell towers in a two-dimensional space. For close cell towers, we use the same sign and color. Fig. 15 depicts that the cell towers with the same marker are close to each other, indicating that the location representer enables close cell towers to have similar representations. This confirms the spatial-aware characteristic of learned representation.

5.5 RL optimizer in DMM

We investigate the performance of the RL optimizer and also use examples to show how it helps for capturing the global hints.

5.5.1 Effectiveness of the RL optimizer. We first study the performance gain of RL optimizer on the accuracy of the map matcher. We report results in Fig. 16. We observe that the RL optimizer

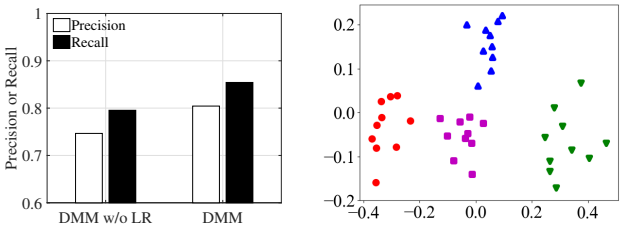


Figure 14: Effectiveness of the location representer.

Figure 15: Spatial proximity of the cell towers.

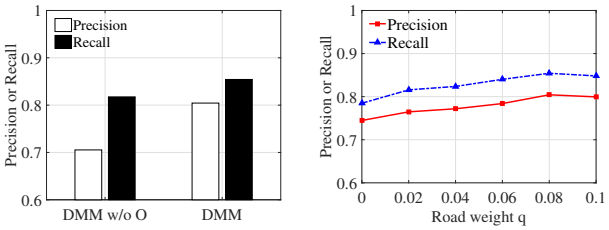


Figure 16: Effectiveness of the RL optimizer.

Figure 17: Effect of the road weight q.

significantly improves the accuracy of basic map matching model in precision and recall by 14.04% and 4.49%, respectively. This indicates that our reinforcement learning based scheme succeeds in optimizing the map matching model with global hints we observed in the real driving scenarios, such as preferring the routes with major roads and less turns.

5.5.2 Effect of the road weight. The road weight q determines the degree of tendency for the main roads. A small q means that the map matching model is more inclined to choose a route with more proportion of side roads, while a large q corresponds to more main roads. We exploit the map matcher performance as the road weight q varies in Fig. 17. We discover that DMM achieves the best performance at $q = 0.08$. As the road weight q increases, both the precision and recall increase, because it is more likely that the map-matched results prefer to choose the routes with more proportion of main roads, which is in line with the observation that the main roads are more likely to be chosen.

5.5.3 Case study of the RL optimizer. The reward $r(Y)$ of a map matching result Y is the weighted sum of the three components aimed at capturing the global hints of the output route, i.e., spatial proximity to the cell tower sequence, less frequency of turns and U-turns. Fig. 18 illustrates by examples to show how three components in the reward help in the map matching results. The top row shows the cell tower sequences (blue points) and the ground truth (blue lines) collected from the volunteers. The bottom row depicts the map matching results of the basic map matching model and DMM, denoted by dashed black lines and red lines, respectively.

Spatial proximity. We first exploit the effectiveness of the spatial proximity hint, which rewards the routes spatially-close to the input cell tower sequence. As depicted in Fig. 18(a), the encoder-decoder model identifies the most path in the output result except a side road, which is closer to the cell tower sequence. After incorporating the spatial proximity hint, DMM takes the route with a sequence of major roads, and thus obtains a better result.

Less frequency of turns. Due to the sparsity of cell tower sequence, there may be multiple routes among cell tower samples. According to the observation that users prefer to choose the route with less frequency of turns [45], we incorporate the hint by a specific design of reward r_T. From Fig. 18(b), we find that DMM can effectively select the route with less turns among multiple possible routes. However, the encoder-decoder model selects the shortest path between two consecutive cell towers. This is because the basic model does not consider the route choice preference of the trip.

Less U-turns. Due to the large location error of cellular data, the encoder-decoder model identifies the most path accurately except unexpected U-turns. We use the reward r_U to eliminate this phenomenon. In Fig. 18(c), we discover that DMM succeeds in avoiding a U-turn. If the cell tower samples actually indicate a real occurrence of U-turn in the raw cell tower sequence, DMM can generate a correct result with U-turns adaptively.

