DMM: A Deep Reinforcement Learning based Map Matching Framework for Cellular Data

Zhihao Shen, Kang Yang, Xi Zhao, Jianhua Zou, Wan Du, Junjie Wu

***Abstract*—This paper presents a novel map matching framework that adopts deep learning techniques to map a sequence of cell tower locations to a trajectory on a road network. Map matching is an essential pre-processing step for many applications, such as traffic optimization and human mobility analysis. However, most recent approaches are based on hidden Markov models (HMMs) or neural networks that are hard to consider high-order location information or heuristics observed from real driving scenarios. In this paper, we develop a deep reinforcement learning based map matching framework for cellular data, named as DMM, which adopts a recurrent neural network (RNN) coupled with a reinforcement learning scheme to identify the most-likely trajectory of roads given a sequence of cell towers. 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 based RNN network for map matching model with variable-length input and output, and a global heuristics-driven reinforcement learning based scheme for optimizing the parameters of the encoder-decoder map matching model. Extensive experiments on a large-scale anonymized cellular dataset reveal that DMM provides high map matching accuracy and fast inference time.**

***Index Terms*—Map matching, Deep reinforcement learning, Location-based services**

I. Introduction

Cellular data are a set of location sequences of cell towers collected by mobile carriers, with which the mobile phones have been associated. They have become a way of taking the pulse of a population, or the pulse of a city and processed for many applications of urban computing and smart cities [1]–[5], such as transportation analysis [2], [3] and human mobility analysis [5]–[7]. An essential processing step of the above applications is map matching that transforms the large-scale offline collected cell tower sequences into road trajectories on a road map. Efficient map matching algorithms are necessary for providing fast processing for large-scale cellular data and minimizing computational resource consumption (e.g., power, storage, and computation). For example, a transportation analysis system that provides decision makers with road information requires to map-match the cell tower sequences from the urban population on a road map.

TABLE I: Comparison over different map matching methods

	HMM	Neural Network	**DMM**
High-order dependency	✗	✓	✓
Local heuristics	✓	✗	✓
Global heuristics	✗	✗	✓
Fast inference	✗	✓	✓

In recent years, two types of machine learning models are widely used to solve map matching tasks, i.e., hidden Markov model (HMM) [8]–[13] and Neural Network (NN) [14]–[17]. Unfortunately, as summarized in Table I, either HMM or NN yields its own advantages and disadvantages for obtaining the map matching model for cellular data.

On one side, the advantage of HMM models is that it can incorporate the heuristics rooted on the observations from real-life scenarios by designing customized emission and transition probabilities. But in the meantime, the HMM-based methods still have several weaknesses. First, they work based on assumption of Markov property, 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 [18], especially when people have a specific destination. As a result, such an assumption leads to the loss of contextual information, and thus reduces the map matching accuracy. Second, under the assumption of Markov property, they can only consider the heuristics observed from local information of current location samples (e.g., preferring major roads near the current sample [19], taking the shortest path between the last and current samples [8]). Third, they often assume to follow the shortest paths between the surrounding roads of two samples, which leads to extensive searches of the shortest paths during inference. This incurs high computational overheads, especially for the low-sampling-rate cell tower sequences, since they will produce large possible road candidates. For a sequence of only 7 cell towers with 68 possible road candidates around each cell tower, the HMM-based methods take 32,368 ($68^2 \times 7$) calculation times of the shortest paths, corresponding to ~ 82.5 seconds of inference.

On the other side, there are also some NN based map matching methods, which directly learn the mappings between location samples and road segments in a data-driven manner, so as to consider high-order location dependency and

Z. Shen, J. Zou are with the School of Electronic and Information Engineering, Xi'an Jiaotong University, Guangdong Xi'an Jiaotong University Academy, and Shaanxi Engineering Research Center of Medical and Health Big Data, Xi'an 710049, China.
E-mail: szh1095738849@stu.xjtu.edu.cn, jhzou@sei.xjtu.edu.cn
K. Yang and W. Du are with the Department of Computer Science and Engineering, University of California, Merced, CA 95340 USA.
E-mail: kyang73@ucmerced.edu, wdu3@ucmerced.edu
X. Zhao is with the School of Management, Xi'an Jiaotong University, and the Key Lab of the Ministry of Education for process control & Efficiency Egineering, Xi'an 710049, China.
E-mail: Zhaoxi1@mail.xjtu.edu.cn
J. Wu is with Beihang University, Beijing Key Laboratory of Emergency Support Simulation Technologies for City Operations, and MoE Key Laboratory of Complex System Analysis and Management Decision.
E-mail: wujj@buaa.edu.cn
Xi Zhao is the first corresponding author and Jianhua Zou is the second corresponding author.

achieve fast map matching during inference. Unfortunately, the above methods adopt one-hot encoding to represent the input locations, which makes it difficult to generalize the learned matching patterns to unknown locations, and cannot use the heuristics observed from real-life scenarios to improve map matching accuracy.

Towards this end, in this paper, we propose a deep reinforcement learning based map matching framework for cellular data, named as DMM. Central to our framework is a recurrent neural network (RNN) [20] model that takes a sequence of cell tower locations as input and infers a trajectory composed of road segments. By directly learning the mappings between cell towers and roads based on training data, it can avoid extensive computations of the shortest paths during inference and reduce computational overheads. The RNN model is also expressive of representing the sequence of cell towers by a hidden vector during inference, which allows to consider multiple previous roads for inferring the next road segment, but not just the last road. In addition, in DMM, the RNN model is coupled with a reinforcement learning (RL) scheme to explore the space of possible better map matching results by considering the real-life heuristics. To transform DMM into a practical system, we tackle a set of challenges.

RNN 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, the 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 similar representations 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) [20] or Gated Recurrent Unit (GRU) [21], 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 based 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 incorporates a number of heuristics to refine the inference model. Besides the heuristics considered in previous work [8] (i.e., staying on the same road), we also propose two novel global heuristics, i.e., preferring to choose a road trajectory with small speed difference between moving speed and road speed limits, and less frequency of turns. To incorporate these heuristics into the map matching model, we customize the basic map matching model into a RL scheme with well-defined reward function, which encourages the map-matched outputs to meet the designed heuristics.

We implement DMM in PyTorch. In order to train DMM, we use an anonymous city-level cellular dataset provided by mobile carriers in a large city. We evaluate DMM with real-world cell tower sequences generated by volunteers traveling more than 1,700 kilometers. The experimental results demonstrate that DMM can not only achieve high accuracy, but also have fast map matching speed. In particular, it provides precision and recall of 82.4% and 86.3%, respectively, corresponding to performance gains of 22.3% and 16.3% over the state-of-the-art approach [8]. In addition, DMM achieves fast inference time about 0.98 second.

In summary, this paper makes the following contributions.

- We develop DMM, a deep reinforcement learning based map matching scheme for cellular data, which takes a sequence of cell tower locations as input and infers a trajectory composed of road segments.
- We customize DMM to tackle a set of challenges, including an encoder-decoder model for input and output sequences of variable length, a spatial-aware representation model for cell towers, and a reinforcement learning scheme for refining output results to meet two novel heuristics observed from real-world driving scenarios.
- Extensive experimental results demonstrate the effectiveness and efficiency of DMM based on a large-scale cellular dataset. We also examine the system robustness in terms of sensitivity to different location characteristics and attributes of input cell tower sequences.

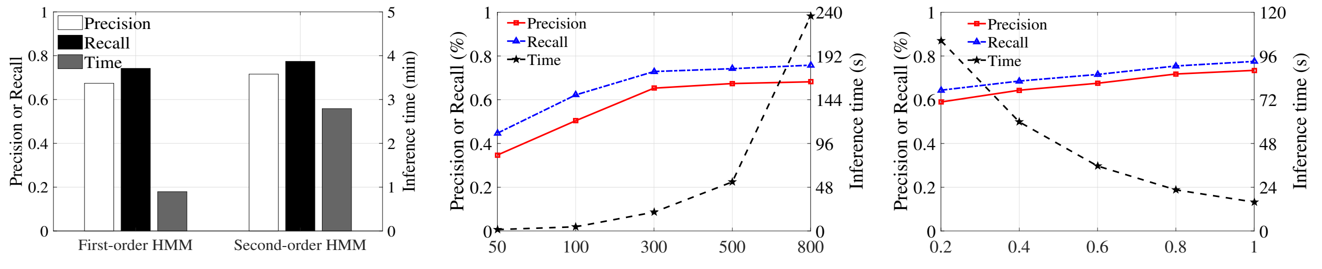
The rest of the paper is organized as follows. In Section II, we introduce the background and motivation of our study. Section III presents the overview of DMM, and the three key modules for accurate and fast map matching. In Section IV, we describe the implementation details of DMM. Experimental setup and results are given in Section V and Section VI. Section VII reviews map matching techniques. In Section VIII, we discuss some key issues in our study and conclude our study in Section IX.

