MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning

Huimin Ren[1,3,4], Sijie Ruan[2,3,4], Yanhua Li[1,*], Jie Bao[3,4],
Chuishi Meng[3,4], Ruiyuan Li[3,4], Yu Zheng[2,3,4]
[1]Worcester Polytechnic Institute, MA, USA [2]Xidian University, Shaanxi, China
[3]JD iCity, JD Technology, Beijing, China [4]JD Intelligent Cities Research
hren@wpi.edu;sjruan@stu.xidian.edu.cn;yli15@wpi.edu;
{baojie,meng.chuishi}@jd.com;liruiyuan@whu.edu.cn;msyuzheng@outlook.com

ABSTRACT

With the increasing adoption of GPS modules, there are a wide range of urban applications based on trajectory data analysis, such as vehicle navigation, travel time estimation, and driver behavior analysis. The effectiveness of urban applications relies greatly on the high sampling rates of trajectories precisely matched to the map. However, a large number of trajectories are collected under a low sampling rate in real-world practice, due to certain communication loss and energy constraints. To enhance the trajectory data and support the urban applications more effectively, many trajectory recovery methods are proposed to infer the trajectories in free space. In addition, the recovered trajectory still needs to be mapped to the road network, before it can be used in the applications. However, the two-stage pipeline, which first infers high-sampling-rate trajectories and then performs the map matching, is inaccurate and inefficient. In this paper, we propose a Map-constrained Trajectory Recovery framework, MTrajRec, to recover the fine-grained points in trajectories and map match them on the road network in an end-to-end manner. MTrajRec implements a multi-task sequence-to-sequence learning architecture to predict road segment and moving ratio simultaneously. *Constraint mask*, *attention mechanism*, and *attribute module* are proposed to overcome the limits of coarse grid representation and improve the performance. Extensive experiments based on large-scale real-world trajectory data confirm the effectiveness and efficiency of our approach.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**.

KEYWORDS

Trajectory recovery; Road network; Multi-task learning; Sequence-to-sequence model

*Yanhua Li is corresponding author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467238

ACM Reference Format:
Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, Yu Zheng. 2021. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467238

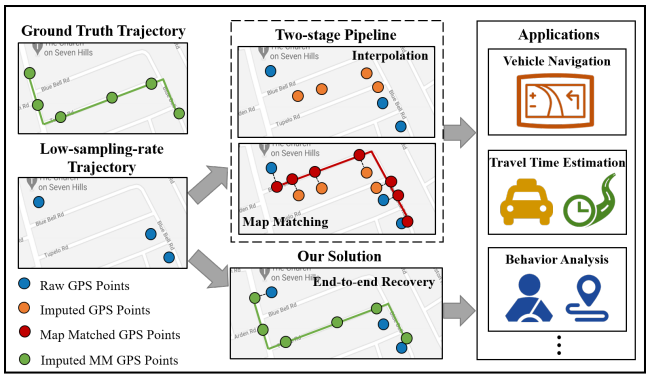


Figure 1: Intuition of Trajectory Recovery Problem

1 INTRODUCTION

Nowadays, GPS modules have been widely used throughout all kinds of mobile devices, and the generated trajectory data have empowered many applications, such as vehicle navigation [11], travel time estimation [6], driver behavior analysis [5]. The effectiveness of these applications relies on the sampling rate of trajectories, since low-sampling-rate trajectories lose detailed information for moving objects and increase the uncertainty between two consecutive sampled locations. However, in reality, there is a large quantity of low-sampling-rate GPS trajectory data. For instance, taxis usually report GPS locations every 2 ∼ 6 minutes to reduce the energy consumption of communication [31].

In order to better utilize these low-sampling-rate trajectories, many inference methods have been proposed to recover low-sampling-rate trajectories. One straight forward solution is to assume that vehicles are moving with the uniform speeds [10]. However, the dynamic behavior mobility pattern cannot be captured in this way. To tackle this challenge, many deep learning based models have been proposed [28–30]. For example, Wang et al. [28] recovered a high-sampling-rate trajectory from a low-sampling-rate one with DHTR, which predicts the coarse-grained grid of high-sampling-rate point by integrating a sequence-to-sequence model with a calibration component of Kalman Filter. However, after the trajectory recovery

process, there is still a map matching [20] task to be done, before the trajectory data can be used by the applications. The map matching task aligns GPS points in trajectories with the road network, and is a fundamental pre-processing step. It not only empowers the road-based applications, e.g., vehicle navigation [11], travel time estimation [6], but also enriches trajectories with more semantic meanings to benefit driver-based applications, e.g., behavior analysis [5].