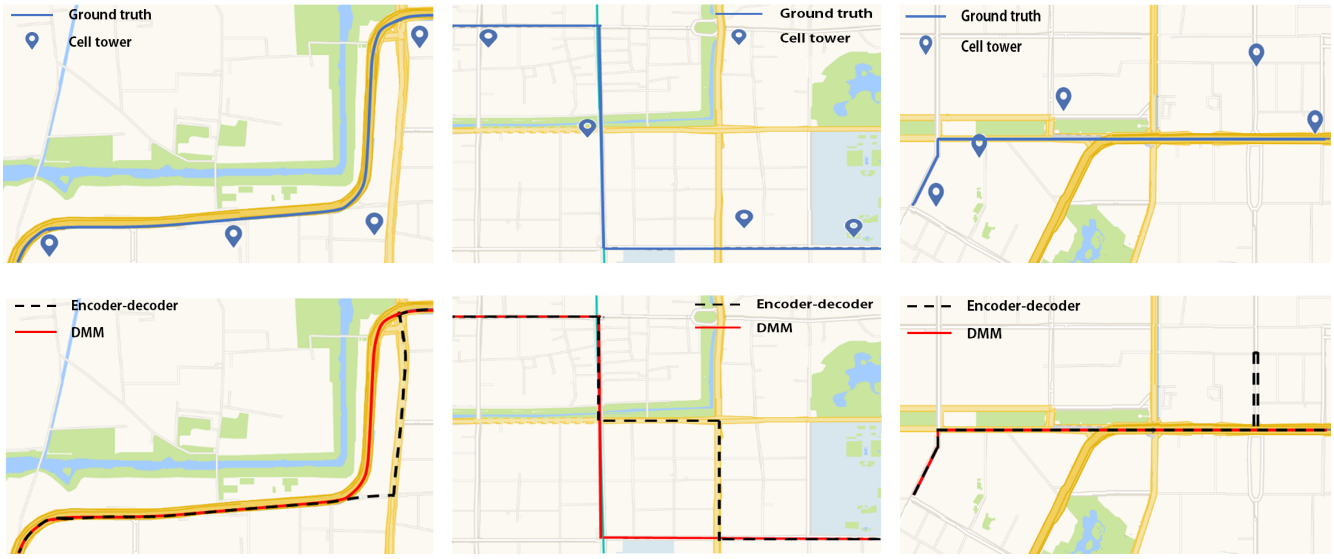
6 RELATED WORK

Many works [12, 19, 46–49] have explored map matching using the cellular data. Algizawy *et al.* [46] extend the typical HMM to mapping cellular-based trajectories for traffic analysis. CTrack [19] proposes a grid-based HMM approach to identify the most likely roads. SnapNet [12] develops an HMM-based model for map matching in view of the road information. However, these approaches cannot consider high-order historical cell tower information. Several data augmentation techniques [47–49] are proposed for the cellular-based map matching model to handle insufficient training data. In our work, we train an RNN-based model using the labels generated by the HMM-based method [12] and optimize the basic model in the reinforcement learning framework.

Meanwhile, several works [16, 50–56] have been proposed to localize the cellular measurement record (MR) data collected by network infrastructures. The types of MR data include sector information, signal latency, signal strength, signal quality, etc. *Cell** [50] and CTS [16] estimates more precise location using sector information. DeepLoc [51] localizes the accurate position using ubiquitous cellular signals received from adjacent cell towers. Ergen *et al.* [53] develop an HMM-based localization model based on the received signal strength indicator (RSSI) sent by adjacent cell towers. RecuLSTM [54] develops a deep learning based framework to infer the positions from measurement records. However, these data are not available in our dataset.

Besides cellular-based data, many previous map matching approaches are designed for GPS data [13–15, 57–60]. Mosig *et al.* [58] apply *Fréchet* distance for map matching, but they cannot consider road network information. Many advanced algorithms, such as conditional random field [59], particle filter [60] and hidden Markov model [13–15], are developed to deal with complex road networks. For example, ST-Matching [14] map-matches GPS trajectories with spatial and temporal information. However, these works cannot be used in DMM because of the large location error and the low sampling rate of cellular data.

Map matching can also be used as a fundamental step for many trajectory mining applications [4, 18, 61–64]. VTrack [18] leverages an HMM-based map matching scheme to estimate road traffic. TS-Join [62] proposes a network-based trajectory similarity join by mapping massive trajectories on the road. Prokhorchuk *et al.* [4]



(a) Spatial proxmity. (b) Less frequency of turns. (c) Less U-turns.

