# L2MM: Learning to Map Matching with Deep Models for Low-Quality GPS Trajectory Data

LINLI JIANG*, CHAO-XIONG CHEN*, and CHAO CHEN†, Chongqing University, China

Map matching is a fundamental research topic with the objective of aligning GPS trajectories to paths on the road network. However, existing models fail to achieve satisfactory performance for low-quality (i.e., noisy, low-frequency, and non-uniform) trajectory data. To this end, we propose a general and robust deep learning-based model, **L2MM**, to tackle these issues at all. Firstly, high-quality representations of low-quality trajectories are learned by two representation enhancement methods, i.e. *enhancement with high-frequency trajectories* and *enhancement with the data distribution*. The former employs high-frequency trajectories to enhance the expressive capability of representations while the latter regularizes the representation distribution over the latent space to improve the generalization ability of representations. Secondly, to embrace more heuristic clues, typical mobility patterns are recognized in the latent space and further incorporated into the map matching task. Finally, based on the available representations and patterns, a mapping from trajectories to corresponding paths is constructed through a joint optimization method. Extensive experiments are conducted based on a range of datasets, which demonstrate the superiority of **L2MM** and validate the significance of high-quality representations as well as mobility patterns.

CCS Concepts: • **Human-centered computing** → **Ubiquitous computing**; • **Computing methodologies** → *Machine learning algorithms*.

Additional Key Words and Phrases: map matching, trajectory representation learning, mobility pattern, deep learning, trajectory data

**ACM Reference Format:**
Linli Jiang, Chao-Xiong Chen, and Chao Chen. 2022. L2MM: Learning to Map Matching with Deep Models for Low-Quality GPS Trajectory Data. *ACM Trans. Knowl. Discov. Data.* 0, 0, Article 0 (July 2022), 25 pages. https://doi.org/XX

## 1 INTRODUCTION

With the popularization and application of location acquisition technologies, such as GPS, the massive GPS trajectory data are collected and widely used in various areas. However, affected by challenging environments, raw GPS trajectories data often suffer from three key issues, i.e., noise, low-frequency and non-uniformness, resulting in an imprecise and uncertain positioning. For example, tall buildings in the urban environment block a number of GPS satellites and cause GPS positioning errors [8]; limited energy and transmission-bandwidth restrict that GPS locations can be only sampled at a low frequency; and different GPS device settings and unpredictable

*Both authors contributed equally to this work and share the first authorship.
†This is the corresponding author.

The work was supported by the National Natural Science Foundation of China (No. 62172066 and 61872050), and also supported by DiDi GAIA Research Collaboration Plan.
Authors' address: Linli Jiang, linlijiang010@gmail.com; Chao-Xiong Chen, cxchen@cqu.edu.cn; Chao Chen, cschaochen@cqu.edu.cn, Chongqing University, Chongqing, China, 400044.


1556-4681/2022/7-ART0 $15.00
https://doi.org/XX

communication failures may generate non-uniform sampling time intervals as well. Hence, map matching is proposed to solve the problem of snapping raw GPS trajectory to the road networks to identify the true moving path and true position of moving objects. It is a fundamental pre-processing step for many downstream location-based services and trajectory-based applications, such as navigation [21, 41], travel time estimation [37], route optimization [46]. The task of map matching is of great significance for many applicaitons. However, it is extremely hard of map matching models to achieve the satisfactory performance for such low-quality trajectories.

In the literature, extensive attention has been paid to map matching for low-frequency GPS trajectories with noise. The Hidden Markov Model (HMM) and its variants [6, 29, 35, 40] are popular and widely used. To reduce the uncertainty in a sparse trajectory, some approaches employ additional data to improve the matching accuracy, such as historical GPS trajectory or third-party trajectory data [17]. The performance of these algorithms is usually unsatisfactory due to many factors, e.g., the assumptions of predefined models [34], the frequency of data sampling and the complexity of the road network [15]. Note that non-uniformly sampled trajectories are common in reality [16], but the effect of non-uniform sampling is under-explored. As the boom in machine learning in recent years, it has encouraged researchers to solve the problem from the data-driven perspective, resulting in learning-based map matching models. For instance, Feng et al. [9] propose a deep learning-based model (i.e., DeepMM) for matching sparse and noisy trajectories. Notably, learning-based models are data-driven, which means they are sensitive to data quality. With noise, low-frequency and non-uniformness, low-quality GPS trajectories pose a great challenge in developing excellent learning-based models. Worse still, learning-based models are data-hungry. Thus, it is another challenge that they are sensitive to the training data size.

To address these challenges, we need to develop a general and robust model. In this paper, we propose a deep learning-based map matching model for low-quality GPS trajectories (i.e., **L**earning **TO** **M**ap **M**atching, **L2MM** for short hereafter). Deep learning models are proven to be powerful tools that can automatically learn knowledge from massive data. However, it is impractical to apply them directly to the map matching task. For example, to reduce the harmful effects of noise points in the geographic space, it is natural to consider using embedding techniques (e.g., recurrent neural networks) to encode trajectories as high-dimensional vectors in the latent space. But for trajectories with noise, low-frequency and non-uniformness, the simplistic and rude approach will only produce low-quality trajectory representations, resulting in unsatisfactory performance. To be specific, **L2MM** is inspired by the following observations.

**Observation I**: The matching accuracy degrades significantly as the sampling time interval increases, for most of the existing map-matching algorithms (e.g., DeepMM [9]). This observation inspires us to enhance the expressive capability of representations for low-frequency trajectories. Intuitively, a low-frequency trajectory loses most of the movement details. It contains less information about the real driving path, probably resulting in an ambiguous representation. In contrast, a high-frequency trajectory contains more details, potentially providing more explicit clues to indicate the real path that a vehicle traverses. Thus, we expect to embed the rich spatial information of the high-frequency trajectory into the representation of the low-frequency trajectory. To this end, we employ a seq2seq model to learn trajectory embedding representations for low-quality trajectories. In this manner, the representation vector is able to contain both features of the low-frequency trajectory and characteristics of the high-frequency trajectory, thus enhancing its expressive capability.

**Observation II**: The matching accuracy of learning-based models (e.g., DeepMM [9]) is sensitive to the size of training data, which is extremely low when using a small amount of training data. This observation motivates us to improve the generalization ability of trajectory representations. After encoding, trajectories are projected and scattered in the latent space. Each trajectory representation

corresponds to a driving path once it is decoded and reconstructed. Due to the limited size of training data, they are sparsely distributed over the latent space, which performs poorly in providing effective clues about the real path for an unseen trajectory. Inspired by variational Auto-Encoder (VAE), we regularize the distribution of representations over the latent space to strengthen the generalization ability. In this way, the distribution of trajectory representations can be more concise and regular, facilitating the map matching for an unseen trajectory potentially. Thus, we can use limited trajectory data to train the model, without severely degrading model performance.

**Observation III**: Mobility patterns and user preferences extracted from real driving scenarios can improve the matching results [9, 33]. This observation encourages us to incorporate prior knowledge into the map matching task. For instance, drivers prefer main roads with turns and U-turns. Although such knowledge is helpful to find the real driving paths, it is too implicit and abstract. Worse still, how to make use of them in the map matching task is also unclear. To this end, we first uncover the typical mobility patterns hidden in historical trajectories in the latent space. They are integrated into map matching as knowledge through a joint optimization scheme.

This paper presents the full model design, architecture, details of the different components, as well as extensive evaluations. In short, the main contributions can be summarized as follows:

- We design a general and robust deep learning-based model to solve the map matching problem for low-quality GPS trajectories. It is more robust to low sampling rates, non-uniform sampling rates, and noise. Moreover, it can achieve quite good performance, even with a small amount of training data.
- We propose two representation enhancement methods to learn high-quality trajectory representations for low-quality GPS trajectories. To the best of our knowledge, we are the first to improve the performance of map matching by enhancing the capability of trajectory representations.
- We develop a pattern recognition method to explore patterns hidden in historical trajectories based on the learned representations, and explicitly incorporate them into the map matching by employing a joint optimization scheme. Note that it is the first attempt to introduce pattern recognition as an important component of the map matching task, to improve the generalization ability of the proposed model.
- We conduct extensive experiments on both real-world and synthetic datasets. Results show that **L2MM** is superior to previous methods in accuracy and efficiency. Moreover, it also demonstrates the significance and effectiveness of each designed component.

The remainder of this paper is organized as follows. Section 3 introduces the definitions, problem statement and the overview of **L2MM**. Section 4 elaborates details on each component in **L2MM**. Section 5 presents the results on extensive comparison experiments. Section 2 reviews the related work. Section 6 concludes this paper and discusses future directions.

## 2 RELATED WORK

### 2.1 Map Matching

Affected by challenging environments [27], GPS trajectories generally suffer from data quality issues, which makes the task of map matching extremely difficult. Most of the works focus on map matching for low-frequency GPS trajectories with noise. Existing map matching algorithms can be roughly divided into two categories, i.e., conventional model-based methods and emerging learning-based models.

Traditional model-based map matching algorithms can be further divided into several categories. Some methods make use of various information, such as spatial-temporal features [24, 26], heading direction [3, 15], moving speed [16], trajectory similarity [1], driver behavior [43], to find the correct

paths. Among them, HMM and its variants [3, 6, 18, 27, 29, 35, 40] are widely used in mapping low-frequency trajectory data. Under the Markov assumption, HMM-based algorithms can achieve good matching accuracy. To reduce the uncertainty in low-frequency trajectories, Mohamed et al. [27] use a linear interpolation on two consecutive GPS points. The route choice model is introduced to reassess generated paths along with a set of feasible alternative paths in [18]. However, human mobility is non-Markovian [33, 34]. Besides, they cannot avoid costly path searches on road networks, resulting in poor performance for low-sampling-rate trajectories. Some approaches employ additional data to improve the matching accuracy. For example, frequent mobility patterns identified from historical GPS trajectory are used to improve the matching performance in [17]. In summary, the performance of these model-based algorithms is usually unsatisfactory due to many factors, e.g., the assumptions of predefined models [34], the frequency of data sampling and the complexity of the road network [15]. More importantly, non-uniformly sampled trajectories are common in reality [16], but the effect of non-uniform sampling is under-explored.

To overcome the limitations of conventional model-based methods, researchers try to solve the map matching problem from the data-driven perspective. Based on the collected massive trajectory data, learning-based map matching models come into being with the help of emerging machine learning techniques, e.g., $k$-nearest neighbors algorithm (kNN) [13], ANN [39], RL [33, 47]. For example, Shen et al. [33] employ a RL optimizer to further improve the matching results. Based on the seq2seq learning framework, DeepMM [9] is proposed for matching sparse and noisy trajectories, which learns the mapping function from a trajectory to the corresponding driving path with enormous historical trajectories. For a more comprehensive survey on map-matching algorithms, readers can refer to [2, 14, 21, 32].