II. Motivation

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

A. Why cellular data

One of the key motivations for DMM is that it uses high-penetrated, low-cost cellular data than GPS. Due to low location error and high sampling rate of GPS sensors, many map matching studies have been proposed for the GPS-based data [9]–[11], [22], [23]. However, there are still some limitations. First, the GPS sensor can only cover a small population or space in a city. Because of large power consumption or privacy concerns, GPS sensors are not installed on all vehicles or enabled by all mobile subscribers. This limits many applications that rely on group behavior analysis of a large amount of users, such as urban planning optimization [2], [24] and human mobility analysis [5], [6], etc. Second, in places with



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

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

poor antenna signals (e.g., indoors, tunnels, basement, etc.), the GPS sensor has bad localization accuracy. For this end, in this work, we investigate a novel map matching framework for the high-penetrated and low-cost cellular data.

B. Problem definition of 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 x is passively recorded by the cellular network infrastructure. Table II describes two examples of the cell tower samples, including several fields, i.e., anonymous user identifier (UserID), timestamp (Time), location area code (LAC), and cell ID (CID). In these fields, The field of UserID is uniquely associated with a mobile phone. The field of Time indicates the timestamp of current cell tower sample. The fields of LAC and CID represent a unique cell tower, where the LAC is used to denote a location area associated by a mobile phone and the CID is used to identify a base transceiver station (BTS) or a sector of BTS within the location area.

Definition 2 - Cell tower sequence. A cell tower sequence is the input of map matching model, 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.

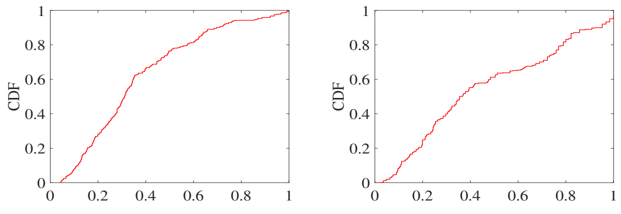
Definition 3 - Road network. A road network can be described as a directed graph $G(V, E)$, where V is a set of nodes on the road network, representing intersections or terminal points, and E is a set of road segments connecting these nodes. In our study, the road network is obtained from a public open-source website (OpenStreetMap [1]). All road information used in DMM can be provided in the downloaded OpenStreetMap road network (e.g., length and speed limits of road segments). In particular, it contains 15,768 nodes and 31,761 edges.

Definition 4 - Candidate road segment. 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 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.

[1]www.openstreetmap.org/

TABLE II: Examples of the cell tower samples

UserID	Time	LAC	CID
1B2A7	201*03*080234	37*6	19*18
5U2F1	201*03*070821	37*9	19*57



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

Fig. 2: Properties of cellular data

Definition 5 - Route. A route Y is the output of map matching model, connecting a sequence of road segments on the road network 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 network $G(V, E)$, a map matching model finds the most likely route Y on G.

C. Limitations of existing map matching methods

Most recent map matching approaches are based on Hidden Markov Models (HMMs) [8]–[13]. They define a hidden state (road segment) and an observable state (cell tower) for the map matching process. Each road segment maintains two probabilities, i.e., emission probability and transition probability. The emission probability evaluates the probability of a cell tower localized at this road segment. The transition probability evaluates the probability that transits from the previous road segment (first-order HMM) or the previous two road segments (second-order HMM) to the current road segment. The existing HMM-based map matching methods define different emission probabilities and transition probabilities. For example, for the emission probability, ST-Matching [10] assumes that the closer roads have the larger emission probability. SnapNet [8] assumes that the major roads have the larger emission probability. For the transition probability, the studies [9], [10],

[25] often assume to follow the shortest path between the surrounding roads of two consecutive cell towers. For an inference, HMM first searches for candidate road segments within a search radius R_C of each cell tower. As the HMM process proceeds, the products of emission probabilities and transition probabilities of the road segments of some routes increase faster than others. In the end, an optimal route with the highest product value can be identified using dynamic programming technique [26].

However, in the above process of using HMM to solve map matching, we find three factors influence the effectiveness and efficiency of the HMM-based models, i.e., order of HMM model, search radius for candidate road segments of cell towers and sampling rate of cell tower sequences. We leverage a state-of-the-art HMM-based approach (SnapNet w/o I in Section V-C) and conduct a series of empirical studies to illustrate why the HMM-based methods are not effective and efficient for processing cell tower sequences. For each experiment, we measure precision, recall and inference time on the same hardware. Specific descriptions of the selected method and experiment settings are introduced in Section V.

Impact of order of HMM model. Considering that HMMs assume the Markov property of problems, the emission probability and transition probability of existing methods can only focus on the local information provided by the road network (e.g., preferring the closer roads [10] or the wider roads [8]), ignoring the contextual information brought by historical or future cell tower samples. However, in real driving scenarios, people often have a specific destination, which leads to human mobility obey Markov property. For example, a person may choose a traveling route according to the number of turns, which requires the map matching process to consider the context information provided by multiple cell tower samples.

To alleviates this issue, a high-order HMM model that considers last several cell towers in the HMM process may work. We explore the accuracy and inference time of map matching models with different orders (e.g., first-order HMM and second-order HMM), as depicted in Figure 1 (a). We find that although accuracy of the second-order HMM is higher than that of the first-order HMM, inference time significantly increases.

Impact of search radius for candidate road segments. The setting of search radius R_C determines to different number of candidate road segments in the HMM process. Less number of 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 settings of R_C in Figure 1 (b). When R_C is small, the accuracy decreases sharply despite fast inference time. As R_C increases, more possible road segments will be considered into the HMM process. This significantly increases the calculation times of the shortest path searches between two cell tower samples, leading to a larger search space and much more inference time, as Figure 1 (b) shown.

We study the location error of cellular data. We depict the Cumulative Distribution Function (CDF) of the location error of collected data (Section V), Location error is measured as the distance between the user's GPS position and the cell tower position. As shown in Figure 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 long inference time.

Impact of sampling rate of cell tower sequences. The sampling rate of cell tower sequences determines the distance between two consecutive cell tower samples, influencing the running time of calculating the shortest paths in the HMM process. We depict the inference time of HMM-based map matching model with respect to different sampling rates in Figure 1 (c). As the sampling rate increases, the inference time decreases sharply. Unfortunately, since the cell tower can only receive signals when requesting location updates or network services, nearly all cell tower sequences have an average sampling rate less than 1 sample per minute, as shown in Figure 2 (b), which depicts the sampling rates of our cell tower sequences. This leads to an infeasible inference time for map matching.

To provide fast map matching, two studies (SnapNet [8] and FMM [25]) make attempts to speed up the HMM process of map matching. In particular, SnapNet [8] increases the sampling rate of cell tower sequences by interpolating some samples between two adjacent cell towers. In this way, 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 cell tower locations. As a result, simple interpolation degrades the map matching accuracy in urban areas. FMM [25] focuses on reducing inference time of the HMM-based methods by pre-computing an origin-destination table to store all pairs of the shortest paths between nodes within a certain range in the road network. Although FMM reduces inference time of the HMM-based models, it is still difficult to make accurate map matching in the cellular environment and encounter performance degradation on inference time for cellular data with large location error, where the number of candidate road segments of each cell tower sample increases greatly, resulting in large increase in calculation times of the shortest paths during the HMM process. We also compare the accuracy and inference time with those two methods in Section VI-A.

Summary. From the above empirical experiments, we conclude that it is difficult for the HMM-based map matching approaches to achieve the best performance on both accuracy and inference time in the cellular environment. Therefore, it is necessary to investigate an effective and efficient map matching model for cellular data.

III. DESIGN OF DMM

In this section, we introduce an overview of DMM and the design of key modules in DMM, i.e., data pre-processor, location representer, map matcher, and RL optimizer.

A. DMM Overview

Figure 3 depicts the architecture of DMM, consisting of two stages, i.e., offline training and online inference. Before the offline training and online inference, we first pre-process the cell tower sequences in the cellular dataset to remove noisy data (Section III-E).

Offline Training. Given the cell tower sequences in the cellular dataset, we first learn a location representer to capture high-quality representations for cell towers (Section III-B).

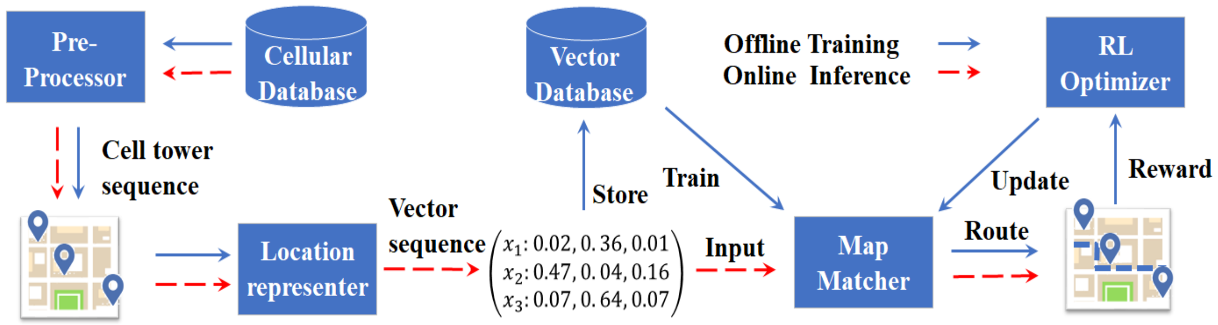


Fig. 3: The architecture of DMM

Based on the location representer, we transform the cell tower sequences into vector sequences and store them in a vector database, which will be used for training map matching model. Then, we learn an RNN-based map matching model to generate the most-likely route on the road network given a vector sequence (Section III-C). The vector sequences as well as the estimated ground truth labels generated from an HMM-based method [8] are used to train the RNN model. Moreover, we customize the map matching model into a reinforcement learning framework to refine the map matching results (Section III-D). 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 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 transforms it to a vector sequence by the location representer and passes the vector sequence into the trained map matching model to identify its most-likely route on the road network.