Traditional methods solve the map-constrained trajectory recovery problem through a two-stage pipeline, which first recovers low-sampling-rate trajectories and then implements a map matching algorithm to project trajectories onto the road network. Although we can perform map matching based on the recovered trajectory as shown in Fig. 1, the inference error can be accumulated. In addition, the two-stage pipeline is also inefficient because map matching algorithms are time-consuming [18, 20, 31]. Based on these observations, a natural question arises: can we recover a high-sampling-rate trajectory based on a low-sampling-rate one and perform map matching on it simultaneously? The end-to-end solution is expected to reduce the inference error as shown in the bottom part of Fig. 1, and improve the efficiency.

Luckily, with the renaissance of neural networks, the deep learning technique provides a promising computational framework to solve complicated tasks, which gives us an opportunity to tackle the challenges in an end-to-end fashion. To our best knowledge, this work is the first attempt to recover low-sampling-rate trajectories and map match them onto the road network simultaneously. Specifically, the map-constrained trajectory recovery problem is challenging due to the following reasons:

(1) *Map constraints.* Previous work [28–30] focus on trajectory recovery in the free space. It is difficult for a deep learning model to generate road network constrained coordinates.
(2) *Coarse grid representations.* Converting the numerical coordinates to discrete units is a common preprocessing strategy for deep-learning-based trajectory modeling [7, 21, 23, 28, 30], since it can reduce the computational complexity of unconstrained numerical coordinates. However, using discrete units is likely to introduce noises or inaccurate information into the model, which incurs challenges in the fine-grained trajectory recovery problem.
(3) *Diverse complex factors.* The recovery accuracy is influenced by traffic conditions, as vehicles are not moving at constant speeds in the real world. The traffic conditions are determined by many complex factors, such as the spatial context, temporal dependencies, and weather conditions [27]. In such scenario, only using the low-sampling-rate trajectory data is not be sufficient to recover the missing points accurately.

To tackle these challenges, in this paper, we propose a novel Map-constrained Trajectory Recovery model, i.e., MTrajRec, which is based on the sequence-to-sequence (Seq2Seq) multi-task learning. MTrajRec recoveries the trajectories by interpolating the missing points on the road network. First, to guarantee the recovered trajectories constrained on the road network, we introduce a multi-task learning into the classic Seq2Seq generation framework by predicting road segment IDs and moving ratios simultaneously. In order to deal with coarse grid representation, we design a constraint mask layer to extract the fine-grained information. Finally, since the traffic is affected by complex factors, we employ an attribute module to capture the external influences. Overall, our main contributions can be summarized as follows:

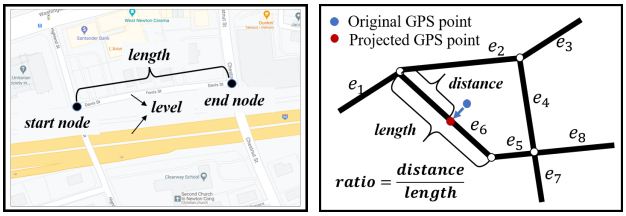
- We present the first attempt to solve the map-constrained trajectory recovery problem via Seq2Seq multi-task learning.
- We devise a novel MTrajRec model, which can recover the trajectory and map match it onto the road network simultaneously. We utilize *constraint mask, attention mechanism* and *attribute module* to improve the performance.
- We conduct substantial experiments using a real-world taxi trajectory dataset to evaluate the effectiveness and efficiency of our proposed MTrajRec.

2 OVERVIEW

2.1 Preliminaries

Definition 1. Trajectory. A trajectory τ can be defined as a sequence of GPS positions with timestamps, i.e, $\tau = \langle p_1, p_2, \cdots, p_n \rangle$, where $p_i = \langle lat, lng, t \rangle, \forall i, 1 \leq i \leq n$, which captures the latitude and longitude of the GPS position at timestamp t.

Definition 2. Road Network. A road network is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_K\}$ is a set of nodes representing intersections of road segments, and $\mathcal{E} = \{e_1, e_2, \cdots, e_L\}$ refers a set of edges representing the road segments which connect nodes v in $\mathcal{V}$. For each $e \in \mathcal{E}$, it contains three properties: 1) start and end nodes, indicating the start and end GPS position of a road segment; 2) length, which refers to the distance of a road segment in meter; 3) level, which indicates the type of road, such as highway, street, etc., with different colors shown in Fig. 2(a).



(a) Road Segments Properties (b) Moving Ratio r

Figure 2: Examples of Preliminary Concepts

Definition 3. Map Matching. Due to GPS device measurement errors, the GPS data is not precise. Map matching is a procedure to convert a sequence of raw latitude/longitude coordinates, so that the raw GPS points will be projected onto the road network.