DeepMM is a state-of-the-art method among learning-based methods in recent years. The basic workflow is as follows. It first converts the raw low-frequency trajectories into latent representation vectors, followed by projecting the vectors to ground truth trajectories. Since the dataset is too small to train the model, two data augmentation methods are proposed to enrich the training dataset. It is more robust to noise with the help of augmented trajectories and embedding techniques. Besides, it can also handle the matching of low-frequency and non-uniform trajectories. DeepMM is a learning-based model driven by data. It is thus sensitive to data quality and also data-hungry. To be more specific, without considering the quality of trajectories, DeepMM simply encodes raw trajectories into representation vectors, resulting in poor-quality trajectory representations. Without enough training data, it cannot achieve good performance. To solve these problems, we propose a general and robust deep learning-based model for low-quality GPS trajectories. Based on corresponding high-frequency trajectories, it can learn high-quality trajectory representations for low-quality trajectories. We regularize the distribution of representations over the latent space to strengthen the generalization ability of the representations. By exploring and incorporating patterns hidden in historical trajectories, we further improve the generalization ability of the proposed model. In this way, **L2MM** is more robust to the training data size. It can achieve quite good performance, even with a *relatively* small amount of training data.

### 2.2 Trajectory Representation Learning

Trajectory representation learning is to embed both spatial and structure information of a trajectory into a high-level and low-dimensional vector in the latent space. Trajectory representation learning plays a significant role in many trajectory mining tasks based on deep learning, including trajectory similarity computation [22, 42], human mobility prediction [11, 23, 49], travel mode detection [19, 45], driving style identification [4], and so on. In our **L2MM**, trajectory representation learning can reduce the impact of noise while preserving the information of the trajectory. Specifically, RNN is widely applied to generate the representation vector [9]. It processes each point in a trajectory and

output a hidden state sequentially. Usually, the last hidden state is regarded as the representation vector. The expressive capability is restricted to the information contained in the trajectory. Due to the limited information in the low-frequency trajectory, the representation vector generated by the traditional trajectory learning method is semantically ambiguous, resulting in the poor performance in map matching. Thus, we propose to enhance the trajectory representation capability to improve the matching accuracy for low-frequency trajectories.

### 2.3 Mobility Pattern Incorporation

Trajectories contain the mobility patterns of drivers in the city. The mobility pattern exhibits sequential transition regularities and human mobility characteristics, providing useful insights for many trajectory mining tasks. To name a few, Chen et al. [3] utilized the idea that people tend to go straight at intersections to compress trajectory. Liu et al. [25] recognized major mobility patterns to detect anomalous trajectories. Also, mobility pattern can provide heuristic hints for map matching as evidenced by [9, 30, 33]. To be more specific, based on traversed paths, Osogami and Raymond [30] utilized inverse reinforcement learning to learn transition patterns between road segments, and further estimated the parameters of the map matching algorithm. Feng et al. [9] attempted to apply a seq2seq model for map matching, where mobility patterns embedded in training trajectories and corresponding paths are implicitly exploited to support the matching of new-coming trajectories. Shen et al. [33] directly used the high-level mobility pattern knowledge (e.g., people prefer to take the major roads) to optimize the map matching results. **L2MM** is different from them in two aspects: (1) while previous studies do not recognize and represent mobility patterns, **L2MM** seeks to explicitly uncover and represent mobility patterns, and (2) while previous studies do not explicitly exploit mobility patterns to guide map matching, **L2MM** fully exploits them as heuristic hints to facilitate map matching.

## 3 PRELIMINARY

### 3.1 Definitions

Definition 1 (Point-based Trajectory). *A point-based trajectory is a sequence of chronologically ordered GPS points, denoted as* $T^p = \{p_1, p_2, ..., p_n\}$*, where* $p_i$ *is a GPS point. It is represented as* $p_i = (lat_i, lon_i, t_i)$*, where* $lat_i$*,* $lon_i$ *and* $t_i$ *correspond to the latitude, longitude and sampling timestamp, respectively. The sampling time interval (*$\Delta t$*) is defined as the time difference between two consecutive GPS points.*

The point-based trajectory is generated by the GPS-enabled moving vehicles, which is also known as the *raw* trajectory. It should be noted that $\Delta t$ varies across different GPS devices installed in vehicles of a city, due to different deployment time, device brands and maker, and etc. Worse still, it also varies even for the same GPS device, due to unpredictable communication failures [22]. As a result, the sampling time interval of the trajectory data in a city can be *non-uniform*. A trajectory with a big value of $\Delta t$ is equivalent to a low-frequency trajectory, which is denoted as $T^{pL}$. Otherwise, it is defined as a high-frequency trajectory and denoted as $T^{pH}$. For vehicle-based trajectories, the threshold value of 30 seconds is generally used to distinguish them [31].

Definition 2 (Grid-based Trajectory). *A grid-based trajectory is a sequence of grid cells, denoted as* $T^g = \{g_1, g_2, ..., g_n\}$*, where* $g_i$ *refers to a grid cell in a city. It is defined on top of and corresponds to a given point-based trajectory.*

Compared to the point-based trajectory, the grid-based trajectory is more tolerant to the positioning noise caused by GPS devices. Thus, such trajectory representation is commonly used in the trajectory data mining tasks [19, 22, 36, 42]. It is easy to transform a given point-based trajectory to

a grid-based one, mainly including the *city division* and *point-to-grid identification*. The whole city is firstly divided into a number of equal-sized grid cells, and the grid cell that each GPS point seats is identified according to their GPS coordinates. In this manner, two different GPS points may be identified with the same grid cell. The number of grid cells is exactly the same to the number of GPS points. Thus, a grid-based trajectory can contain repetitive grid cells. The sampling timestamps contained in the point-based trajectory are no longer retained in the grid-based trajectory. Similar to the high-frequency and low-frequency point-based trajectory, we use symbols $T^{gH}$ and $T^{gL}$ to denote high-frequency and low-frequency grid-based trajectory, respectively.

Definition 3 (Road Network). *A road network is a directed graph $G = (N, E)$ consisting of nodes and edges, where $N$ denotes the set of nodes including intersections and dead-ends; $E \subseteq N \times N$ denotes the set of directed edges. Each node $n_i$ in $N$ is represented by a pair of longitude and latitude coordinates, representing its spatial location. Each edge $e_i$ in $E$ refers to a directed segment and can be represented by a pair of ordered nodes.*

In this study, the road network is obtained from a public open-source website called OpenStreetMap (OSM). Although road networks of many world-wide cities can be freely available from the OSM platform, they also suffer from the data quality issue, including error/missing edges. Such issue becomes quite severe especially for cities undergoing rapid development. As a result, the map matching performance can vary a lot across different cities due to diverse qualities of the provided road network. Although the quality of the road network itself plays a crucial role in the performance of map matching [28], we simply leave it as a separated research problem.

Definition 4 (Segment-based Trajectory). *A segment-based trajectory is a sequence of connected road segments in a city road network, denoted as $T^s = \{e_1, e_2, ..., e_m\}$, where $e_i$ and $e_{i+1}$ should be physically connected through a shared node in the road network. It indicates the true driving path in real cases when the driver generated the GPS trajectory data.*

As discussed above, for a given point-based trajectory, it is trivial to obtain the corresponding grid-based trajectory. However, both transformations from the point-based trajectory and the grid-based trajectory to the segment-based trajectory are challenging, which are also our main focus in this paper.

### 3.2 Deep Learning-Based Map Matching Problem

The general objective of the map matching problem is to identify the segment-based trajectories ($\{T^s\}$) in the road network ($G$) for the set of given raw GPS trajectories ($\{T^p\}$). Applying deep learning based methods for map matching is *completely data-driven*, and it can be divided into two stages, i.e., the offline training and online inference respectively. The offline training stage aims to learn a mapping function $f : \{T^p\} \rightarrow \{T^s\}$ with a number of training samples in which each sample contains a pair of a $T^p$ and its corresponding $T^s$. Note that obtaining and labelling true driving paths on the road network from raw trajectories are usually labor-intensive and time-consuming. Moreover, learning a mapping function from the training data correctly is extremely challenging. To reduce the potential side-effect caused by the spatial noise from GPS devices, the set of point-based trajectories is commonly transformed to the set of grid-based trajectories in advance. The deep models are then trained given the set of $\{(T^g, T^s)\}$ pairs in the training data. The offline training stage is quite time-costly, needing a huge amount of computing resources and training data. As a comparison, the online inference stage is much easier and faster. With the mapping function contained by the well-trained deep models, once given a point-based trajectory, it is able to return the mapped result (i.e., the true driving path) accurately in a cost-effective manner.

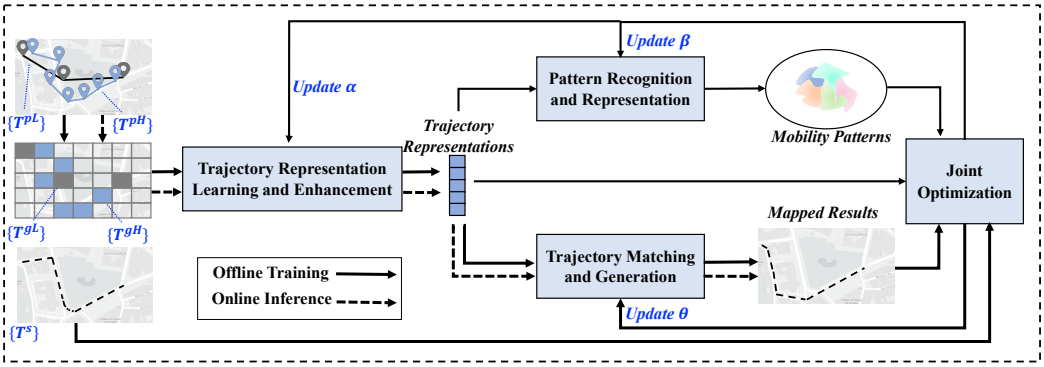



Fig. 1. Overview of **L2MM**.

### 3.3 L2MM Overview

Figure 1 overviews the architecture of **L2MM**, mainly consisting of four components, i.e., *Trajectory Representation Learning and Enhancement, Pattern Recognition and Representation, Trajectory Matching and Generation, and Joint Optimization.* As discussed before, **L2MM** has two working stages, including the offline training and the online inference. In the offline training stage, these four components work cooperatively to train the deep models for map matching. While in the online inference stage, with the readily available deep models, only two components (i.e., trajectory representation learning and enhancement, trajectory matching and generation) of them take participation to work and return the mapped result for the given test point-based trajectory. Below we will briefly introduce how different components work together in these two stages.

**Offline Training.** Given the point-based trajectories in the training dataset, they are initially converted to the grid-based trajectories. We learn and enhance the trajectory representation in the latent space using high-frequency trajectories and data distribution to make it more robust to the spatial noise, data sparseness, as well as the size of training data in the component of *Trajectory Representation Learning and Enhancement* (see details in Section 4.1). Since the majority mobility patterns of drivers being encoded in the generated trajectories implicitly are proved to be effective in improving the quality of map matching [9, 33], we recognize and represent them based on the learned trajectory representations in the component of *Pattern Recognition and Representation* (see Section 4.2 for details). On the other hand, the learned trajectory representations is fed into the component of *Trajectory Matching and Generation* (see details in Section 4.3) to generate the corresponding mapped results. Finally, the mapped results together with segment-based trajectories correspond to the true driving paths, the learned trajectory representations as well as the recognized patterns are inputted to the component of *Joint Optimization* (see details in Section 4.4) to update the parameters (i.e., $\alpha$, $\beta$, $\theta$) of the previous components. Readers can refer to the solid lines with arrows shown in Fig. 1 for the procedure of the offline training stage.