B. Location representer

Intuitively, DMM can represent a cell tower using two approaches, i.e., one-hot representation and GPS coordinates of the cell tower. For the one-hot representation, we represent a cell tower as a high dimensional binary vector, in which all bits are '0' except one '1' referring to the specific cell tower. However, the one-hot representation suffers from two drawbacks. First, the redundant representation reduces training efficiency of map matching model, especially in the environment with a large number of cell towers. Second, the learned map matching patterns cannot be effectively utilized for unobserved cell towers. For the GPS-based representation, compared with one-hot representation, this method encodes spatial similarity between cell towers inherently. However, it limits the representations in two-dimensional space, which is difficult to be further optimized in parameter space during the training process. Towards this end, we propose to leverage auto-encoder model [27] to automatically learn high-level cell tower representations.

The auto-encoder model leverages a multi-layer neural network to learn identity mapping for same input and output. The middle layer learns high-level representations for cell

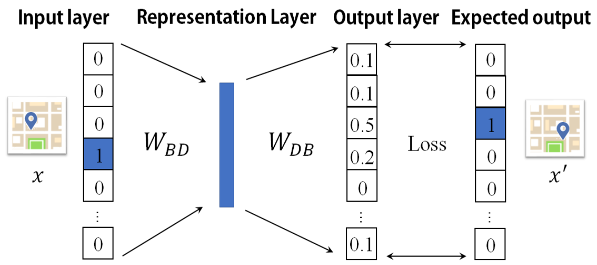


Fig. 4: The architecture of location representer

towers, where the number of hidden neurons is less than that of the input and output layers. However, the basic auto-encoder model is hard to capture spatial-aware feature among cell towers. For this end, we instead use spatially-close cell tower pairs as expected output of the auto-encoder model. In this way, the spatial characteristic of close cell towers can be incorporated into the representations.

Given a cell tower x, we learn a 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 following cell towers in a search radius of the current cell tower.

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

Figure 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 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 cell tower 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 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 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.

C. Map matcher

Inspired by recent advancements on recurrent neural networks (RNNs) for sequential-based applications [21], we design an RNN-based map matcher to learn the mapping between cell tower sequences and road segments on the road network. However, simply applying RNN will not be able to accomplish our goal. First, it is difficult for the basic RNN models to handle the input and output with variable length. Second, it is difficult for the RNN models to conserve historical information of long cell tower sequences, especially those that are longer than the cell tower sequences in the training dataset. Towards this end, we propose a novel architecture for the map matcher in Figure 5, consisting of an RNN encoder-decoder model (blue blocks) and a plug-in alignment component (red block).

1) Encoder-decoder based map matching model: The input of map matching model is a represented cell tower sequence X. An encoder network first transforms the input cell tower sequence X into a sequence of hidden states $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{|X|}$. After encoding the input, the context vector $\boldsymbol{c}$ (the last hidden state $\boldsymbol{h}_{|X|}$) is passed to a decoder network. 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 [21]) is a non-linear function. After encoding the whole cell tower sequence, a continuous vector $\boldsymbol{c}$ (i.e., the hidden state $\boldsymbol{h}_{|X|}$) is generated and served as the input of decoder, which conserves the historical 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.

2) Alignment model: In the above encoder-decoder model, the encoder compresses 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 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

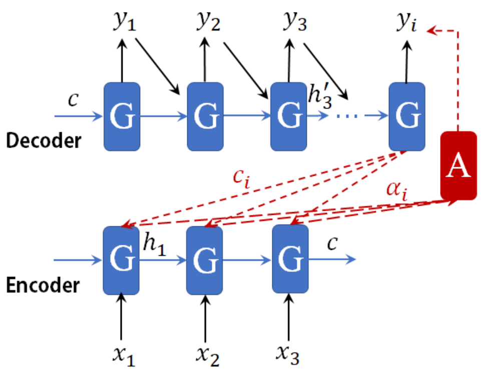


Fig. 5: The architecture of map matcher

the map-matched route jointly. Specifically, the alignment component considers all the 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 allows to handle the long cell tower sequences. Next, we present how the alignment component works.

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

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 practice. We take the second best to generate the labels of cell tower sequences using a state-of-the-art HMM-based method [8]. Although this will bring deficiencies to the map matcher model, it is only used for the initialization for RL optimizer.

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

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

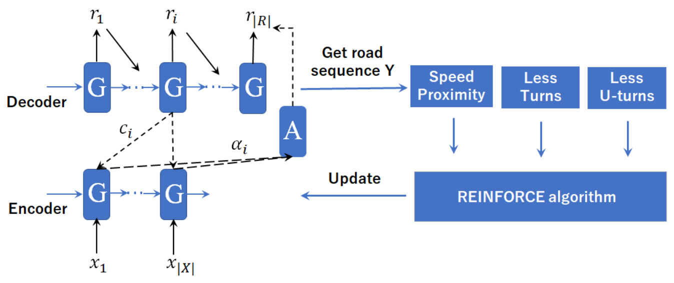


Fig. 6: The architecture of RL optimizer

TABLE III: Key elements of our RL framework

Elements	Meanings
Agent	The map matcher model
State	A sequence of vectors of cell towers
Action	Map-matched road segment at each step
Policy	Neural encoder-decoder network
Reward	The satisfactions of heuristics of map-matched route

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 input cell tower sequence and 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 $t-1$.

D. RL optimizer

To further improve map matching performance, we exploit global heuristics 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 heuristics, inspired by the recent advance of reinforcement learning (RL) approaches [24], [28]–[30], we customize the basic map matching model into a reinforcement learning framework.

1) RL formulation: 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 in Table III. Figure 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 heuristics of route Y, is then computed to assess the quality of output route. Finally, the REINFORCE algorithm [31] 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.

2) Reward design: We incorporate a number of global heuristics into DMM. First, people are more likely to select a sequence of roads with speed limits close to moving speed. Second, people prefer the routes with turns as few as possible if exists multiple possible routes between 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 route, 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 output route, namely, weighted spatial proximity to input cell tower sequence, less frequency of turns, less U-turns. In the following, we present design details for the reward.

Weighted spatial proximity. The reward of spatial proximity r_P needs to ensure that the generated route is spatially-closest to input cell tower sequence, which is in line with the intuition of map matching task. However, due to 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 average weighted projection distance between the input cell tower sequence and the map-matched route as the design of r_P.

Specifically, we calculate the average weighted projection distance between the input cell tower sequence and the map-matched route as follows. First, for each cell tower sample, we calculate basic projection distance to neighboring roads of the cell tower sample by the geodesic distance between the GPS coordinates of cell tower and its projected GPS coordinates on each road. Then, we assign the basic projection distance to each road with a weight w_s. Following the first observation, the road with small speed difference between the moving speed of a user and the speed limit of current road is assigned with a smaller weight to make the projection distance to these roads smaller, and vice versa. In our implementation, the road weight is calculated as $w_s = |v_l - v_c|/v_l$, where v_l is the speed limit of road obtained from the digital map, v_c is the local moving speed of user estimated by the mean value of the speed from the previous sample to the current sample and the speed from the current sample to the next sample.

Less frequency of turns. To avoid the unnecessary turns in the output route, we design a reward r_T, which rewards the route with similar number of turns between the cell tower 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 in the 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 route. 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) REINFORCE algorithm: In terms of the characteristics of encoder-decoder based policy in the agent of map matcher, we adopt the REINFORCE algorithm [31] to refine the policy. 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.

Algorithm 1 Training process of the RL optimizer

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

The training process of RL optimizer is outlined in Algo. 1. We first initialize the parameters of policy $\pi_{\boldsymbol{\theta}}$ with a pre-trained map matcher agent. Given a cell tower sequence X, we generate a route Y based on the policy $\pi_{\boldsymbol{\theta}}$, 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.

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

There may be infinite map-matched routes for a cell tower sequence X. As a result, the expectation of reward $E_{Y\sim\pi_{\boldsymbol{\theta}}(\cdot|X)}$ from the distribution $\pi_{\boldsymbol{\theta}}(\cdot|X)$ cannot be estimated directly. We approximate this expectation by sampling M routes from the distribution $\pi_{\boldsymbol{\theta}}(\cdot|X)$ [28]. To reduce the variance that leads to inaccurate estimation of expected reward, we subtract the reward $r(Y)$ from a baseline b [32]. b is defined as an average reward of the sampled M routes. Then, the gradient can be approximated as Eq. 10.

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

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

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

E. Data pre-processor

Before offline training and online inference, we develop three noise filters to process noisy data.