Definition 4. Map-matched Trajectory Point. A map-matched trajectory point is denoted as $a = \langle e, r, t \rangle$, where e is the road segment ID, r is the moving ratio, which represents the ratio of moving distance over the length of the road segment, and t is the timestamp. Fig.2(b) gives a detailed explanation of r. With the road segment ID e and the moving ratio r, we can uniquely represent a location on the road network. The convert function is formulated as follows:

$$\begin{aligned} p.lat &= a.e.start.lat + a.r * (a.e.end.lat - a.e.start.lat) \\ p.lng &= a.e.start.lng + a.r * (a.e.end.lng - a.e.start.lng) \\ p.t &= a.t \end{aligned} \tag{1}$$

Definition 5. Sampling Rate. A sampling rate ϵ is the time difference between two consecutive sampled points for a trajectory, which usually depends on device settings.

Definition 6. Map-matched ϵ-Sampling Rate Trajectory. A map-matched trajectory $\tilde{\tau}$ with ϵ-sampling rate is a sequence of map-matched trajectory points, i.e, $\tilde{\tau} = \langle a_1, a_2, \cdots, a_m \rangle$, where $a_j = \langle e, r, t \rangle, \forall j, 1 \leq j \leq m$ and $a_{j+1}.t - a_j.t = \epsilon$. For simplicity, we name $\tilde{\tau}$ as ϵ-MM trajectory.

Note that although an ϵ-MM trajectory $\tilde{\tau}$ is uniformly sampled, points in a trajectory τ may not be uniformly distributed in timestamps, which is more challenging than uniformly distributed. Besides, the ϵ-MM trajectory $\tilde{\tau}$ is map matched on the road network, while the trajectory τ is not perfectly constrained on the road network due to GPS noises. Usually, points in trajectory τ are collected with a low sampling rate.

2.2 Problem Definition

Given a low-sampling-rate trajectory $\tau = \langle p_1, p_2, \cdots, p_n \rangle$ and a target sampling rate ϵ, we aim to recover the real map-matched ϵ-sampling-rate trajectory $\tilde{\tau} = \langle a_1, a_2, \cdots, a_m \rangle$. This is to say, for each low-sampling-rate trajectory, we will infer its missing points and map match it onto the road network simultaneously.

3 METHODOLOGY

To solve the trajectory recovery problem, we are inspired by the classic Seq2Seq model [25], since our trajectory recovery problem is similar to the machine translation problem, where low-sampling-rate trajectories can be treated as English sentences and ϵ-MM trajectories can be treated as the same sentences in French. Thus, Seq2Seq structure could be potentially used to solve the trajectory recovery problem. However, the challenges of our problem mentioned in Sec. 1 prevent the Seq2Seq model from solving it, since the Seq2Seq model can generate locations step by step but cannot guarantee the generated trajectories are constrained on the road network in an end-to-end manner. Hence, we propose a novel model – MTrajRec, which can recover missing points and map match them onto the road network simultaneously.

In this section, we first introduce the basic structure of a multi-task Seq2Seq model to tackle challenge (1) *map constraint* (see Sec. 3.1), and then present three enhancement components of our MTrajRec: a constraint mask layer, which deals with challenge (2) *coarse point representation* (see Sec. 3.2), an attention mechanism, which learns global correlations (see Sec. 3.3), and an attribute module to solve challenge (3) *diverse complex factors* (see Sec. 3.4). In the following, we elaborate them in details.

3.1 Multi-task Seq2Seq Structure

As illustrated in Fig. 3, MTrajRec is composed of an encoder and a decoder. The encoder learns sequential dependencies for low-sampling-rate trajectories, while the decoder predicts road segment ID e and moving ratio r iteratively, using the previous output as the input vector.

Encoder. The low-sampling-rate trajectory $\tau = \langle p_1, p_2, ..., p_n \rangle$ is encoded into a single vector to capture the spatial and temporal dependencies of τ. We refer the single vector as a context vector. Instead of directly using coordinates, we convert the GPS locations