**Online Inference.** In this stage, point-based trajectories that can be with either high-frequency or low-frequency are continuously fed into **L2MM** for the driving routes inference. For a given point-based trajectory, after transforming to a grid-based trajectory, **L2MM** obtains its trajectory representation (a vector sequence) in the latent space with the trained deep model in the component of *Trajectory Representation Learning and Enhancement.* The trajectory representation is then passed to the trained deep model in the component of *Trajectory Matching and Generation* to identify the most-likely segment-based trajectory (i.e., mapped result). The procedure of the online inference stage is illustrated using the dashed lines with arrows in Fig. 1.

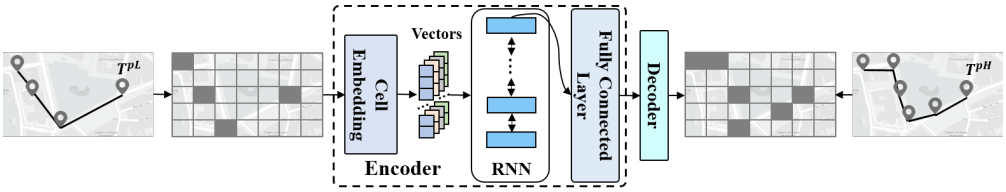



Fig. 2. Illustration of learning trajectory representation enhancement with the high-frequency trajectories.

## 4 DESIGN DETAILS

### 4.1 Trajectory Representation Learning and Enhancement

Trajectory representation learning is to convert the raw trajectories in the physical space into representation vectors in the latent space, which are more robust to the noise [9]. It serves as the fundamental role in deep learning models for map matching. However, the generated representation that governs the features of the trajectory is not satisfactory, either due to the poor quality of the trajectory data or the limited size of training data. More specifically, the low-quality point-based trajectory is very sparse and noisy, which contains scant information about the true driving path. The great uncertainty in the trajectory leads to an ambiguous semantic representation, undermining the expressive capacity of the latent vector. Worse still, the limited size of trajectory data impairs the generalization ability of the latent representation. To address those two issues, we respectively propose two methods to enhance the capability of trajectory representation, i.e., *Enhancement with the High-Quality Trajectories*, and *Enhancement with the Data Distribution*, detailed as follows.

**Enhancement with the High-Frequency Trajectories.** The movement of vehicles on the same path can be recorded at different sampling rates, generating different point-based trajectories of low and high frequency. The limited information as well as noise in the low-frequency trajectory lead to great uncertainty when finding the true moving path. Though trajectory representation learning could reduce the impact of noise, the inherent weakness of scant information results in an ambiguous semantic representation. On the contrary, the high-frequency trajectory contains more details of the movement of the vehicle. The rich spatial information holds explicit semantics, indicating the road segment where the vehicle is located. This inspires us to enhance the latent representation of the low-frequency trajectory, by incorporating the rich information of the corresponding high-frequency trajectory. To this end, we propose to leverage the seq2seq model to automatically learn high-quality trajectory representations.

The seq2seq model employs deep neural networks to learn the mapping from the input sequence to the target sequence, which is widely used in the field of Natural Language Processing (NLP) [44]. In our model, the low-frequency trajectory is taken as the input sequence and the corresponding high-frequency trajectory is regarded as the target sequence. As shown in Fig. 2, it mainly consists of two modules: an encoder and a decoder. The former maps the variable-length input sequence to a fixed-dimensional context vector, and the latter decodes the target sequence from the vector. The generated context vector not only embeds the features of the input sequence but also integrates the characteristics of the target sequence.

Specifically, the grid-based trajectory $T^{gL} = \{g_1, g_2, \ldots, g_n\}$ derived from a low-frequency point-based trajectory $T^{pL} = \{p_1, p_2, \ldots, p_n\}$ is fed to the encoder $q_\alpha(\mathbf{z}|T^{gL})$ to generate the latent representation $\mathbf{z} \in \mathbb{R}^D$, where $\alpha$ denotes the parameters of the encoder and $D$ represents the dimensionality of the latent space of trajectories. In particular, the encoder $q_\alpha(\mathbf{z}|T^{gL})$ is implemented with a cell embedding layer, a Recurrent Neural Network (RNN) layer, and a fully connected layer. The cell embedding layer transforms each grid (i.e., $g_i$) into a one-hot vector. The RNN layer reads

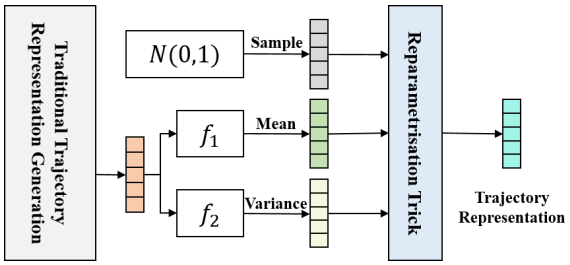



Fig. 3. Illustration of learning trajectory representation enhancement with the data distribution.

the trajectory $T^{gL}$ successively and produces a sequence of hidden state $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$. In the $i$-th step of encoding, it reads $g_i$ and $\boldsymbol{h}_{i-1}$ to update the hidden state $\boldsymbol{h}_i \in \mathbb{R}^D$, as shown in Eq. 1.

$$\boldsymbol{h}_i = \text{GRU}\left(\boldsymbol{h}_{i-1}, g_i\right) \tag{1}$$

where Gated Recurrent Unit (GRU) is a non-linear function to be learned. The normal GRU only extracts features from the preceding part of the trajectory and ignores the information in the future trajectory. In order to enhance the capability of feature extraction, we use the bidirectional GRU as the basic unit to capture complex sequential information in the trajectory from both forward and backward directions. Thus, we obtain the $2D$-dimensional hidden state $\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}_i} \oplus \overleftarrow{\boldsymbol{h}_i}\right]$. Finally, a fully connected layer is used to transform the shape of hidden states, i.e., $\boldsymbol{h}_i \in \mathbb{R}^D$. The last hidden state $\boldsymbol{h}_n$ is regarded as the context vector (i.e., the latent representation). Note that it is often to use the encoder separately to produce trajectory representation in previous studies[9, 33].

Given the context vector $\mathbf{z}$, the decoder is to generates the recovered trajectory $T^{gH'}$. The decoder is comprised of three layers: a RNN layer, a fully connected layer, and a softmax layer. The RNN layer employs a unidirectional GRU to generate an output successively based on the context vector. Then, the fully connected layer is applied to transform the output vector into an appropriate shape. Finally, the softmax layer is used to identify the optimal grids to produce the matched trajectory. We optimize the loss between the recovered trajectory $T^{gH'}$ and the expected trajectory $T^{gH}$, which is the corresponding grid-based trajectory derived from the paired high-frequency point-based trajectory $T^{pH}$. By this way, rich details of the high-frequency trajectory can be learned and incorporated into the context vector, thus enhancing its expressive capability.

**Enhancement with the Data Distribution.** Learning the mapping from trajectories to the corresponding paths is driven by a huge amount of training data. However, training data are limited or even insufficient due to sampling cost and storage limitation. The limited size of training data critically degrades the performance of learning-based models. In previous studies, methods of data augmentation and data generation are used to extend the training dataset to alleviate the problem [9]. Here, we propose to enhance the trajectory representation by leveraging the distribution of representation vectors in the latent space, without expanding the dataset.

After preprocessing by the seq2seq model, low-frequency trajectories are encoded as fix-dimensional context vectors. They are points scattered in the high-dimensional latent space. Each point could correspond to a segment-based trajectory once it is decoded and reconstructed. Owing to the limited size of training trajectories, they are sparsely distributed over the latent space, which perform poorly in providing effective clues about the real driving path to an unseen trajectory. Intuitively, the distribution of these points should have some regularity. First, similar trajectory inputs should be encoded as adjacent points. Second, points decoded to the same segment-based trajectories

should be as close as possible in the latent space. Therefore, we reconstruct data distributions by leveraging the existing points over the latent space.

Motivated by VAE [20], we regularise the organization of the latent space by adding a regularisation term. With this regularisation term, we encourage data distributions to draw close and guarantee to satisfy the expected regularity. Such regularisation term is expressed as the Kulback-Leibler (KL) divergence between the data distribution and a standard Gaussian distribution, which enforces the data distribution to be close to a standard Gaussian distribution. Specifically, the data distribution of $\mathbf{z}$ is inferred from a posterior distribution, as shown in Eq. 2.

$$\mathbf{z} \sim q_\alpha(\mathbf{z}|T^g) = N(\boldsymbol{\mu}_{T^g}, \boldsymbol{\sigma}^2_{T^g}\mathbf{I}) \tag{2}$$

where $q_\alpha(\mathbf{z}|T^g)$ denotes the latent representation of a given trajectory $T^g$ derived from the encoder, $\mathbf{I}$ is an identity matrix, $\boldsymbol{\mu}_{T^g} \in \mathbb{R}^D$ and $\boldsymbol{\sigma}_{T^g} \in \mathbb{R}^D$ are the mean and standard deviation vectors respectively. To be specific, $\boldsymbol{\sigma}_{T^g} = f_1(\boldsymbol{h}_n)$ and $\boldsymbol{\sigma}^2_{T^g} = f_2(\boldsymbol{h}_n)$ where $f_1$ and $f_2$ are two fully connected feed-forward neural networks to be learned, $\boldsymbol{h}_n$ denotes the generated context vector of the trajectory $T^g$, as depicted in Fig. 3.

Note that the random sampling during the training may prevent the error to be backpropagated through the network [20]. To resolve this issue, we employ a reparametrisation trick to make the gradient descent possible. By reconstructing the data distribution, we strengthen the generalization ability of the latent representation under a limited size of training data. Even given an unseen trajectory, it can be encoded as a meaningful point in the latent space, thus obtaining more effective clues about the real driving path and improving the quality of map matching.

### 4.2 Pattern Recognition and Representation

Pattern recognition and representation is to mine patterns hidden in trajectories based on the learned high-quality representations. The trajectory data records users' trips on the urban road network, which contains the mobility regularity of users in the city, such as common routes, preferences of the different user groups. At the same time, it also implies the layout of the urban road network. For example, there are special highways, beltways, and airport expressways in the city. Such knowledge has been implicitly proved to be effective in improving the quality of map matching [9, 33]. Thus, we intend to recognize typical patterns and further introduce them into map matching.

As trajectories are encoded as points in the high-dimensional latent space, pattern mining is performed based on these points. In general, clustering techniques can be directly used to discover the patterns, regardless of the representation generalization process. But the result is not satisfactory, since it is an unsupervised task and the key information to distinguish different patterns may be removed in the trajectory embedding process [50]. Hence, we take pattern recognition as an integral part of the map matching task and automatically identify patterns through the supervised training of **L2MM**.