Ping-Pong filter. A mobile user may be sometimes in the middle of several adjacent cell towers, and their phones perform handover between cell towers frequently in a short period of time. When the filter processes one cell tower sample, it first checks whether the following several samples have Ping-Pong phenomenon, i.e., some samples are from the cell towers that different from current sample and some samples are from the same cell tower as current sample. If these samples do have Ping-Pong phenomenon, the samples from other cell towers are discarded. In addition, we sometimes cannot handle all Ping-Pong noises in one time, we apply the Ping-Pong filter to process the same cell tower sequence iteratively until the number of samples in the cell tower sequence does not decrease.

Backward filter. When a person moves along with a direction, the mobile phone may switch to another cell tower that is opposite to the moving direction. This may be caused by the fluctuation of wireless signals or an imperfect handover algorithm. As a result, the sampled cell tower sequence often

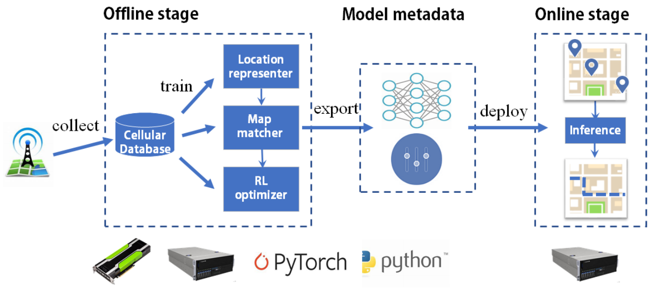


Fig. 7: The workflow of DMM

changes direction at some samples and returns to the normal direction shortly. To remove such type of noise, we develop a backward filter. When the moving direction of one sample is different from the current direction, the filter caches that sample and checks the moving directions of its next several samples. If the direction is changed, confirmed by the following samples, the filter adds all samples to the pre-processing result; otherwise, the filter deletes the cached samples.

Drifting filter. A mobile phone may switch to another cell tower that is far from the position of current cell towers suddenly. As a result, the cell tower fingerprint sometimes moves quickly with an impractical moving speed. To handle this issue, we estimate the moving speed between the current cell tower sample and the next sample, based on the distance of their cell towers and their time intervals. When the moving speed is larger than the road speed limit, the next sample will be discarded.

IV. Implementation

In this section, we introduce implementation details on three models and online inference process of DMM in Figure 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, an open-source machine learning framework.

A. Offline stage

With the cellular dataset provided by mobile carriers, we conduct offline training of DMM. Three models in DMM are trained. Considering that our cellular dataset contains the cell tower sequences sampled from users under static and mobile scenarios with different transportation modes (e.g., cycling, car, etc.), we only retain the cell tower sequences in mobile scenarios for model training. Specifically, after preprocessing the cell tower sequences of each user, we identify staying points of each user using the method in [33] and leverage the cell tower sequences between the staying points for model training. We first train the location representer to obtain the high-quality cell tower representations (Section III-B), and then perform the map matcher model training (Section III-C). Finally, we train the reinforcement learning model to refine map matching results (Section III-D).

Training for the location representer. To train the location representer, we first construct a spatially-close cell tower pair set from our cell tower sequences. For any one cell tower in the cell tower sequence, we obtain 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 current 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 number of cell towers in cell tower set B. The size of hidden unit is 64. Cross entropy loss 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 allows to 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 [8]. The parameters of map matcher are uniformly initialized to $[-0.1, 0.1]$. We use Adam optimizer to update the parameters. Batch size is set as 128. We use Gated Recurrent Unit (GRU) [21] as the RNN units of encoder and decoder networks due to its higher computational efficiency than LSTM [20]. 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 [34] 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 cell tower samples [34]. 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 the 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 of 0.01. We set $\lambda_P = 0.5$, $\lambda_T = 0.25$, $\lambda_U = 0.25$.

B. Online stage

Once the above 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 network.

V. Experimental Setup

In this section, we describe the experimental settings in detail for evaluating DMM, including experimental dataset, performance criteria, and benchmarks.

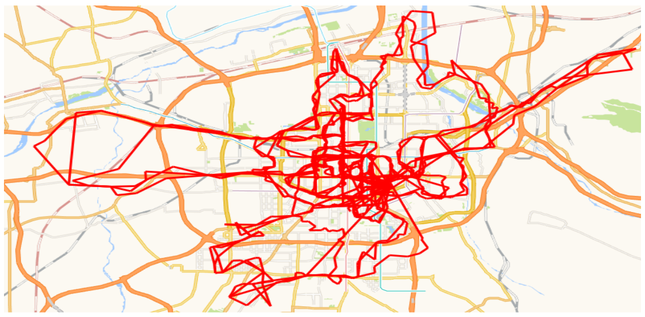


Fig. 8: Coverage map of our collected dataset

A. Experimental dataset

We first describe the evaluation dataset for our map matching framework.

Data collection. We recruited volunteers and collected their data for evaluation. 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[2]) to record GPS locations at a high sampling rate up to 1 sample per second. The mobile carrier also provides the corresponding anonymous cell tower sequences of the volunteers. We map-match all the GPS-based location sequences to obtain the true routes as the ground truth [10]. In the end, we collect 198 car driving traces, with 1,701 kilometers of total length. Figure 8 depicts the distribution of our collected traces. As shown, our data have a dense coverage in the urban area (within 9 kilometers of the city center in our study), especially in the central area of the city, since our volunteers need to go there every day. The collected data are available on the website[3].

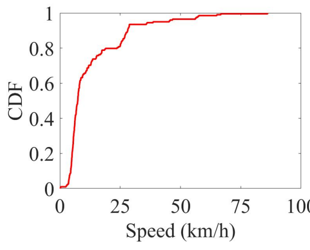
Attributes of the collected data. We describe statistical analysis of our collected dataset according to the following attributes, i.e., moving speed, sampling rate, length, time duration. Figure 9 plots the cumulative distribution functions (CDFs) of these attributes.

As shown in Figure 9 (a) of the distribution of moving speed, we can see that the average moving speeds of 76.8% of the sequences are below 18 km/h, because a large portion of the sequences are collected in urban areas with various traffic conditions. We still have some sequences (23.2%) that have an average speed larger than 18 km/h.

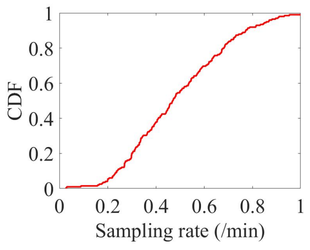
From Figure 9 (b), we can see that 99% sampling rates of the sequences are less than 1 sample per minute. For 5% of the sequences, their sampling rates are less than 0.2/min. By comparing the distributions in Figure 2 (b) and Figure 9 (b), we find that the minimum sampling rate of our collected dataset is relatively higher than that of all subscribers in the city. This is because our collected dataset has a large proportion of the sequences (i.e., 84%) that are collected in urban areas, which usually has the higher density of cell towers and in turn the higher sampling rates of cell tower sequences.

From the distribution of lengths of our cell tower sequences in our dataset (Figure 9 (c)), we can see that the length varies

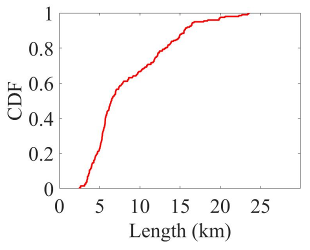
[2]https://play.google.com/store/apps/details?id=net.gotele.gpsbox&hl=zh
[3]https://github.com/zhshen0831/dmm.git



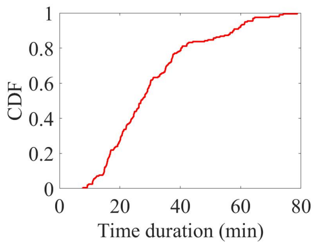
(a) Moving speed



(b) Sampling rate

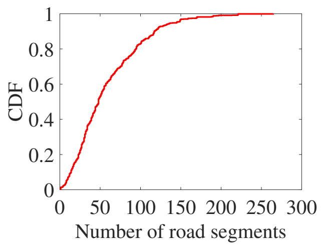


(c) Length

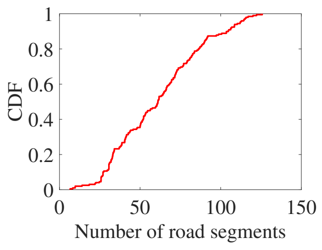


(d) Time duration

Fig. 9: Statistics of the collected cell tower sequences



(a) Road network complexity of cell tower samples



(b) Road network complexity of cell tower sequences

Fig. 10: Location characteristics of the collected cell tower sequences

from 2.5 km to 23.6 km. For 22.7% of the sequences, their lengths are shorter than 5 km. For 87.4% of the sequences, their lengths are shorter than 15 km.

From the distribution of time duration (Figure 9 (d)), we can see that the time duration of the sequences vary from 7.3 minutes to 78.75 minutes. For 77.7% of the sequences, their time durations are shorter than 40 minutes.