Figure 18: Case study of the RL optimizer, showing the raw cell tower sequences, the ground truth (top), and the map-matched routes of basic encoder-decoder model and DMM (bottom).

infer travel time distributions by map-matched floating car data. TrajCompressor [63] designs a trajectory compression framework, along with the first pre-processing step of map matching.

7 DISCUSSION

DMM heuristics. DMM incorporates several heuristics to achieve the goal of accurate map matching. In the following, we show the validity and rationality of these heuristics. First, we assume that people normally prefer to choose the route that has more proportion of major roads. The assumption is confirmed by [65]. In that work, Yao *et al.* used Multinomial Logit Model to analyze the route choice behaviors of taxi drivers using the GPS data of taxis in China. The result shows that users tend to choose the route with the larger proportion of major roads. Second, we assume that people normally prefer the routes with less frequency of turns between origin and destination. Venigalla *et al.* [45] used a real-world GPS data in urban areas to exploit the effect on route choices and revealed that drivers would rather spend more time or travel longer distance on roads than make frequent turns. Third, we also assume that people normally prefer to follow the same direction, rather than completely changing the moving direction. This is confirmed by the work [66]. Mondal *et al.* analyzed the vehicles at six areas and showed that 93.4% of drivers prefer straight roads.

Deployment cost. For online inference, a CPU with 2 cores is enough. For offline training, we need to process about 0.6 million anonymized cell tower sequences to train DMM. The training data can be acquired in cooperation with the mobile carriers. The amount of data is about the number of cell tower sequences that can be collected from all subscribers of the mobile carriers of a metropolis in one day. We use a graphics processing unit (GPU) to accelerate the training process. Besides, a reliable storage system is used to store the cellular data.

Privacy issues. We use the cellular dataset provided by mobile carriers to train the models in DMM. The data have been anonymized to protect users' privacy by replacing users' identifiers by hash codes. The data only contain anonymized samples of cell towers, without any information related to text messages or mobile phone usages. Moreover, we randomly select a portion of cell tower sequences, which can further prevent leaking privacy.

We collected GPS locations and cellular data from volunteers for evaluation. We anonymized users' identifiers in our data. We explained the experiment design to the volunteers and obtained their consents to use the data for this study.

Limitations. DMM has several limitations. First, to ensure both high precision and recall, higher sampling rate of cell tower sequences (larger than 0.2/min in the urban area) is required for our system (Fig. 12). It will be better to extend our system, where cell tower density and mobile app usages will be further increased in the future. Second, our system targets the driving scenario that has long moving distance and moving time. The scenario of short-distance or short-time movement (e.g. walking) remains to be explored in the future. Third, DMM leverages the estimated labels generated from an HMM algorithm to train its map matching model. It may learn some inaccurate map matching patterns of the HMM algorithm. More labeling methods for training data are worthy to be explored in the future.

8 CONCLUSION

In this paper, we develop an RNN-based map matching framework for the coarse-grained and low-sampling-rate cellular-based location sequences. By combining an encoder-decoder based map matching model, a location representation model, and a reinforcement learning based optimizer together, DMM provides effective and efficient map matching for cellular data. Extensive experiments on a large dataset and real-world collected cell tower sequences in a large city show that DMM can achieve high map matching precision and recall of 80.43% and 85.42%. In addition, DMM also achieves an average speedup about 46.58× faster than the HMM-based methods.

REFERENCES

[1] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. SurroundSense: mobile phone localization via ambience fingerprinting. In *ACM MobiCom*, 2009.

[2] Vincent D Blondel, Adeline Decuyper, and Gautier Krings. A survey of results on mobile phone datasets analysis. *EPJ Data Science*, 4(1):10, 2015.

[3] Zhenni Feng and Yanmin Zhu. A survey on trajectory data mining: Techniques and applications. *IEEE Access*, 4:2056–2067, 2016.

[4] Anatolii Prokhorchuk, Justin Dauwels, and Patrick Jaillet. Estimating travel time distributions by Bayesian network inference. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2019.

[5] Francesco Calabrese, Massimo Colonna, Piero Lovisolo, Dario Parata, and Carlo Ratti. Real-time urban monitoring using cell phones: A case study in Rome. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):141–151, 2010.

[6] Ran He, Jin Cao, Lisa Zhang, and Denny Lee. Statistical enrichment models for activity inference from imprecise location data. In *IEEE INFOCOM*, 2019.

[7] Etienne Thuillier, Laurent Moalic, Sid Lamrous, and Alexandre Caminada. Clustering weekly patterns of human mobility through mobile phone data. *IEEE Transactions on Mobile Computing*, 17(4):817–830, 2018.