As mentioned, trajectories can reflect patterns of users' mobility and road layouts in the geographic space. They are encoded as points in the high-dimensional latent space. Correspondingly, trajectory representations should also keep the same regularity in the latent space. Assuming that there are $K$ patterns hidden in trajectory data, there are $K$ corresponding groups in the latent space. To achieve this, we use a Gaussian mixture distribution as the expected representation distribution over the latent space. It grants us the opportunity to segregate our latent space into multiple classes [7] and each class corresponds to a pattern. We assume that the trajectory representation of each pattern follows the standard Gaussian distribution in the latent space [25]. Accordingly, a trajectory representation $\mathbf{z}$ in a pattern $c$ can be formulated as:

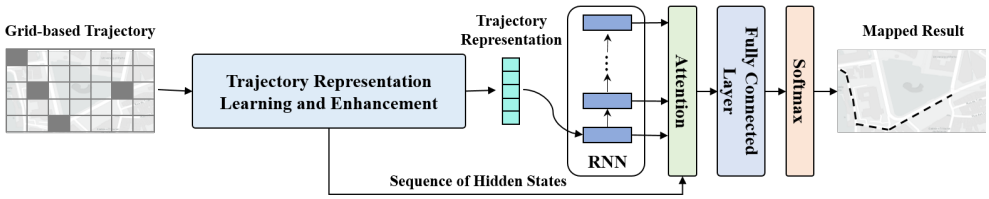



Fig. 4. The process of trajectory matching and generation.

$$p_\beta(\mathbf{z}|c) = N(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2\mathbf{I}) \tag{3}$$

where $\boldsymbol{\mu}_c \in \mathbb{R}^D$ and $\boldsymbol{\sigma}_c \in \mathbb{R}^D$ are the mean and standard deviation vectors, respectively. $p_\beta(c) = Mult(\boldsymbol{\tau})$ is a multinomial distribution to describe the probability of trajectories belonging to the pattern $c$, where $\boldsymbol{\tau} \in \mathbb{R}^K$ are parameters to be learned. Therefore, the distribution of the representation over the latent representation space can be modeled as:

$$p_\beta(\boldsymbol{z}) = \sum_{i=1}^{K} p_\beta(z|c)p_\beta(c) \tag{4}$$

where $\sum_{i=1}^{K} p_\beta(c) = 1$ and $\beta = \{\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c, \boldsymbol{\tau}\}$ denotes all the parameters in the Gaussian mixture model.

Given a trajectory $T^g$, recognizing the pattern $c$ and the representation vector $z$ is modeled as $q_\alpha(\mathbf{z}, c|T^g)$. With the help of the mean-field approximation, it can be factorized as:

$$q_\alpha(\mathbf{z}, c|T^g) = q_\alpha(\mathbf{z}|T^g)q_\alpha(c|T^g) \tag{5}$$

where $q_\alpha(\mathbf{z}|T^g)$ is calculated according to Eq. 2, and $q_\alpha(c|T^g)$ models the the probability of a trajectory $T^g$ belonging to the specific pattern $c$. Since the trajectory $T^g$ is encoded as a latent representation $\mathbf{z}$, we have $q_\alpha(c|T^g) := p_\beta(c|\mathbf{z})$. According to Bayes' theorem [12], the posterior distribution can be defined as:

$$p_\beta(c|\mathbf{z}) = \frac{p_\beta(c)p_\beta(\mathbf{z}|c)}{p_\beta(\mathbf{z})} \tag{6}$$

We take the pattern recognition as an integral part of the map matching task pattern, by designing a pattern-aware loss function (see Eq. 13 in Section 4.4). Once the model is trained appropriately through the joint optimization method, the Gaussian mixture distribution of the latent representation is determined, i.e., the underlying patterns hidden in trajectories are found. Meanwhile, the distribution of trajectory representations in the latent space is regularised by the identified patterns. In this way, we incorporate the knowledge of patterns into map matching.

### 4.3 Trajectory Matching and Generation

Trajectory matching and generation is to decode the trajectory representation and map the decoded vector to the geographic space, generating the matched segment-based trajectory successively. Formally, trajectory matching and generation is a decoder $p_\theta(T^s|\mathbf{z})$, where $\mathbf{z}$ is a latent representation vector, $T^s = \{e_1, e_2, ..., e_m\}$ denotes the generated segment-based trajectory, and $\theta$ represents the set of decoder parameters. Moreover, we employ the attention mechanism to augment the decoder to capture long and complicated dependencies in the trajectory.

At the beginning, the Start Of Sequence (*SOS*) token and the latent representation $\boldsymbol{z}$ are passed to the decoder. At the $i$-step of decoding, the segment $e_i$ is identified based on the previous generated segments $e_{<i} = \{e_1, e_2, ..., e_{i-1}\}$, until the output segment is the End OF Sequence (*EOS*) token. As

shown in Fig. 4, the decoder is implemented with a RNN layer, an attention layer, a fully connected layer, and a softmax layer. To be specific, we first utilize a unidirectional GRU to encode the last identified segment $e_{i-1}$ and the last hidden state $\boldsymbol{s}_{i-1}$ into a new hidden vector $\boldsymbol{s}_i \in \mathbb{R}^D$. Note that $e_0 = SOS$, $\boldsymbol{s}_0 = \boldsymbol{z}$, and $\boldsymbol{s}_i$ is updated based on:

$$\boldsymbol{s}_i = \text{GRU}\left(e_{i-1}, \boldsymbol{s}_{i-1}\right) \tag{7}$$

To improve the performance of the decoder, the attention layer is also applied to search for the most relevant representation vectors, generating an adaptive context vector. Given the sequence of hidden states $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$ from the *Trajectory Representation Learning and Enhancement*, the adaptive context vector $\mathbf{c}_i \in \mathbb{R}^D$ is the weighted sum of all the hidden states, defined as:

$$\mathbf{c}_i = \sum_{j=1}^{|T^g|} (\omega_{ij} \boldsymbol{h}_j) \tag{8}$$

where $\boldsymbol{h}_j$ is the $j$th hidden state, $|T^g|$ gets the length of the grid-based trajectory, and $\omega_{ij}$ measures the importance of $\boldsymbol{h}_j$, which can be computed as:

$$\omega_{ij} = \frac{\exp\left(f(\boldsymbol{s}_i, \boldsymbol{h}_j)\right)}{\sum_{k=1}^{|T^g|} \exp\left(f(\boldsymbol{s}_i, \boldsymbol{h}_k)\right)} \tag{9}$$

where $f(\boldsymbol{s}_i, \boldsymbol{h}_j)$ is a score function calculating the correlation between the current hidden state $\boldsymbol{s}_i$ and a historical hidden state $\boldsymbol{h}_j$. Then, $\boldsymbol{s}_i$ and $\boldsymbol{c}_i$ are concatenated, and further fed to a fully connected layer to ensure a proper shape. Finally, it is passed to the softmax layer to identify the matched segment $e_i$, which can be defined as:

$$p_\theta(e_i | e_{<i}, \mathbf{z}) = softmax(F([\boldsymbol{s}_i \oplus \boldsymbol{c}_i])) \tag{10}$$

Here, $\oplus$ is the concatenation operation, and the function $F(\cdot)$ converts the concatenated vector to a $|\mathcal{G}|$-dimensional vector, where $|\mathcal{G}|$ obtains the total number of grids in the city. In this way, a segment-based trajectory can be sequentially generated by following Eq. 10.

### 4.4 Joint Optimization for L2MM

To train **L2MM**, we aim to jointly optimize three components in **L2MM**, i.e., *Trajectory Representation Learning and Enhancement*, *Pattern Recognition and Representation*, and *Trajectory Matching and Generation*. As a result, the last component can properly produce a segment-based trajectory with the latent representation, which is derived from a sparse point-based trajectory through the first component. Meanwhile, the latent trajectory distribution is regularized and determined, and the patterns behind trajectories are discovered in the latent space.

The objective of **L2MM** is to maximize the joint probability distribution of the generated segment-based trajectory (i.e., the sequence of output segments). More formally, it is the log-likelihood of generating segment-based trajectories in the training data:

$$\max \frac{1}{N} \sum_{i=1}^{N} \log p_\theta \left(T_i^s | T_i^g\right) \tag{11}$$

where $N$ represents the total number of training samples, and $(T_i^g, T_i^s)$ is the $i$-th training sample containing a pair of the grid-based trajectory $T^g$ and the segment-based trajectory $T^s$. Note that the grid-based trajectory $T^g$ is transformed from the point-based trajectory $T^p$. We derive the evidence lower bound (ELBO) of each pair of trajectories as:

$$\log p_\theta\left(T^s | T^g\right) \geq \mathcal{L}_{ELBO} = \mathbb{E}_{q_\alpha(\mathbf{z}, c | T^g)} [\log \frac{p_{\beta,\theta}(T^s, \mathbf{z}, c)}{q_\alpha(\mathbf{z}, c | T^g)}] \tag{12}$$

where $\alpha$, $\beta$ and $\theta$ represent the parameters in the three components, respectively. Afterwards, the lower bound can be written as:

$$\begin{aligned}\mathcal{L}_{ELBO} =& \mathbb{E}_{q_\alpha(\mathbf{z}|T^g)}[\log p_\theta(T^s|\mathbf{z})] - \lambda \mathbb{E}_{q_\alpha(c|T^g)}[KL(q_\alpha(\mathbf{z}|T^g)||p_\beta(\mathbf{z}|c))] \\ & - \zeta \mathbb{E}_{q_\alpha(\mathbf{z}|T^g)}[KL(q_\alpha(c|\mathbf{z})||p_\beta(c)]\end{aligned} \tag{13}$$

where $KL(p, q)$ represents KL divergence between two distributions of $q$ and $p$. Thus, The objective of **L2MM** is to maximize Eq.13.

As shown in Eq.13, the first item represents the map matching accuracy, which is the probability that a latent representation $\mathbf{z}$ from the grid-based trajectory $T^g$ is decoded to generate the correct segment-based trajectory $T^s$ in the training data. The last two items are related to pattern recognition. Concretely, the second item aims to regularize the distribution of representation with the patterns. It can make two probability distributions close to each other. The former is the probability of generating a representation vector from the trajectory embedding, while the latter is the probability that the trajectory belongs to a certain pattern in the latent space. Similarly, the third item makes the probability that the latent representation belongs to a pattern close to the occurrence probability of that pattern in the spatial trajectories. $\lambda$ and $\zeta$ are weights of the two KL-divergence terms. In our experiments, we set both $\lambda$ and $\zeta$ to 0.1.