Location characteristics of the collected data. We also explore location characteristics of our collected cell tower sequences by analyzing road network complexity around each cell tower sample and cell tower sequence. We first analyze the road network complexity of our collected cell tower samples, which is defined as the number of road segments within radius of 500 meters of each cell tower sample. As shown in the Figure 10 (a), we find that the number of road segments varies from 1 to 265. In addition, about 53.3% of the cell tower samples are surrounded by more than 50 road segments.

We also report the road network complexity of our collected cell tower sequences. Figure 10 (a) shows the CDF of average number of road segments around all cell tower sequences. As shown, the road network complexities of our cell tower sequences are different, with the average number of road segments of each sequence varies from 6.6 to 125.8.

B. Performance criteria

We assess the accuracy of 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. For each cell tower sequence, we calculate the precision and recall as follows.

$$Precision = \frac{\text{Length of the correctly-matched road segments}}{\text{Length of the map-matched route}}$$

$$Recall = \frac{\text{Length of the correctly-matched road segments}}{\text{Length of the ground truth route}}$$

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.

C. Benchmarks

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

- *ST-Matching.* ST-Matching [10] is a widely used HMM-based approach for matching low-sampling-rate GPS-based cell tower sequences, which takes the spatial topological structure of road network and the temporal constraints of moving speed into account simultaneously.
- *SnapNet.* SnapNet [8] 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 [8] 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 Section II-C. 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 the pre-processing steps of SnapNet.
- *FMM.* FMM [25] FMM pre-computes all pairs of shortest paths between nodes under a specific range in the road network, and replaces the shortest path query in HMM with hash table search, so as to reduce the inference time of map matching.

VI. Experimental Results

In this section, we conduct a series of experiments to answer the following research questions.

- RQ1: What is overall online and offline map matching performance of DMM?

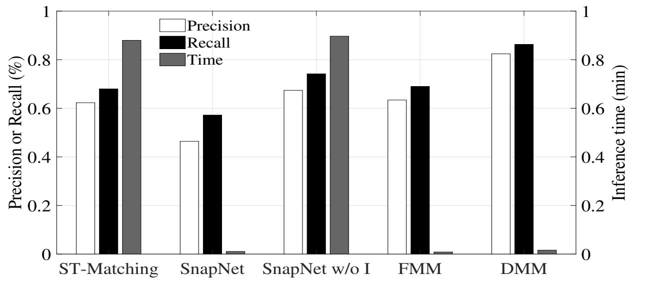


Fig. 11: Online map matching results of different approaches

- RQ2: How do the three proposed modules in DMM contribute to map matching performance?
- RQ3: Is DMM robust against different sampling rates and location characteristics of input cell tower sequences?

A. Overall performance of DMM

In response to RQ1, this section evaluates the overall performance of DMM from the perspective of online inference and offline training respectively.

1) Online map matching performance of DMM: We first evaluate online map matching performance of DMM by comparing to four baselines on our collected dataset.

Accuracy. We use the map matcher in DMM to transform the cell tower sequences of our volunteers into the routes on the road network and compare the generated results with the ground truth. All the 1701-km data are used in the testing stage. Figure 11 depicts the overall map matching accuracy of different approaches. As depicted in Figure 11, we find that DMM provides the best accuracy. For example, DMM provides precision and recall of 82.4% and 86.3%, respectively, corresponding to performance gains of 22.3% and 16.3% over SnapNet. The reasons are as follows. First, DMM adopts an RNN-based model to transform a cell tower sequence into a context vector, which conserves the historical location information for map matching. For the HMM-based approaches, they can only take the last road segment into account for 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 towers. Third, we leverage a reinforcement learning based framework to incorporate the proposed heuristics into the map matching model. In the following, we decompose the performance of location representer, map matcher and RL optimizer in Section VI-B.

Inference time. We use the collected dataset to evaluate the running efficiency of different map matching methods in Figure 11. We have the following observations. First, DMM runs much faster than the basic HMM-based methods (SnapNet and SnapNet w/o I). This is because DMM only needs to make a forward computation of neural networks to identify an optimal route during the inference stage. In contrast, the basic HMM-based approaches rely on heavy computations of the shortest path calculations. When the sampling rate decreases, the moving distance between cell tower samples increases. This results in a slower search process for the shortest path. Second, the inference time of FMM and SnapNet is similar and faster than that of DMM. This is because these two methods accelerate the calculation of the shortest paths during the HMM process through pre-computation or interpolation technology. For example, FMM pre-computes the shortest paths between nodes to avoid the calculations of shortest paths during the HMM process. However, the precision and recall of SnapNet and FMM are worse than those of DMM although those methods have less inference time. This is because FMM only focuses on the efficiency enhancement and SnapNet is more capable of handling the routes on highways. In urban areas, the linear interpolation of low-sampling-rate cell tower sequences introduces large noise between two cell towers.

TABLE IV: Online 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
FMM	0.08	0.11	0.11	0.09	0.10
DMM	0.91	0.74	0.85	1.01	0.92

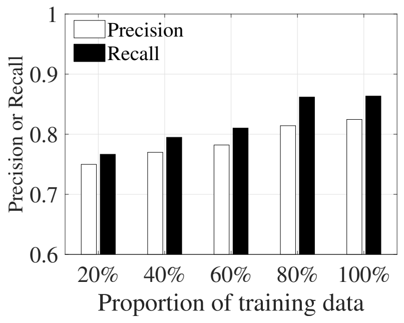
Note that the inference time of FMM is quite different from that reported in their study (about 25,000 samples per second). This is because FMM focuses on the map matching for the GPS-based samples. Because of small location errors, they may have fewer candidate road segments, making map matching speed fast. However, for the cellular data with high location error, the number of possible road segments around each cell tower sample increases greatly, resulting in a significant increase in total number of shortest path queries in the whole map matching process.

We also exploit inference time of different approaches as sampling rate varies. By discretizing the sampling rate into five levels, we obtain results in Table IV. As seen, when the sampling rate is low, DMM can still maintain fast inference. This is because DMM learns the mappings between cell tower samples and road segments directly, and thus is insensitive to sampling rate. In contrast, the inference time of basic HMM models (i.e., ST-Matching and SnapNet w/o I) increases significantly as the sampling rate decreases, since when the sampling rate decreases, the distance between cell tower samples may increase greatly, resulting in too much time spent searching for the shortest path.

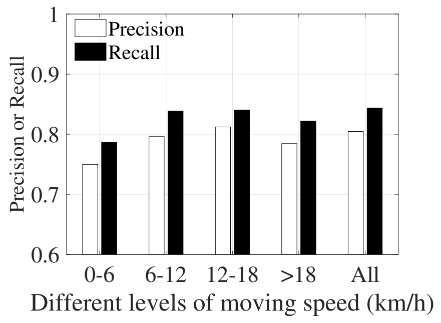
2) Offline training performance of DMM: We also evaluate offline training performance of DMM by investigating training accuracy in terms of two parameter settings (i.e., hidden layer dimension and training data size) and training time.

Training accuracy on different training data sizes. We first investigate the accuracy of DMM trained on different amounts of training data, varying from 20% to 100%. Figure 12 (a) presents the experimental results. As shown, the precision and recall improve rapidly as we increase the training data size from 20% to 80%, and slow down when we continue to increase the training data size. This is because when the training data are enough, more data provide very little enhancement into performance gain.

We also explore the impact of training data with different moving speeds on model accuracy. Considering that our



(a) Different size of training data



(b) Different types of training data

Fig. 12: Offline training accuracy in terms of different training data sizes

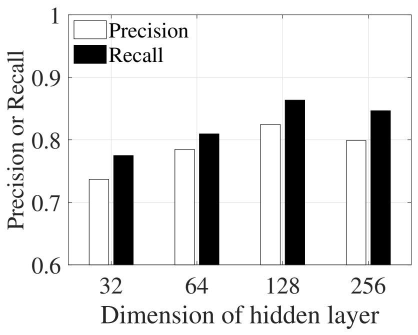


Fig. 13: Offline training accuracy in terms of different hidden layer dimensions

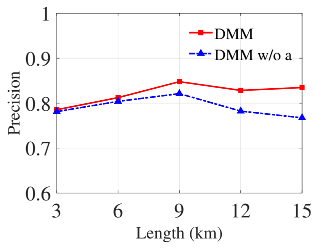
passively-collected training data cannot reveal information of transportation modes (e.g., bicycle, car driving, etc.), we split our dataset into four sub-datasets with different levels of moving speeds (i.e., $\{\geq 0km/h \ \& < 6km/h\}$, $\{\geq 6km/h \ \& < 12km/h\}$, $\{\geq 12km/h \ \& < 18km/h\}$, $\{\geq 18km/h\}$) according to the speed distribution of our testing data. For fair comparison, we use the cell tower sequences in each sub-dataset to train the map matching model and keep the same number of cell tower sequences for different sub-datasets. We also sample a dataset with same number cell tower sequences from our raw dataset randomly to include various levels of moving speed, which is denoted as *All*. Figure 12 (b) depicts the experimental results. As shown, *All* brings the best map matching performance. This may be because training data with different moving speeds can learn more map matching patterns in real mobile scenarios. On the contrary, the model trained with low-speed cell tower sequences has the worst performance. This may be because the cell tower sequences with low speed are more likely to be sampled in the urban area, where the road network density is high. As a result, each cell tower sample has more possible road segments for map matching, making it difficult for the model to determine which road segment actually moves on.