[8] Richard Becker, Karrie Hanson, Sibren Isaacman, Meng Loh Ji, Margaret Martonosi, James Rowland, Simon Urbanek, Alexander Varshavsky, and Chris Volinsky. Human mobility characterization from cellular network data. *Communications of the ACM*, 56(1):74–82, 2013.

[9] Desheng Zhang, Jun Huang, Ye Li, Fan Zhang, Chengzhong Xu, and Tian He. Exploring human mobility with multi-source data at extremely large metropolitan scales. In *ACM MobiCom*, 2014.

[10] Zhidan Liu, Zhenjiang Li, Kaishun Wu, and Mo Li. Urban traffic prediction from mobility data using deep learning. *IEEE Network*, 32(4):40–46, 2018.

[11] Zhidan Liu, Zengyang Gong, Jiangzhou Li, and Kaishun Wu. Mobility-aware dynamic taxi ridesharing. In *IEEE ICDE*, 2020.

[12] Reham Mohamed, Heba Aly, and Moustafa Youssef. Accurate real-time map matching for challenging environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):847–857, 2017.

[13] Gang Hu, Jie Shao, Fenglin Liu, Yuan Wang, and Heng Tao Shen. If-Matching: Towards accurate map-matching with information fusion. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):114–127, 2016.

[14] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *ACM SIGSPATIAL GIS*, 2009.

[15] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL GIS*, 2009.

[16] Xingyu Huang, Yong Li, Yue Wang, Xinlei Chen, Yu Xiao, and Lin Zhang. CTS: A cellular-based trajectory tracking system with GPS-level accuracy. In *ACM IMWUT*, 2018.

[17] George R Jagadeesh and Thambipillai Srikanthan. Online map-matching of noisy and sparse location data with hidden markov and route choice models. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2423–2434, 2017.

[18] Arvind Thiagarajan, Lenin Ravindranath, Katrina Lacurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *ACM SenSys*, 2009.

[19] Arvind Thiagarajan, Lenin Ravindranath, Hari Balakrishnan, Samuel Madden, and Lewis Girod. Accurate, low-energy trajectory mapping for mobile devices. In *USENIX NSDI*, 2011.

[20] Mudhakar Srivatsa, Raghu Ganti, Jingjing Wang, and Vinay Kolar. Map matching:facts and myths. In *ACM SIGSPATIAL GIS*, 2013.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[23] PyTorch. https://pytorch.org/.

[24] OpenStreepMap. www.openstreetmap.org/.

[25] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[26] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[27] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. DeepMove: Predicting human mobility with attentional recurrent networks. In *WWW*, 2018.

[28] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li. FS-Net: A flow sequence network for encrypted traffic classification. In *IEEE INFOCOM*, 2019.

[29] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *ACM SIGKDD*, 2016.

[30] Yang Liu, Zhenjiang Li, Zhidan Liu, and Kaishun Wu. Real-time arm skeleton tracking and gesture inference tolerant to missing wearable sensors. In *ACM MobiSys*, 2019.

[31] Wenguang Mao, Mei Wang, Wei Sun, Lili Qiu, Swadhin Pradhan, and Yi-Chao Chen. RNN-based room scale hand motion tracking. In *ACM MobiCom*, 2019.

[32] Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. In *EMNLP*, 2017.

[33] Dario Bega, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. DeepCog: Cognitive network management in sliced 5G networks with deep learning. In *IEEE INFOCOM*, 2019.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[35] Xianzhong Ding, Wan Du, and Alberto Cerpa. OCTOPUS: Deep reinforcement learning for holistic smart building control. In *ACM BuildSys*, 2019.

[36] Zhihao Shen, Kang Yang, Wan Du, Xi Zhao, and Jianhua Zou. DeepAPP: A deep reinforcement learning framework for mobile application usage prediction. In *ACM SenSys*, 2019.

[37] Yu Wei, Minjia Mao, Xi Zhao, Jianhua Zou, and Ping An. City metro network expansion with reinforcement learning. In *ACM SIGKDD*, 2020.

[38] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[39] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[40] Pieter Tjerk De Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

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

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

[43] GPS Toolbox. https://play.google.com/store/apps/details?id=net.gotele.gpsbox&hl=zh.

[44] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[45] Mohan Venigalla, Xi Zhou, and Shanjiang Zhu. Psychology of route choice in familiar networks: Minimizing turns and embracing signals. *Journal of Urban Planning and Development*, 143(2):04016030, 2017.