Specifically, we elaborate the 9-step process of the joint optimization. As depicted in Fig. 5, we first initialize the three components, i.e., initializing $\alpha$, $\beta$ and $\theta$. The grid-based trajectories are passed to *Trajectory Representation Learning and Enhancement* to generate trajectory representations (steps 1 and 2). The generated representations constitute the sample space. Note that as the patterns are determined (i.e., being initialized or updated), and the occurrence probability of each pattern is known, the true latent space can be determined. The trajectory representation derived from *Trajectory Representation Learning and Enhancement* should be close to the latent representation sampled from the true latent space. Meanwhile, the probability that a trajectory belongs to the pattern is proportional to the occurrence probability of a pattern. Therefore, we also sample representations from the true latent space (step 2). Then, the second and third item of our objective function are calculated to make the sample space close to the true latent space (step 3). On the other hand, the trajectory representations are fed to *Trajectory Matching and Generation* to produce mapped results (steps 4 and 5). The difference between the mapped results and the expected segment-based trajectories (i.e., the first item of our objective function) is calculated (step 6). Following the gradient back propagation, the parameters of two components, i.e., $\alpha$ and $\theta$ are updated (step 7). Steps 1~7 are circularly executed with sufficient iterations, until $\alpha$ and $\theta$ are well trained. Then, *Trajectory Representation Learning and Enhancement* can generate new trajectory representations (step 8). These new trajectory representations are further used to update the true latent space, i.e., update $\beta$ (step 9). The updated true latent space will guide the next update loop for $\alpha$ and $\theta$ (steps 1~7), until $\alpha$ and $\theta$ are well trained.

Algorithm 1 outlines the training process of **L2MM**. We employ the idea of the Expectation-Maximization (EM) algorithm [38] to optimize $\alpha$, $\beta$ and $\theta$ step by step. As illustrated in Alg.1, $\mathcal{D}$ represents the available training data, i.e., grid-based trajectories and segment-based trajectories in pairs, $\mathcal{L}$ is the objective function. $c$ represents a pattern behind trajectories. First, $\alpha$ and $\theta$ are randomly initialized. In more detail, all pattern mean vectors and standard deviation vectors in $\beta$ are set to $\mathbf{0}$ and $\mathbf{1}$, respectively in line 1. Then, **L2MM** is trained using data in $\mathcal{D}$ with the object function $\mathcal{L}$ to obtain optimal parameters $\alpha$ and $\theta$ in lines 2~4. Subsequently, part of the training data $\mathcal{D}_1$ is randomly selected to compute the corresponding trajectory representations $\mathcal{Z}$ in line 5 and 6. These trajectory representations are used to discover the optimal parameters (i.e., $\beta$) of mobility patterns through Gaussian Mixture Estimation in line 7. Based on the newest parameters

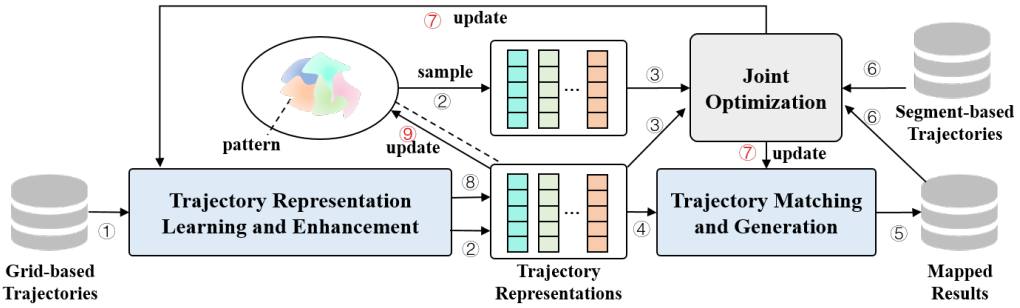



Fig. 5. The process of the joint optimization. Each step is marked using a circle with a number.

**Algorithm 1:** Joint Optimization for **L2MM**

**Input:** Training data $\mathcal{D} = (T^g, T^s)$, pattern number $K$, and objective function $\mathcal{L}$
**Output:** Model parameters $\alpha, \beta, \theta$

1 Randomly initialize $\alpha, \theta$; set $p_\beta(c)$ to $\frac{1}{K}$; set $\boldsymbol{\mu}_c$ to $\mathbf{0}$, set $\boldsymbol{\sigma}_c$ to $\mathbf{1}$, $\forall(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c) \in \beta$;
2 **for** $i \in \{1, 2, \cdots, EPOCH\}$ **do**
3  Update $\alpha$ and $\theta$ for maximizing $\mathcal{L}$ with $\beta$ and $\mathcal{D}$;
4  Stop training when criteria is met;
5 Randomly select $\mathcal{D}_1 \subset \mathcal{D}$;
6 Calculate $\mathcal{Z} = \{\boldsymbol{z} | \boldsymbol{z} = \underset{\alpha}{L2MM}(T^g), \forall T^g \in \mathcal{D}_1\}$;
7 $\beta \leftarrow$ Gaussian Mixture Estimation $(\mathcal{Z}, K)$;
8 **for** $i \in \{1, 2, \cdots, EPOCH\}$ **do**
9  Update $\alpha$ and $\theta$ for maximizing $\mathcal{L}$ with $\beta$ and $\mathcal{D}$;
10  Stop training when the criteria is met;
11 **return** $\alpha, \beta, \theta$.

of patterns, the network parameters of **L2MM** are re-updated in lines 8~10. Finally, all trained parameters are returned in line 11.

## 5 EXPERIMENTS

### 5.1 Experimental setup

*5.1.1 Dataset.* We conduct a series of experiments based on the following datasets.

**Porto dataset**[1]: It contains over 1.72 million trajectories generated by 442 taxis in the city of Porto, from July 1st 2013 to June 30th 2014. We randomly select about 0.2 million trajectories in the square area from [-8.65, 41.17] to [-8.58, 41.14] in longitude and latitude. According to statistics, the sampling time interval for over 99% of trajectories is 15s, as evidenced by Fig. 6. The average positioning error is around 7.8 meters, which is the average distance between the mapped points and the raw GPS points.

**Chongqing dataset**: It consists of trajectories generated by over 12,000 taxis in the city of Chongqing in March, 2017. We randomly select about 0.1 million trajectories in the square area from [106.47, 29.59] to [106.55, 29.62] in longitude and latitude. Statistically, about 92% of the

[1]https://www.kaggle.com/c/pkdd-15-taxi-trip-time-prediction-ii

trajectories have a sample interval of 15s, as evidenced by Fig. 6. Moreover, the average positioning error is around 26.7 meters.

**Synthetic datasets**: To validate the effectiveness of **L2MM** in addressing all low quality data issues, i.e., noise, low frequency and non-uniformness, we process the datasets to generate synthetic datasets with different characteristics. Note that the average positioning error of Chongqing dataset is higher than that of Porto dataset. Thus, we utilize the two datasets to evaluate the performance of **L2MM** under different noise levels. The low-frequency trajectory dataset and the non-uniform trajectory dataset are generated as follows.

- **Low-frequency trajectory dataset.** It is generated from the raw dataset by the uniform sampling. Specifically, given a raw trajectory with sampling time interval of 15s, we sub-sample its points at a fixed interval (e.g., 30s, 60s), with preserving the origin and destination. In our experiments, we set the sampling time interval to 30s, 60s, 90s, 120s and 135s.
- **Non-uniform trajectory dataset.** It is generated by non-uniformly sub-sampling the raw dataset of Porto, following the processing method used in [22]. Specifically, given a raw trajectory, we randomly drop points between the origin and destination with user-specified dropping rate. In our experiments, we set the dropping rate to 0.6. Fig. 6 shows the sampling time interval distribution of the non-uniform Porto dataset and the two raw datasets, where that of the non-uniform Porto dataset varies dynamically from 15s to 60s.

**Road network dataset:** We extract the corresponding road networks from the OpenStreetMap platform, where 6911 and 3027 road segments are contained in Porto and Chongqing, respectively. Furthermore, the research areas in Porto and Chongqing are divided into equal-size grid cells, with 2,242 and 2,695 grids, respectively. The default size of the grid cell is 100 meters.

**Ground-truth dataset**: To train and evaluate the deep learning-based model **L2MM**, we generate the ground truth for the trajectories in Porto and Chongqing datasets, referring to the processing method used in [9, 48]. Specifically, we employ the HMM map-matching algorithm [29] to map high-frequency trajectories (i.e., a sampling interval of 15s) to the road network. We then take the matched results as the ground truth. Note that each ground-truth dataset is divided into the training, validation, and test datasets at a ratio of 8: 1: 1.

*5.1.2 Baselines.* We compare **L2MM** with classic method called **HMM** as well as a recent deep learning based model. In addition, we also implement five variations of our proposed **L2MM**.

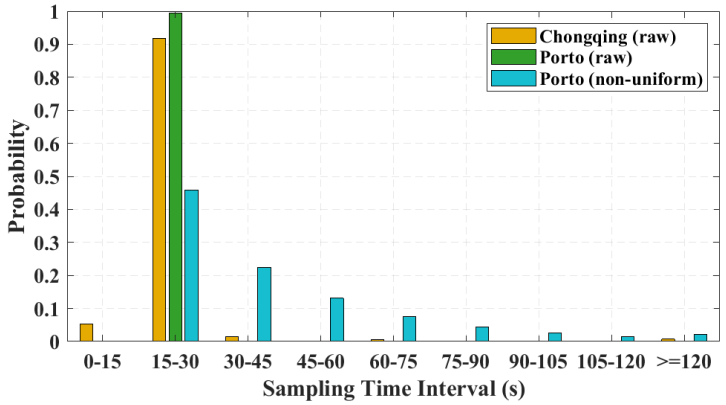



Fig. 6. Sampling time interval distribution of Porto and Chongqing datasets.

To facilitate the reproduction of our results, we have made the source code of **L2MM** publicly available[2].

(1) **HMM** [29]: It is the state-of-the-art method which maps each point-based trajectory individually onto the road network. First, it identifies candidate road segments for each point in the trajectory based on the spatial distance. Then, emission and transition probabilities for candidate segments are calculated. Finally, the best mapped results can be found by the Viterbi algorithm. As the representative of traditional methods, we evaluate its performance in dealing with low-quality trajectories.
(2) **DeepMM** [9]: It is a pioneering work that applies the seq2seq model for map matching. It designs two data augmentation methods (i.e., repeating real trajectories and simulating trajectories with the shortest path assumption) to expand the trajectory dataset. Note that they are not used in our experiments for two reasons. First, data augmentation cannot accurately reflect patterns in the real-world trajectories [10]. Second, we already have sufficient data to support the training, which can be validated in Section 5.2.
(3) **BL2MM**: A basic seq2seq model. It only contains *Trajectory Representation Learning* and *Trajectory Matching and Generation*, and is served as the base of the following variants.
(4) **BL2MM w HF**: A model combines the **BL2MM** and the trajectory representation enhancement with the high-frequency data. It utilizes the high-frequency trajectories to enhance capability of the latent representation of the low-frequency trajectories. The latent representations are fed to *Trajectory Matching and Generation* to generate the mapped results.
(5) **BL2MM w GD**: A model combines the **BL2MM** and the trajectory representation enhancement with the Gaussian distribution. It employs the second enhancement method in Section 4.1 to model the latent representation as a Gaussian distribution.
(6) **BL2MM w HF&GD**: A model combines the **BL2MM** and the two enhancement methods in Section 4.1. It utilizes high-frequency trajectories to enhance the latent representation, and regularizes the latent space by using the Gaussian mixture distribution.
(7) **BL2MM w P(K)**: A model combines the **BL2MM** and *Pattern Recognition and Representation*. It recognizes typical $K$ patterns in the latent space and incorporates them into map matching, as mentioned in Section 4.2.