Training accuracy on different hidden layer dimensions. We analyze the impact of hidden layer dimension on training accuracy of DMM. Figure 13 summarizes the impact of the dimension of hidden layer on DMM. We find that increasing the dimension from 32 to 128 significantly improves the accuracy, but shows a slight downward trend when the dimension is larger than 128. This is because higher hidden layer dimension causes over-fitting in the fixed number of training size.

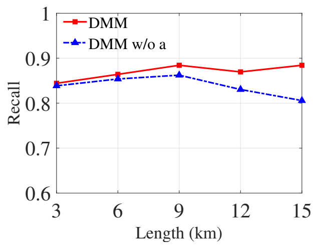
Training time. We also test the training time of DMM. For the location representer, we train it for three epochs on the spatially-close cell tower pairs. For the other two models, we terminate the training processes after the models converge. The training time of location representer, map matcher and RL optimizer take about 14 hours, 22 hours, and 17 hours, respectively. Although the total training time seems long, it only needs to be conducted once for inference.

B. Contribution on our proposed models in DMM

In response to RQ2, this section evaluates the effectiveness of map matcher, location representer, and RL optimizer.



(a) Precision



(b) Recall

Fig. 14: Effectiveness of our map matcher

1) Effectiveness of our map matcher: 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 Figure 14. We find 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 that of the basic encoder-decoder 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 the whole cell tower sequence.

2) Effectiveness of our location representer: We verify the effectiveness of our spatial-aware cell tower representation method in DMM and visualize learned representations of cell towers to better understand our location representer.

Effectiveness of location representer. We implement a variant of DMM, named as DMM w/o LR, which simply uses binary vectors to represent cell towers. As depicted in Figure 15, the precision and recall of DMM w/o LR are 76.5% and 81.6%, 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.

Case study of 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

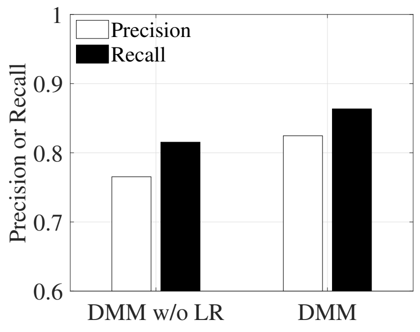


Fig. 15: Effectiveness of location representer

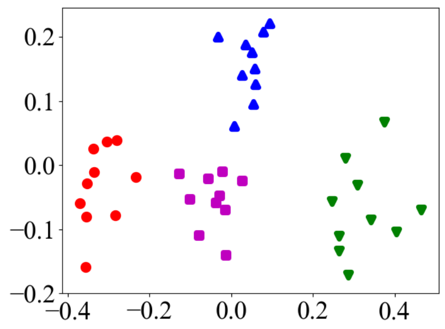


Fig. 16: Spatial proximity of cell towers

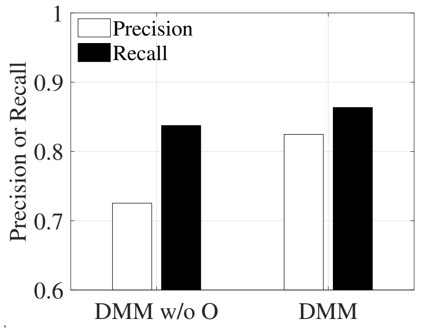


Fig. 17: Effectiveness of the RL optimizer

represented by the location representer. Finally, we use Principal Component Analysis (PCA) technique [35] (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. Figure 16 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.

3) Effectiveness of our RL optimizer: We investigate the performance of RL optimizer and use examples to show how it helps for capturing the proposed heuristics.

Effectiveness of RL optimizer. We first study the performance gain of RL optimizer on the accuracy and report results in Figure 17. We observe that the RL optimizer significantly improves the accuracy of basic map matching model in precision and recall by 13.7% and 3.1%, respectively. This indicates that our reinforcement learning based scheme succeeds in optimizing the map matching model with the heuristics we observed in the real driving scenarios, such as preferring the routes with less turns.

Case study of RL optimizer. The reward $r(Y)$ of a map matching route Y is the weighted sum of three components aimed at capturing the proposed heuristics of the output route, i.e., weighted spatial proximity to the cell tower sequence, less frequency of turns and U-turns. Figure 18 illustrates by examples to show how the 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. From the figure, we demonstrate the effectiveness of three heuristics.

First, we exploit the effectiveness of our first heuristic, i.e., weighted spatial proximity, which rewards the routes that are not only spatially proximity to the input cell tower sequence, but also have moving speeds close to the road limits. As shown in Figure 18 (a), a user drives near two roads, i.e., a national highway (the above yellow road with speed limit 60 km/h) and an expressway (the bottom orange road with speed limit 120 km/h), and generates three cell tower samples (moving speed about 36 km/h). Based on the smaller projected distance between the cell tower sequence and the route, the basic encoder-decoder model considers that the user is moving on the expressway. However, according to the real driving speed of the user, it is unrealistic for the user to drive on the expressway. Towards this end, we optimize the basic map matching model by considering the difference between road speed limit and driving speed, so that the model can estimate that users are more likely to drive on the national highway rather than the expressway.

Next, we exploit the effectiveness of our second heuristic, i.e., 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 [36], we incorporate the hint by a specific design of reward r_T. From Figure 18 (b), we find that DMM can select the route with less turns among multiple possible routes. However, the encoder-decoder model selects the shortest path between two consecutive cell tower samples. This is because the basic model cannot consider the route choice preference.

Finally, we examine the effectiveness of our third heuristic, i.e., less U-turns. Due to large location error of cellular data, the encoder-decoder model identifies the most paths accurately except unexpected U-turns. We use the reward r_U to eliminate this phenomenon. From the results in Figure 18 (c), we find that DMM succeeds in avoiding a U-turn. If the cell tower samples actually indicate an occurrence of U-turn in the raw cell tower sequence, DMM can generate a correct result with U-turns adaptively.

C. DMM Robustness

In response to RQ3, this section investigates the robustness of DMM in terms of sampling rates and location characteristics of input cell tower sequences.

1) Robustness on sampling rate of cell tower sequences: We first explore the impact of sampling rate of input cell tower sequences on map matching performance. However, only exploring model performance at different sampling rates is limited: (i) Due to the difference in moving speed, the movement distances between cell tower samples with different sampling rates vary greatly. As a result, under different movement distances, it is difficult to accurately evaluate the map matching performance. (ii) Due to the differences in factors such as smartphone use frequency and travel distance of different users, the number of cell tower samples in the cell tower sequences is different. When the number of cell tower samples is large, it is possible to use the dependencies between

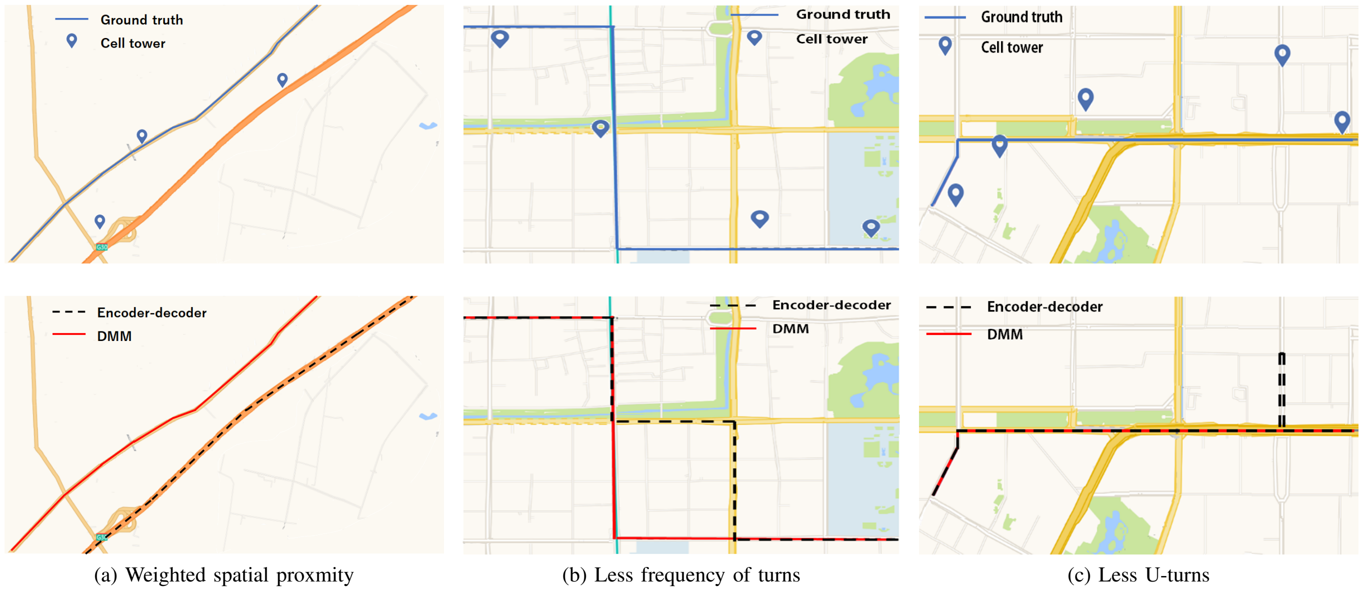


Fig. 18: Case study of RL optimizer, showing the raw cell tower sequences, the ground truth (top), and the map-matched routes of basic encoder-decoder model and DMM (bottom)

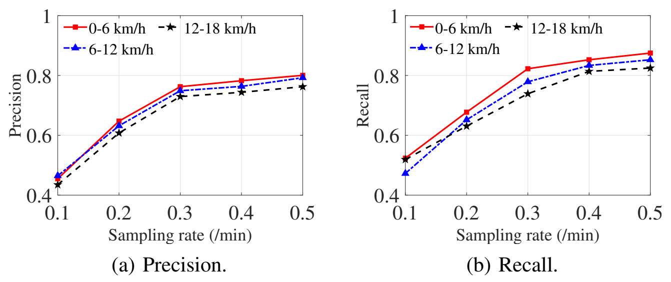


Fig. 19: Performance of DMM in terms of different sampling rates of input cell tower sequences

cell tower samples to improve map matching performance; whereas the number is small, it is difficult to capture the dependencies. Towards this end, in this experiment, we investigate the performance of map matching model for input cell tower sequences with different moving speeds and numbers of cell tower samples at different levels of sampling rates.