[46] Essam Algizawy, Tetsuji Ogawa, and Ahmed El-Mahdy. Real-time large-scale map matching using mobile phone data. *ACM Transactions on Knowledge Discovery from Data*, 11(4):1–38, 2017.

[47] Hamada Rizk, Ahmed Shokry, and Moustafa Youssef. Effectiveness of data augmentation in cellular-based localization using deep learning. In *IEEE WCNC*, 2019.

[48] Kai Zhao, Jie Feng, Zhao Xu, Tong Xia, Lin Chen, Funing Sun, Diansheng Guo, Depeng Jin, and Yong Li. DeepMM: Deep learning based map matching with data augmentation. In *ACM SIGSPATIAL GIS*, 2019.

[49] Yige Zhang, Aaron Yi Ding, Jörg Ott, Mingxuan Yuan, Jia Zeng, Kun Zhang, and Weixiong Rao. Transfer learning-based outdoor position recovery with telco data. *IEEE Transactions on Mobile Computing*, 2020.

[50] Ilias Leontiadis, Antonio Lima, Haewoon Kwak, Rade Stanojevic, David Wetherall, and Konstantina Papagiannaki. From cells to streets: Estimating mobile paths with cellular-side data. In *ACM CoNEXT*, 2014.

[51] Ahmed Shokry, Marwan Torki, and Moustafa Youssef. DeepLoc: A ubiquitous accurate and low-overhead outdoor cellular localization system. In *ACM SIGSPATIAL GIS*, 2018.

[52] Hamada Rizk and Moustafa Youssef. Monodcell: A ubiquitous and low-overhead deep learning-based indoor localization with limited cellular information. In *ACM SIGSPATIAL GIS*, 2019.

[53] Sinem Coleri Ergen, Huseyin Serhat Tetikol, Mehmet Kontik, Raffi Sevlian, Ram Rajagopal, and Pravin Varaiya. RSSI-fingerprinting-based mobile phone localization with route constraints. *IEEE Transactions on Vehicular Technology*, 63(1):423–428, 2013.

[54] Yige Zhang, Weixiong Rao, and Yu Xiao. Deep neural network-based telco outdoor localization. In *ACM SenSys*, 2018.

[55] Heng Qi, Yanming Shen, and Baocai Yin. Intelligent trajectory inference through cellular signaling data. *IEEE Transactions on Cognitive Communications and Networking*, 6(2):586–596, 2019.

[56] Avik Ray, Supratim Deb, and Pantelis Monogioudis. Localization of LTE measurement records with missing information. In *IEEE INFOCOM*, 2016.

[57] Washington Y Ochieng, Mohammed A Quddus, and Robert B Noland. Map matching in complex urban road networks. *Revista Brasileira de Cartografia*, 2(55), 2003.

[58] Axel Mosig and Michael Clausen. Approximately matching polygonal curves with respect to the *fréchet* distance. *European Workshop on Computational Geometry*, 30(2):113–127, 2005.

[59] Xiliang Liu, Liu Kang, Mingxiao Li, and Lu Feng. A ST-CRF map-matching method for low-frequency floating car data. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1241–1254, 2017.

[60] Ali Ufuk Peker, Oguz Tosun, and Tankut Acarman. Particle filter vehicle localization and map-matching using map topology. In *IEEE IV*, 2011.

[61] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. Retrieving similar trajectories from cellular data at city scale. *arXiv preprint arXiv:1907.12371*, 2019.

[62] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Zheng Kai, and Panos Kalnis. Parallel trajectory similarity joins in spatial networks. *VLDB Journal*, 27(3):395–420, 2018.

[63] Chao Chen, Yan Ding, Xuefeng Xie, Shu Zhang, Zhu Wang, and Liang Feng. TrajCompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–17, 2019.

[64] Panrong Tong, Wan Du, Mo Li, Jianqiang Huang, Wenqiang Wang, and Zheng Qin. Last-mile school shuttle planning with crowdsensed student trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[65] En Jian Yao, Long Pan, Yang Yang, and Yong Sheng Zhang. Taxi driver's route choice behavior analysis based on floating car data. In *Applied Mechanics and Materials*, 2013.

[66] Vinay Kumar Sharma, Satyajit Mondal, and Ankit Gupta. Analysis of u-turning behaviour of vehicles at mid-block median opening in six lane urban road: A case study. *International Journal for Traffic & Transport Engineering*, 7(2), 2017.