*5.1.3 Metrics and Parameter Settings.* Following the previous study [9], we calculate the matching accuracy by comparing the mapped result to the ground truth. It is defined as:

$$Accuracy = \frac{len(T^{s^*} \cap T^s)}{max(len(T^{s^*}), len(T^s))} \tag{14}$$

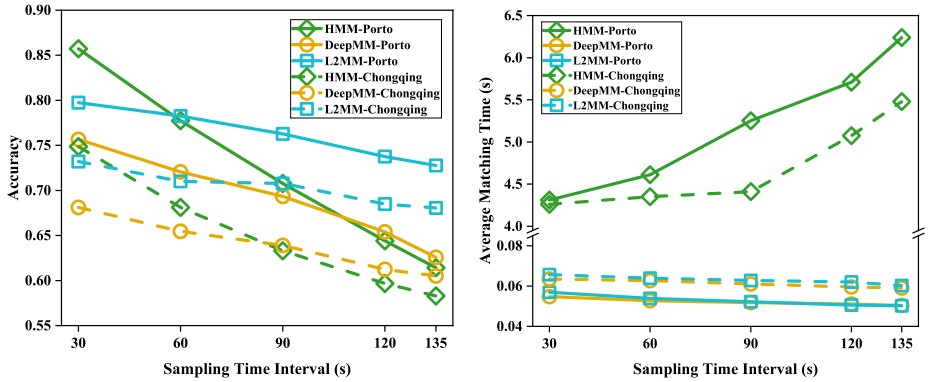
where $T^{s^*}$ and $T^s$ refer to the mapped result and the ground truth trajectory, $len()$ obtains the length of road segments, and $T^{s^*} \cap T^s$ is the correctly mapped road segments. Moreover, we compute the average matching time of each trajectory, which is defined as:

$$Average\ matching\ time = \frac{\sum_{\forall T^{s^*} \in \mathcal{M}} time(T^{s^*})}{|\mathcal{M}|} \tag{15}$$

where $time()$ calculates the matching time for a trajectory, $\mathcal{M}$ is the test trajectory dataset, and $|\mathcal{M}|$ is the total number of test trajectories.

In our implementation, we set batch size and learning rate as 128 and 0.01, respectively. The dimension of vector from cell embedding is set to 128. We use two-layer GRU in the *Trajectory Representation Learning and Enhancement*, and one-layer GUR in the *Trajectory Matching and Generation*. The dimension of hidden state is set to 128. To avoid gradient explosion, we regularize

[2]https://github.com/JiangLinLi/L2MM




(a) Accuracy under different sampling time intervals. (b) Efficiency under different sampling time intervals.

Fig. 7. The matching performance of the three models under different datasets.

RNN with a dropout rate of 0.1 and employ the Adam optimizer. Besides, the deep model of learning trajectory representation enhancement with the high frequency trajectories is pre-trained independently. Our model is programmed using Python with Pytorch. All methods are running on a server with NVIDIA GeForce RTX 2080 Ti GPUs and one AMD EPYC 7452 32-Core CPU.

### 5.2 Overall Performance of L2MM

We first compare **L2MM** with **HMM** and **DeepMM** on different datasets. The overall performance of the three methods is shown in Fig. 7.

**Accuracy**. Figure 7(a) depicts the matching accuracy of the three approaches on two datasets. As we can see, **L2MM** overall achieves the best performance when the sampling time interval goes over than 60s, compared to **HMM** and **DeepMM**. There is a sharp fall in the performance of **HMM** while our approach performs much better, especially when the sampling time interval is greater than 90s. The reason is that the distance between two consecutive GPS points could be several kilometers at low frequencies (e.g., a sampling time interval of 135s), leading to the great uncertainty of **HMM** in finding the true driving path. With the help of the deep learning model, **DeepMM** can map the sparse trajectories to the driving paths. It encodes the sparse trajectories into the latent representation and learns the map function through massive training data, resulting the better performance. However, the scant information in the sparse trajectories degrades the expressive capability of the latent representation. To this end, we use two methods to learn high-quality latent representation. Besides, we mine patterns hidden in trajectories and introduce them into map matching. The results validate the superiority of **L2MM**.

We observe that the matching accuracy of **L2MM** is slightly worse than **HMM** but much better than **DeepMM**, when the sampling time interval is below the 60s. There are two reasons. One is that **HMM** is the state-of-the-art method to do map matching for high-frequency GPS trajectories. It can achieve good performance when the sampling time interval is below the 60s. Another is that we employ the matching results of **HMM** at the sampling time interval of 15s as the ground truth to train and evaluate the proposed model. Thus, it is reasonable that **L2MM** performs slightly worse than **HMM** under the relatively high-frequency GPS trajectories ( i.e., the Chongqing dataset of the sampling time interval of 30s). As shown in Fig. 7(a), **L2MM** shows better performance compared with **HMM** and **DeepMM** on other datasets. Moreover, **L2MM** shows greater superiority in matching accuracy as the sampling time interval increases.

**Running efficiency**. Figure 7(b) shows the average matching time of the three models under different low-frequency datasets. As noted, **L2MM** and **DeepMM** show better performance compared with **HMM** on all datasets. Particularly, the average matching time of **L2MM** and **DeepMM** is about 0.06s, which is only about 1% of that of **HMM**. It verifies the high efficiency of deep learning-based map matching models. Besides, as the sampling time interval increases, **L2MM** and **DeepMM** remain stable, while the average matching time of **HMM** increases dramatically. It is because that **HMM** needs the time-costly path inference between two consecutive GPS points by searching the road network. Intuitively, the distance between two consecutive GPS points will become longer when the sampling time interval increases, which covers more road segments and results in more time consumption on inferring path for **HMM**. On the other hand, **L2MM** and **DeepMM** are based on the deep neural network, which only makes the simple forward calculation when mapping a trajectory. Thus, they are not sensitive to the sampling time interval.

Here, we discuss the time consumption of our proposed model. **L2MM** has two working stages, i.e., offline training and online inference. In the stage of offline training, **L2MM** learns the mapping function between point-based trajectories and segment-based trajectories from training data. It is time-consuming for **L2MM** to learn a valid mapping function. For example, it takes about 30 minutes to train the model when using 50,000 trajectories with a sampling time interval of 30s from the Porto dataset. In the process of online inference, it only needs to perform a simple forward calculation when dealing with the trajectories. The computational cost is greatly reduced, which is superior to those traditional models that need to search the road network.

**Robustness**. We show the robustness of **L2MM** by further analyzing the performance of each model in different datasets, as follows.

1) **L2MM** *is more robust to sparseness.* As depicted in Fig. 7(a), the accuracy of all the three models drops when the sampling time interval increases. Nevertheless, **L2MM** changes much more slightly compared with the other two models. Specifically, when the sampling time interval increases from 30s to 135s, the accuracy of **L2MM** only drops by 7%, while that of **HMM** and **DeepMM** decrease by 24% and 7.6%, respectively. This result verifies **L2MM** performs more robust to the decrease of frequency.
2) **L2MM** *is more robust to noise.* As shown in Fig. 7(a), the matching accuracy of the three models on the Porto dataset is better than that on the Chongqing dataset. It may be due to the impact of different noise levels on the two datasets. As mentioned, the trajectories in the Chongqing dataset are made up of coarse position estimation, with an average positioning error of 26.7 meters. In particular, with the sampling time interval of 30s, the accuracy of **L2MM** and **DeepMM** reduce by 6.5% and 7.5%, while that of **HMM** is 10.8%. Such results demonstrate that **L2MM** is more robust to noise at the relatively high frequency.
3) **L2MM** *is more robust to the training data size.* The performance of **L2MM** and **DeepMM** under different sizes of training data is shown in Fig. 8, using the Porto dataset with a sampling time interval of 90s. As noted, the accuracy of both models increases as more data is used. It is because more training data can help to train the model adequately, learn more effective latent representations, and identify more comprehensive patterns. Furthermore, when the training data size reaching 10,000, the accracy of **L2MM** is improved by about 18% compared with **DeepMM**. It demonstrates that **L2MM** is more robust to the training data size. Indeed, it is of great significance. With a small amount of training data, we can achieve decent performance, which improves the training efficiency to a certain extent. When increasing the size of training data from 110,000, the performance of both models keeps stable. It indicates that when the model is fully trained, it is useless to use more training data, It also shows

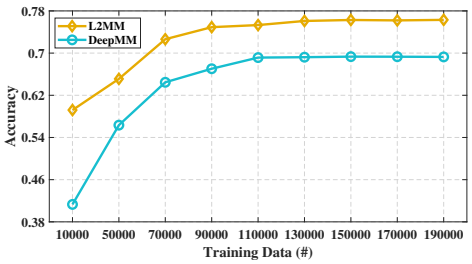



Fig. 8. Accuracy under different training data size.

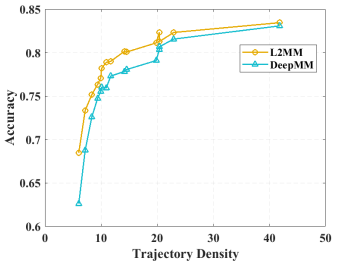



Fig. 9. Accuracy under different trajectory density.

the effect of data augmentation is limited, and explains why we do not utilize the simulated trajectories of **DeepMM**.

4) **L2MM** *is more robust to non-uniform trajectories.* We conduct experiment on the non-uniform Porto dataset. **L2MM** can achieve 78% accuracy, surpassing **DeepMM** by 4%. It confirms the superiority of **L2MM** in addressing non-uniformness.

**Reliability**. To further explore the performance of **L2MM** in some areas of the city which have less historical data, we calculate the matching accuracy of different areas, based on the Porto dataset with the sampling time interval of 30s. Specifically, we first divide the city into 15 equal-size areas and calculate the trajectory density in each region, which is defined as the ratio of the number of GPS points to the total length of the road segment in the area. We calculate the matching accuracy in different areas, and the results are shown in Fig. 9.

As we can see, both **L2MM** and **DeepMM** have poor performance when the trajectory density of the area is small (e.g., less than 8). It shows that it is difficult for both models to find the correct driving paths in these areas. We notice that **L2MM** performs much better than **DeepMM**, since it benefits from the trajectory representation enhancement and pattern recognition and incorporation. We not only enhance the capability of trajectory representations, but also improve the generalization ability of **L2MM** by exploring and incorporating patterns hidden in historical trajectories. Note that it is hard for the proposed model to identify true driving paths in areas, without any historical trajectories.

In Fig. 9, the matching accuracy increases rapidly as the trajectory density of the region increases. When continuing to increase the trajectory density (e.g., greater than 20), the performance remains basically stable. In areas with sufficient trajectory data (i.e., areas with high trajectory density), the matching accuracy of both models is quite good, and the performance of **L2MM** is slightly better. It indicates that once the deep learning-based models are fully trained, both models can achieve excellent performance. It is worth noting that the distribution of trajectories in the city is uneven, and there are many areas with low trajectory density. Therefore, it demonstrates the superiority of **L2MM**, which can achieve good performance in areas without adequate trajectory data.