Impact of moving speed at different levels of sampling rates. To explore the impact of moving speed at different levels of sampling rates, we process the collected dataset into smaller datasets with different levels of sampling rates and moving speeds. First, we 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. Second, we divide each of the three datasets into five sub-datasets according to the preseted levels of sampling rates (i.e., $0.1/min$, $0.2/min$, $0.3/min$, $0.4/min$, $0.5/min$). 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. Third, we downsample 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 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 Figure 19, 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 43.6% and 51.2%. This is because it is difficult for the map matching model to determine the specific route between the sparse cell towers. In addition, with the increase of the sampling rate or the decrease of the moving speed, DMM provides the 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., 80.1% in precision and 87.2% 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 increase and thus the sampling rate of cell tower sequences will be further increased.

Impact of number of cell tower samples at different levels of sampling rates. We also explore the map matching performance for input cell tower sequences with different numbers of cell tower samples at different levels of sampling rates.

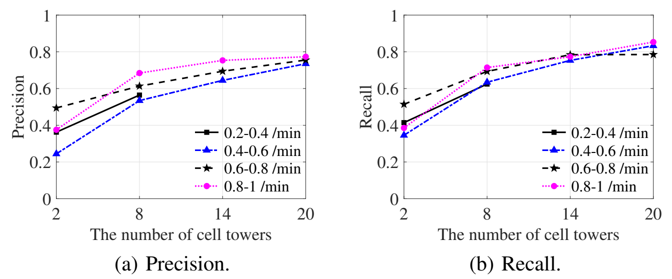


Fig. 20: Performance of DMM in terms of different sampling numbers of input cell tower sequences

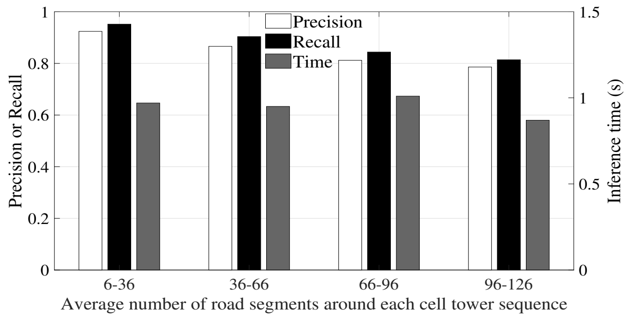


Fig. 21: Performance of DMM in terms of location characteristics of input cell tower sequences

The procedure of processing the collected dataset into smaller datasets with different levels of sampling rates and number of cell tower samples is given as follows. First, we partition the cell tower sequences into four datasets with five levels of sampling rates (i.e., $0.2/min$, $0.4/min$, $0.5/min$, $0.8/min$, $1/min$). Then, for each sequence in the four datasets, we generate a set of cell tower sequences with different number of cell tower samples by connecting the sequence between the first cell tower sample and the remaining cell tower samples. We keep the cell tower sequences with four levels of numbers of cell tower samples, 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 Figure 20, 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 cell tower sequence of two cell towers achieves 24.4% in precision and 34.6% 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 the 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 routes is about 13.33 min and average length is about 2 km with an average speed about 9 km/h), DMM can achieve 61.6% in precision and 69.2% in recall. This is because longer sequences contain more location information that can be used for map matching.

2) Robustness on location characteristics of cell tower sequences: This experiment investigates the impact of location characteristics on map matching performance. By dividing the collected cell tower sequences into 4 sub-datasets according to average number of road segments around each cell tower sequence, we obtain the experimental results in Figure 21. We can find that the map matching accuracy shows a downward trend when the road network complexity around the cell tower sequences becomes higher. This is because when there are a large number of road segments near the cell tower sequences, more feasible candidate road segments are considered into the inference process, making it difficult to determine the most likely route. In contrast, when the average number of road segments is small, a user is more likely to drive in remote areas, which have less feasible routes for map matching.

VII. Related work

A. Map matching for cellular data

Many studies [8], [16], [17], [25], [37]–[39] have explored map matching using cellular data. Algizawy *et al.* [38] extend the typical HMM for map matching cellular-based data for traffic analysis. CTrack [37] proposes a grid-based HMM approach to identify the most likely roads. However, the inference time of these methods is very slow. To provide fast map matching, several studies have been proposed to speed up the searching process of the shortest paths. For example, SnapNet [8] develops an HMM-based model for map matching in view of the road information. FMM [25] proposes an algorithm combining hidden Markov model and precomputation technique. By pre-calculating the shortest paths between the nodes on the road network and storing it in a hash table, FMM avoids repeated routing queries and greatly improves the map matching performance. However, these approaches cannot consider high-order historical location information. In this work, we develop an RNN-based model to directly learn the mappings between cell towers and road segments, which not only avoids extensive computations of the shortest paths during inference but also considers multiple historical roads for inferring next road segment.

Meanwhile, several studies [12], [14], [15], [40], [41] have been proposed to localize the measurement record (MR) data collected by cellular network infrastructures. The types of MR data include sector information, signal latency, signal strength, signal quality, etc. $Cell^*$ [40] and CTS [12] estimate location using sector information. DeepLoc [14] localizes the accurate position using ubiquitous cellular signals received from adjacent cell towers. Ergen *et al.* [42] develop an HMM-based localization model based on the received signal strength indicator sent by adjacent cell towers. RecuLSTM [15] develops a deep learning based framework for location estimations based on measurement records. However, these data are not available in our dataset.

B. Map matching for GPS data

Besides cellular data based map matching studies, many approaches are also developed for GPS data [9]–[11], [22],

[23]. Mosig *et al.* [43] apply $Fr\acute{e}chet$ distance for map matching, but they cannot consider road network hints. Many advanced algorithms, such as conditional random field [23], particle filter [44] and hidden Markov model [9]–[11], are developed to deal with complex road networks. For example, ST-Matching [10] map-matches GPS trajectories with spatial and temporal information. However, these approaches cannot be used in DMM because of large location error and low sampling rate of cellular data.

C. Map matching applications

Map matching can be used as a fundamental step for many trajectory mining applications [2], [19], [45], [46]. VTrack [45] leverages an HMM-based map matching scheme to estimate road traffic. TS-Join [47] proposes a network-based trajectory similarity join by mapping massive trajectories on the road. Prokhorchuk *et al.* [2] infer travel time distributions based on map-matched floating car data. TrajCompressor [46] designs a trajectory compression framework, along with the first pre-processing step of map matching.

VIII. 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 select a sequence of roads with speed limits close to the moving speeds. The assumption is confirmed by vehicle speed transportation statistics for Great Britain [4], where the average speeds are observed at sampled Automatic Traffic Counters (ATC) locations, showing that as the applicable speed limits for road types increase, the average driving speed of each vehicle type becomes higher. Second, we assume that people normally prefer the routes with less frequency of turns between origin and destination. Venigalla *et al.* [36] 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 [48]. Mondal *et al.* analyzed the vehicles at six areas and showed that 93.4% of drivers prefer straight roads.

Privacy issues. We use the cellular dataset provided by mobile carriers to train the models in DMM. The data have been anonymized to protect privacy by replacing the identifiers by hash codes. The data only contain anonymous 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 collect the GPS locations and cellular data from volunteers for evaluation. We anonymize users' identifiers in our dataset. We inform the volunteers of the experimental details and obtain their consents to use the data for this study.