## 5.3 The Impacts of Components of L2MM

*5.3.1 Effectiveness of Trajectory Representation Enhancement.* As mentioned in Section 4.1, we enhance the trajectory representation with two methods, i.e., embedding rich details of High-Frequency trajectories into latent representations of low-frequency trajectories (abbreviated as **HF**), and modeling the latent space as a Gaussian Distribution (abbreviated as **GD**). To evaluate the effectiveness of the two methods, we compare **BL2MM**, **BL2MM w HF**, **BL2MM w GD** and **BL2MM w HF&GD** under different low-frequency datasets of Porto, where the high-frequency dataset with the sampling time interval of 15s is used to enhance the trajectory representation.

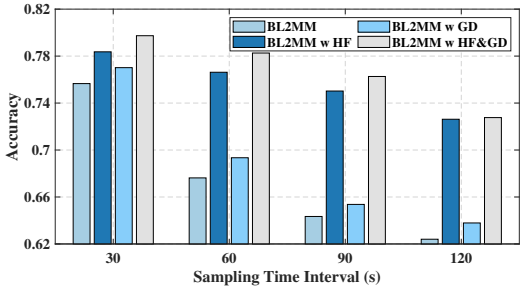



Fig. 10. Effectiveness of representation enhancement.

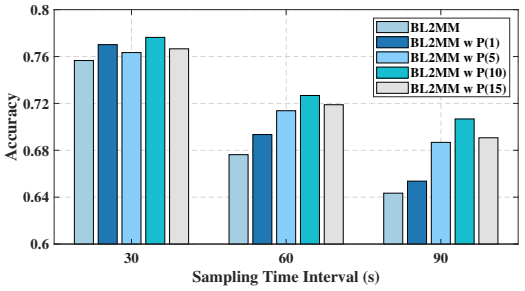



Fig. 11. Effectiveness of pattern recognition.

The accuracy of four variants of our method is shown in Fig. 10, under different sampling time intervals. We observe that **BL2MM w HF&GD** achieves the best performance on all datasets, and **BL2MM w HF** and **BL2MM w GD** perform better than **BL2MM**. The result demonstrates the effectiveness of the two representation enhancement methods. Moreover, **BL2MM w HF** outperforms **BL2MM w GD** in all scenarios, which indicates that the **HF** method plays a more important role in improving the matching accuracy. It is because that the **HF** method can fill the big information gap between low-frequency and high-frequency trajectories, greatly enriching the trajectory representation. While the **GD** method simply uses information of the adjacent trajectories in the latent space to make their spatial characteristics more similar, as a result, the improvement effect may be relatively limited. Furthermore, the accuracy improvement of **BL2MM w HF** is more significant at the low frequency, compared to that of **BL2MM**. More information about movement details is lost as the sampling time interval increases. Accordingly, the **HF** method becomes more effective in dealing with the lower frequency trajectories.

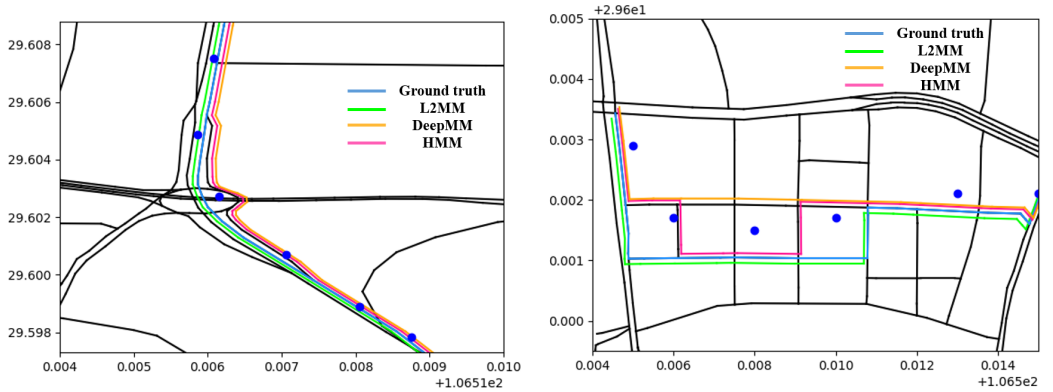
*5.3.2 Effectiveness of Pattern Recognition and Representation.* As mentioned in Section 4.2, we introduce patterns and the latent space is modeled as a Gaussian mixture distribution. To evaluate whether patterns are helpful for improving the matching accuracy, we compare **BL2MM** and **BL2MM w P(K)** under different low-frequency datasets of Porto. $K$ is set to 1, 5, 10 and 15, respectively. It is worth mentioning that **BL2MM w P(K)** degrades to **BL2MM w GD** when $K$ is set to 1.

As shown in Fig. 11, the four **BL2MM w P(K)** models outperform **BL2MM** in all scenarios, which suggests that explicitly recognizing and incorporating patterns can indeed facilitate the map matching task. Moreover, it shows that the number of patterns has an impact on the performance. The accuracy does not increase linearly with $K$. It achieves the best performance when $K$ is set to 10. It suggests that either too many or too few patterns can model the latent space well, resulting in deteriorating the map matching performance.

## 5.4 Case Study

Our proposed model shows superior overall performance in terms of accuracy and running time. To further explore the ability of **L2MM** in handling map matching under complex situations, we use two representative cases to visually show how it outperforms baseline methods, including **HMM** and **DeepMM**. Two cases are shown in Fig. 12. Note that the road network is depicted in black lines, the raw trajectory is represented by a sequence of GPS points in blue, and the matched results of the three map-matching algorithms are shown in different colors.

As shown in Fig. 12(a), the road network is very complex and crisscrossing. The sparse and noisy trajectory goes to the north from the southeast. In the beginning, there are two parallel roads. Then




(a) Matching trajectory in a complex situation.

(b) Matching trajectory on roads in parallel.

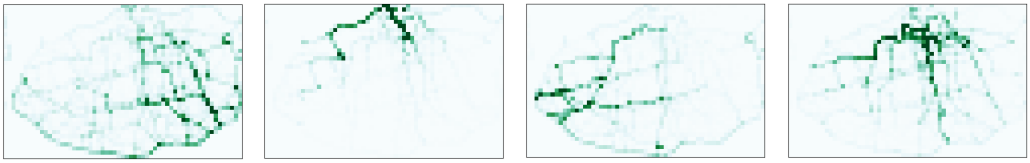
Fig. 12. Two representative case studies in Chongqing.

there is a complex intersection, where a flyover is located and having a roundabout on the ground. Owing to the low frequency and noise, it is even difficult for human beings to find the true driving path for the trajectory. As we can see, the ground truth path goes through the intersection along the flyover, and only **L2MM** successfully identifies the correct path. The matched results of **HMM** and **DeepMM** move along the main road on the ground, pass through the roundabout, and merge into the main road through the auxiliary road. The reason for the failure of **HMM** is that the spatial distances from the GPS observation points to the candidate segments play an important role in matching. Besides, the assumption of the shortest driving path between two observation points leads to the failure of subsequent matching. For **DeepMM**, it directly learns latent representations for the low-frequency and noisy trajectories, producing low-quality trajectory representations. Since the two paths are close to each other, the representations of sparse and noisy trajectories on both paths are very similar. Given such a low-quality trajectory and fine-grained similar paths, it leads to the failure of **DeepMM** in this case. In contrast, we learn a high-quality trajectory representation for the low-quality trajectory with **L2MM**, based on the corresponding high-frequency trajectory. It helps to find the correct driving path under this complex situation.

In Fig. 12(b), the road network is structured, and there are two main roads in parallel. Both **HMM** and **DeepMM** fail to find the correct driving path. As we can see, the results of **HMM** are easily affected by the GPS observation points, and it tends to choose the segment closest to the observation point as the matching result. As for **DeepMM**, it selects the parallel road above as the matching result for the middle part of the trajectory, because it is more common in the trajectory dataset. While it learns the mobility patterns from the massive historical trajectories, it does not distinguish them, nor explicitly introduce them into the map-matching task. Mobility patterns work globally in an implicit way, leading to the common road as the matching result. In **L2MM**, we not only enhance the capability of trajectory representations, but also explicitly incorporate patterns into the map matching. It can help to identify the correct driving path, even if the road is not taken as often as the other road.

### 5.5 Pattern Visualization and Understanding

To facilitate the understanding of patterns, we visualize four typical patterns learned from **L2MM**, using 100,000 trajectories from the Porto dataset. Note that we draw the heatmap based on the grid-based trajectories. A grid with a darker green means a higher density of trajectories. As shown in Fig. 13, four patterns are totally different from each other. We introduce the detail as follows.



(a) Pattern #1. (b) Pattern #2. (c) Pattern #5. (d) Pattern #10.

Fig. 13. Visualization of typical patterns learned from Porto data.

Specifically, Pattern #1 corresponds the basic skeleton of the road network in Porto, including highways and main roads. There are lots of trajectories in the southeast area, where the train station is located. We notice that there are several common routes to the train station, highlighted in dark green. While the trajectories from Pattern #2 start from an area in the top area and stretch down. It is worth noting that there are two routes in dark green, which are two main streets to a popular garden, i.e., R. de Vale Formoso and R. de Monsanto, respectively. It shows the preference of certain user groups.

Most of the trajectories from Pattern #5 are crowded in the west and trend from west to east. However, the trajectories of Pattern #10 are mainly crowded in the central district, with a hotspot region and stretching around. In the hotspot region, there are lots of Points of Interest (POIs), such as hypermarkets, hospitals, a soccer field, restaurants, a catholic church, bus stops and subway stations. It reveals the mobility pattern of users to the central district. In summary, **L2MM** is able to recognize patterns with different semantic meanings from massive trajectory data. It also verifies that mobility patterns of users and the layout of the road network are helpful for map matching.

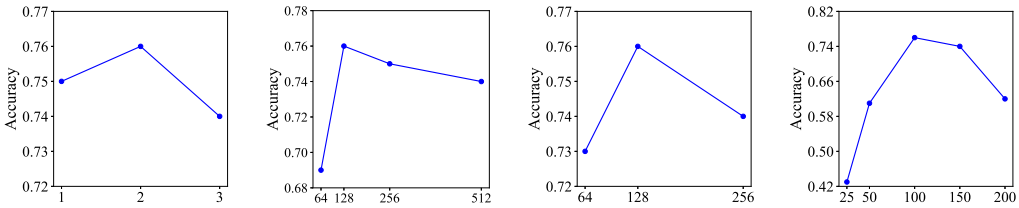
### 5.6 Hyper-Parameter Study

We investigate the impact of some key hyper-parameters on matching accuracy, based on the Porto dataset with the sampling time interval of 90s. The results are shown in Fig. 14. We can observe that **L2MM** achieves the highest accuracy when the number of the GRU layers, hidden state size, and batch size are set to 2, 128, and 128, respectively. The reason is that the model with fewer GRU layers may fail to capture key information of the input trajectory data while too many layers will increase the risk of overfitting to reduce the matching accuracy. For similar reasons, both the hidden state size and the batch size should be set appropriately.

Another important parameter in our work is the size of the grid, which is used for partitioning the city. Considering that it may affect the performance of map matching, we evaluate the accuracy of **L2MM** with different settings of grid size. As shown in Fig. 14(d), the optimal size of the grid size in our work is around 100m. With a smaller grid size setting, the matching performance of **L2MM** is very poor, due to positioning error and noises in trajectories. The matching accuracy increases with the grid size increasing from 25m to 100m. However, as we continue to increase the grid size, the performance degrades significantly. The reason could be that different raw point-based trajectories are mapped to the same grid-based trajectory when we partition the city with a large grid size. It is difficult for **L2MM** to learn an effective mapping function to generate correct fine-grained segment-based trajectories, based on the coarse-grained trajectories.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a deep learning-based model **L2MM** for map matching, aiming at effectively and efficiently mapping the low-quality trajectory data. Solving the map matching problem for such low-quality GPS trajectories has a direct and far-reaching significance for many applications. It employs multiple deep models to learn a mapping function from trajectories to




(a) Layers of GRU. (b) Hidden size. (c) Batch size. (d) Grid size.

Fig. 14. Hyper-parameters study on crucial parameters in **L2MM**.

the corresponding paths and follows the basic working flow, i.e., converting the trajectory to a representation vector, followed by decoding the path from the vector. Different from the existing learning-based map matching model, **L2MM** learns the high-quality representations with two methods to better map the low-quality trajectories and be more robust to the training data size. Moreover, **L2MM** explicitly recognizes the typical mobility patterns in the latent space for exploring more general heuristic hints and wisely incorporates the recognized patterns into the map matching task for utilizing these hints to facilitate map matching. We conduct extensive experiments based on a range of datasets, and results not only demonstrate the superiority of **L2MM** in terms of accuracy and efficiency, but also validate the significance of the high-quality trajectory representations and mobility patterns.

In the future, we plan to broaden this work in the following directions. First, we intend to explore more potential valuable factors (e.g., the destination, the real-time traffic) to further improve the performance of map matching. Second, we would like to explore the spatial-aware loss function to alleviate the problem of the topology continuity of the mapped paths. Third, we are going to explore the model's transferability among different cities (i.e., from a city with high-quality data to the other one with low-quality data, or from a city with massive data to the other one with limited amount of data). Last but not least, we also plan to convert the proposed model into the real-time or the online version, by employing a sliding window mechanism [3, 5]. What is more, how the window size impacts the performance of the upgraded model, and what kinds of supported real-time applications are under-explored.

## REFERENCES

[1] Wentao Bian, Ge Cui, and Xin Wang. 2020. A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories. *Sensors* 20, 7 (2020).

[2] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. 2020. A survey on map-matching algorithms. In *Australasian Database Conference*. Springer, 121–133.

[3] Chao Chen, Yan Ding, Xuefeng Xie, Shu Zhang, Zhu Wang, and Liang Feng. 2020. TrajCompressor: An Online Map-matching-based Trajectory Compression Framework Leveraging Vehicle Heading Direction and Change. *IEEE Transactions on Intelligent Transportation Systems* 21, 5 (2020), 2012–2028.

[4] Chao Chen, Qiang Liu, Xingchen Wang, Chengwu Liao, and Daqing Zhang. 2021. semi-Traj2Graph: Identifying Fine-grained Driving Style with GPS Trajectory Data via Multi-task Learning. *IEEE Transactions on Big Data* (2021), 1–15.

[5] Chao Chen, Daqing Zhang, Yasha Wang, and Hongyu Huang. 2021. *Enabling Smart Urban Services with GPS Trajectory Data.* Springer.

[6] Ge Cui, Wentao Bian, and Xin Wang. 2021. Hidden Markov map matching based on trajectory segmentation with heading homogeneity. *GeoInformatica* 25, 1 (2021), 179–206.

[7] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *CoRR*

abs/1611.02648 (2016).
[8] Marko Dogramadzi and Aftab Khan. 2021. Accelerated Map Matching for GPS Trajectories. *IEEE Transactions on Intelligent Transportation Systems* (2021), 1–10.
[9] Jie Feng, Yong Li, Kai Zhao, Zhao Xu, Tong Xia, Jinglin Zhang, and Depeng Jin. 2020. DeepMM: Deep Learning Based Map Matching with Data Augmentation. *IEEE Transactions on Mobile Computing* (2020), 1–13.
[10] Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to Simulate Human Mobility. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3426–3433.
[11] Qiang Gao, Fan Zhou, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2019. Predicting Human Mobility via Variational Attention. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2750–2756.
[12] John A Hartigan. 2012. *Bayes theory*. Springer Science & Business Media.
[13] Mahdi Hashemi. 2017. Reusability of the Output of Map-Matching Algorithms Across Space and Time Through Machine Learning. *IEEE Transactions on Intelligent Transportation Systems* 18, 11 (2017), 3017–3026.
[14] Mahdi Hashemi and Hassan A Karimi. 2014. A critical review of real-time map-matching algorithms: Current issues and future directions. *Computers, Environment and Urban Systems* 48 (2014), 153–165.
[15] Yu-Ling Hsueh and Ho-Chian Chen. 2018. Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions. *Information Sciences* 433-434 (2018), 55–69.
[16] Gang Hu, Jie Shao, Fenglin Liu, Yuan Wang, and Heng Tao Shen. 2016. If-matching: Towards accurate map-matching with information fusion. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 114–127.
[17] Yukun Huang, Weixiong Rao, Zhiqiang Zhang, Peng Zhao, Mingxuan Yuan, and Jia Zeng. 2018. Frequent pattern-based map-matching on low sampling rate trajectories. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. 266–273.
[18] George R. Jagadeesh and Thambipillai Srikanthan. 2017. Online Map-Matching of Noisy and Sparse Location Data With Hidden Markov and Route Choice Models. *IEEE Transactions on Intelligent Transportation Systems* 18, 9 (2017), 2423–2434.
[19] Xiang Jiang, Erico N. de Souza, Ahmad Pesaranghader, Baifan Hu, Daniel L. Silver, and Stan Matwin. 2017. TrajectoryNet: An Embedded GPS Trajectory Representation for Point-based Classification Using Recurrent Neural Networks. *CoRR* abs/1705.02636 (2017).
[20] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114 (2013).
[21] Matej Kubicka, Arben Cela, Hugues Mounier, and Silviu-Iulian Niculescu. 2018. Comparative study and application-oriented classification of vehicular map-matching methods. *IEEE Intelligent Transportation Systems Magazine* 10, 2 (2018), 150–166.
[22] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE '18)*. 617–628.
[23] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*.
[24] Xiliang Liu, Kang Liu, Mingxiao Li, and Feng Lu. 2016. A ST-CRF map-matching method for low-frequency floating car data. *IEEE Transactions on Intelligent Transportation Systems* 18, 5 (2016), 1241–1254.
[25] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. 2020. Online Anomalous Trajectory Detection with Deep Generative Sequence Modeling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE '20)*. 949–960.
[26] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 352–361.
[27] Reham Mohamed, Heba Aly, and Moustafa Youssef. 2017. Accurate Real-time Map Matching for Challenging Environments. *IEEE Transactions on Intelligent Transportation Systems* 18, 4 (2017), 847–857.
[28] James Murphy, Yuanyuan Pao, and Albert Yuen. 2019. Map Matching When the Map is Wrong: Efficient on/off Road Vehicle Tracking and Map Learning. In *Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS'19)*. Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages.
[29] Paul Newson and John Krumm. 2009. Hidden Markov Map Matching through Noise and Sparseness *(GIS '09)*. Association for Computing Machinery, New York, NY, USA, 336–343.
[30] Takayuki Osogami and Rudy Raymond. 2013. Map Matching with Inverse Reinforcement Learning.. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI '13)*. 2547–2553.
[31] Mohammed Quddus and Simon Washington. 2015. Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies* 55 (2015), 328–339.

[32] Mohammed A Quddus, Washington Y Ochieng, and Robert B Noland. 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation research part c: Emerging technologies* 15, 5 (2007), 312–328.

[33] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. 2020. DMM: Fast Map Matching for Cellular Data. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 60, 14 pages.

[34] Mudhakar Srivatsa, Raghu Ganti, Jingjing Wang, and Vinay Kolar. 2013. Map Matching: Facts and Myths. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13)*. Association for Computing Machinery, New York, NY, USA, 484–487.

[35] Guanfeng Wang and Roger Zimmermann. 2014. Eddy: An Error-Bounded Delay-Bounded Real-Time Map Matching Algorithm Using HMM and Online Viterbi Decoder. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '14)*. Association for Computing Machinery, New York, NY, USA, 33–42.

[36] Wei Wang, Feng Xia, Hansong Nie, Zhikui Chen, Zhiguo Gong, Xiangjie Kong, and Wei Wei. 2021. Vehicle Trajectory Clustering Based on Dynamic Representation Learning of Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 22, 6 (2021), 3567–3576.

[37] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 25–34.

[38] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. 2008. Top 10 algorithms in data mining. *Knowledge and information systems* 14, 1 (2008), 1–37.

[39] Hao Xu, Hongchao Liu, Chin-Woo Tan, and Yuanlu Bao. 2010. Development and Application of an Enhanced Kalman Filter and Global Positioning System Error-Correction Approach for Improved Map-Matching. *Journal of Intelligent Transportation Systems* 14, 1 (2010), 27–36.

[40] Can Yang and Győző Gidófalvi. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science* 32, 3 (2018), 547–570.

[41] Dakai Yang, Baigen Cai, and Yifang Yuan. 2003. An improved map-matching algorithm used in vehicle navigation system. In *Proceedings of the 2003 IEEE international conference on intelligent transportation systems*, Vol. 2. IEEE, 1246–1250.

[42] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE '21)*. 2183–2188.

[43] Yifang Yin, Rajiv Ratn Shah, Guanfeng Wang, and Roger Zimmermann. 2018. Feature-based map matching for low-sampling-rate GPS trajectories. *ACM Transactions on Spatial Algorithms and Systems* 4, 2 (2018), 1–24.

[44] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Computational Intelligence Magazine* 13, 3 (2018), 55–75.

[45] James J. Q. Yu. 2021. Travel Mode Identification With GPS Trajectories Using Wavelet Transform and Deep Learning. *IEEE Transactions on Intelligent Transportation Systems* 22, 2 (2021), 1093–1103.

[46] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering* 25, 1 (2011), 220–232.

[47] Ethan Zhang and Neda Masoud. 2021. Increasing GPS Localization Accuracy With Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* 22, 5 (2021), 2615–2626.

[48] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *2012 IEEE 28th international conference on data engineering*. IEEE, 1144–1155.

[49] Fan Zhou, Xiaoli Yue, Goce Trajcevski, Ting Zhong, and Kunpeng Zhang. 2019. Context-Aware Variational Trajectory Encoding and Human Mobility Inference. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3469–3475.

[50] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *International Conference on Learning Representations (ICLR '18)*.