Limitations. DMM has several limitations. First, in order to ensure high precision and recall, the higher sampling rates of cell tower sequences (larger than 0.2/min in the urban area) are required for our system (Figure 19). It will be better to extend our system, where cell tower density and mobile app usages will be further increased in the future. Second, DMM targets the driving scenario with long moving distance and time. The scenario of short-distance or short-time movement (e.g., walking) remains to be explored. Third, compared with the HMM models that have little deployment cost, DMM requires at least about 0.6 million anonymous cell tower sequences for effective offline training and a heavy training process on sufficient hardware resources (e.g., GPU). One possible way to address this problem is to employ effective data augmentation technique [39] to learn the model. Fourth, 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. Fifth, DMM focuses on the map matching problem for coarse-grained cellular data. For other location data with higher positioning accuracy (e.g., GPS-based data), there may be a large number of possible inputs (GPS coordinates), which leads to the issue of model convergence. As a result, DMM cannot be directly applied to the data with higher positioning accuracy. In the future, the map matching ability of DMM with respect to different positioning accuracy of location sensors is worth exploring.

[4]https://www.gov.uk/government/statistics/vehicle-speed-compliance-statistics-for-great-britain-2020

IX. Conclusion

In this paper, we develop a deep reinforcement learning based map matching framework for 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 accuracy and fast inference time.

References

[1] Z. Feng and Y. Zhu, "A survey on trajectory data mining: Techniques and applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.

[2] A. Prokhorchuk, J. Dauwels, and P. Jaillet, "Estimating travel time distributions by Bayesian network inference," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.

[3] F. Calabrese, M. Colonna, P. Lovisolo, D. Parata, and C. Ratti, "Real-time urban monitoring using cell phones: A case study in Rome," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 141–151, 2010.

[4] S. Liu, Y. Yue, and R. Krishnan, "Non-myopic adaptive route planning in uncertain congestion environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2438–2451, 2015.

[5] R. Becker, K. Hanson, S. Isaacman, M. L. Ji, M. Martonosi, J. Rowland, S. Urbanek, A. Varshavsky, and C. Volinsky, "Human mobility characterization from cellular network data," *Communications of the ACM*, vol. 56, no. 1, pp. 74–82, 2013.

[6] Z. Liu, Z. Li, K. Wu, and M. Li, "Urban traffic prediction from mobility data using deep learning," *IEEE Network*, vol. 32, no. 4, pp. 40–46, 2018.

[7] E. Thuillier, L. Moalic, S. Lamrous, and A. Caminada, "Clustering weekly patterns of human mobility through mobile phone data," *IEEE Transactions on Mobile Computing*, vol. 17, no. 4, pp. 817–830, 2018.

[8] R. Mohamed, H. Aly, and M. Youssef, "Accurate real-time map matching for challenging environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 847–857, 2017.

[9] G. Hu, J. Shao, F. Liu, Y. Wang, and H. T. Shen, "If-Matching: Towards accurate map-matching with information fusion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 114–127, 2016.

[10] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009, pp. 352–361.

[11] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *ACM SIGSPATIAL GIS*, 2009.

[12] X. Huang, Y. Li, Y. Wang, X. Chen, Y. Xiao, and L. Zhang, "Cts: A cellular-based trajectory tracking system with gps-level accuracy," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–29, 2018.

[13] G. R. Jagadeesh and T. Srikanthan, "Online map-matching of noisy and sparse location data with hidden markov and route choice models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2423–2434, 2017.

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

[15] 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*, 2018, pp. 307–308.

[16] J. Feng, Y. Li, K. Zhao, Z. Xu, T. Xia, J. Zhang, and D. Jin, "Deepmm: deep learning based map matching with data augmentation," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2372–2384, 2020.

[17] Y. Zhang, A. Y. Ding, J. Ott, M. Yuan, J. Zeng, K. Zhang, and W. Rao, "Transfer learning-based outdoor position recovery with telco data," *IEEE Transactions on Mobile Computing*, 2020.

[18] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map matching: Facts and myths," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 484–487.

[19] Z. Shen, W. Du, X. Zhao, and J. Zou, "Retrieving similar trajectories from cellular data at city scale," *arXiv preprint arXiv:1907.12371*, 2019.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[22] W. Y. Ochieng, M. A. Quddus, and R. B. Noland, "Map matching in complex urban road networks," *Revista Brasileira de Cartografia*, vol. 2, no. 55, 2003.

[23] X. Liu, L. Kang, M. Li, and L. Feng, "A ST-CRF map-matching method for low-frequency floating car data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1241–1254, 2017.

[24] Y. Wei, M. Mao, X. Zhao, J. Zou, and P. An, "City metro network expansion with reinforcement learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2646–2656.

[25] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2018.

[26] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

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

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[29] X. Ding, W. Du, and A. Cerpa, "Octopus: Deep reinforcement learning for holistic smart building control," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2019, pp. 326–335.

[30] Z. Shen, K. Yang, X. Zhao, J. Zou, and W. Du, "Deepapp: A deep reinforcement learning framework for mobile application usage prediction," *IEEE Transactions on Mobile Computing*, vol. 22, no. 02, pp. 824–840, 2023.

[31] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[32] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *International Conference on Learning Representations*, 2016.

[33] S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky, "Identifying important places in people's lives from cellular network data," in *International Conference on Pervasive Computing*, 2011, pp. 133–151.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[35] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[36] M. Venigalla, X. Zhou, and S. Zhu, "Psychology of route choice in familiar networks: Minimizing turns and embracing signals," *Journal of Urban Planning and Development*, vol. 143, no. 2, p. 04016030, 2017.

[37] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod, "Accurate, low-energy trajectory mapping for mobile devices," in *USENIX NSDI*, 2011.

[38] E. Algizawy, T. Ogawa, and A. El-Mahdy, "Real-time large-scale map matching using mobile phone data," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 4, pp. 1–38, 2017.

[39] H. Rizk, A. Shokry, and M. Youssef, "Effectiveness of data augmentation in cellular-based localization using deep learning," in *2019 IEEE Wireless Communications and Networking Conference*. IEEE, 2019, pp. 1–6.

[40] 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 *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 121–132.

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

[42] S. C. Ergen, H. S. Tetikol, M. Kontik, R. Sevlian, R. Rajagopal, and P. Varaiya, "RSSI-fingerprinting-based mobile phone localization with route constraints," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 1, pp. 423–428, 2013.

[43] A. Mosig and M. Clausen, "Approximately matching polygonal curves with respect to the fréchet distance," *Computational Geometry*, vol. 30, no. 2, pp. 113–127, 2005.

[44] A. U. Peker, O. Tosun, and T. Acarman, "Particle filter vehicle localization and map-matching using map topology," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 248–253.

[45] A. Thiagarajan, L. Ravindranath, K. Lacurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones," in *ACM SenSys*, 2009.

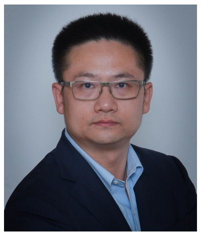
[46] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng, "Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 2012–2028, 2019.

[47] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, "Trajectory similarity join in spatial networks," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, 2017.

[48] V. K. Sharma, S. Mondal, and A. 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*, vol. 7, no. 2, 2017.

Zhihao Shen received his B.E. degree in automation engineering from School of Electronic and Information, Xi'an Jiaotong University, Xi'an, China, in 2016, where he is currently pursuing the Ph.D. degree with the Systems Engineering Institute. His research interests include big data analytics, mobile computing and deep reinforcement learning.

Kang Yang received his B.E. degree in automation engineering from School of Electrical and Control Engineering, Xi'an University of Science and Technology, Xi'an, China, in 2016, and the M.E. degree in control engineering from School of Electronic and Information, Xi'an Jiaotong University, Xi'an, China, in 2019. He is now the Ph.D. candidate of Department of Computer Science and Engineering, University of California, Merced. His research interests include the Internet of Things, mobile computing.



Xi Zhao was awarded his Ph.D. degree in computer science from the Ecole Centrale de Lyon, Ecully, France, in 2010. He conducted research in the fields of biometrics and pattern recognition as a Research Assistant Professor with the Department of Computer Science, University of Houston, USA. He is currently a Professor with the Xi'an Jiaotong University, Xi'an, China. His current research interests include mobile computing and behavior computing.

Jianhua Zou received the Bachelor's, Master's, and Doctor's degrees from the Huazhong University of Science in 1984, 1987, and 1991, respectively. He is currently Professor in Xi'an Jiaotong University. His main research areas include: control systems and computer networks, multimedia, cognition and knowledge discovery, high voltage insulation monitoring, and complex system analysis. Since 1991, he has been the project leader and has completed about 20 research projects, including National Natural Science projects.

Wan Du is currently an Assistant Professor at the University of California, Merced. Before moving to Merced, Dr. Du had worked as a Research Fellow in the School of Computer Science and Engineering, Nanyang Technological University, Singapore 2012-2017. He received the B.E. and M.S. degrees in Electrical Engineering from Beihang University, China, in 2005 and 2008, respectively, and the Ph.D. degree in Electronics from the University of Lyon (cole centrale de Lyon), France, in 2011. His research interests include the Internet of Things, cyber-physical system, distributed networking systems, and mobile systems.

Junjie Wu received the PhD degree in management science and engineering from Tsinghua University. He is currently a full professor with Information Systems Department, Beihang University, the director of the Research Center for Data Intelligence (DIG), and the director of the Institute of Artificial Intelligence for Management. His research interests include data mining and complex networks. He was the recipient of the NSFC Distinguished Young Scholars Award and MOE Changjiang Young Scholars Award in China.