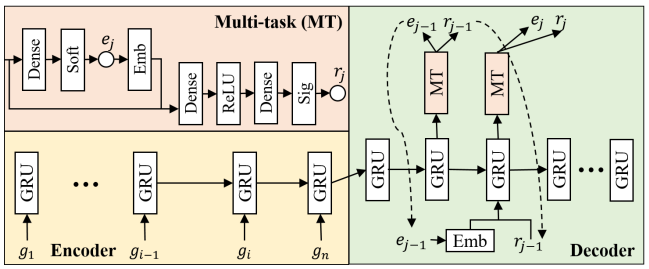


Figure 3: Structure of Basic MTrajRec

into discrete units. As [32] mentioned, it is more reliable and easy to model ID sequence than the original numerical sequence. Each numerical coordinate p_i can be converted to a unit, which is described as $g_i = \langle x_i, y_i, tid_i \rangle, \forall i, 1 \leq i \leq n$, where x_i and y_i represent i-th grid cell and $tid_i = \lfloor \frac{t_i - t_0}{\epsilon} \rfloor$ is the index of the points in the target ϵ-MM trajectory. Here, ϵ is the target sampling rate and t_i is the timestamp of i-th point in the low-sampling-rate trajectory τ. The reason that we extract tid is to help the model learn how many points should be interpolated between two consecutive points. The low-sampling-rate trajectory τ is updated to $\tau' = \langle g_1, g_2, ..., g_n \rangle$. As can be seen in Fig. 4(a), trajectory $\tau = \langle p_1, p_2, p_3 \rangle$ is updated to $\tau' = \langle g_1, g_2, g_3 \rangle$ with $tid = \langle 1, 3, 7 \rangle$. Note that tid is not continuous because the low-sampling-rate trajectory is not uniformly sampled.

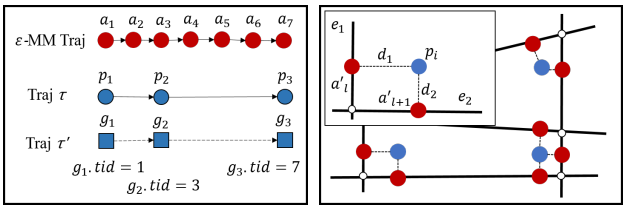
Since the trajectory data have sequential dependencies, we adopt the Gated Recurrent Unit (GRU) [3] as the encoder to obtain the context vector of low-sampling-rate trajectory. Such context vector will be used as the initial hidden state for the decoder. GRU is a variant of Long Short-term Memory networks (LSTM), which is capable of learning long-term dependencies for sequential data without performance decay. GRU sequentially updates a hidden-state by introducing an update gate $\mathbf{z}$ and a reset gate $\mathbf{r}$ to control the flow of information through the time steps. At each time step $i \in \{1, 2, \cdots, n\}$, the hidden-state vector $\mathbf{s}_i$ is:

$$\begin{aligned} \mathbf{z}_i &= \sigma(\mathbf{W}_\mathbf{z} \cdot [\mathbf{s}_{i-1}, \mathbf{g}_i] + \mathbf{b}_\mathbf{z}) \\ \mathbf{r}_i &= \sigma(\mathbf{W}_\mathbf{r} \cdot [\mathbf{s}_{i-1}, \mathbf{g}_i] + \mathbf{b}_\mathbf{r}) \\ \tilde{\mathbf{s}}_i &= \tanh(\mathbf{W}_\mathbf{s} \cdot [\mathbf{r}_i * \mathbf{s}_{i-1}, \mathbf{g}_i] + \mathbf{b}_\mathbf{s}) \\ \mathbf{s}_i &= (1 - \mathbf{z}_i) * \mathbf{s}_{i-1} + \mathbf{z}_i * \tilde{\mathbf{s}}_i, \end{aligned} \tag{2}$$

where $\mathbf{W}_\mathbf{x}$ represents the weights for the respective gate(x) neurons and $\mathbf{b}_\mathbf{x}$ is the bias for the respective gate(x). To simplify, for getting the context vector of low-sampling-rate trajectory, the encoder derives the hidden state $\mathbf{s}_i$ as:

$$\mathbf{s}_i = GRU(\mathbf{s}_{i-1}, \mathbf{g}_{i-1}), \tag{3}$$

where the last state $\mathbf{s}_n$ will be considered as the context vector as well as the initial hidden state for the decoder.



(a) Computation of tid (b) Constraint Distance

Figure 4: Illustration of concepts

Decoder. The decoder is used to recover the low-sampling-rate trajectory to the ϵ-MM trajectory $\tilde{\tau} = \langle a_1, a_2, \cdots, a_m \rangle$. As we mentioned before, the standard Seq2Seq model can predict the numerical coordinates but cannot guarantee the trajectories being map matched onto the road network. To tackle this challenge, instead of directly predicting the coordinate, we propose to predict road segments ID e and moving ratio r to make sure the predicted location must be constrained on the road network. We leverage multi-task learning [2], which solves those two tasks at the same time by sharing parameters across different tasks, since those two tasks are correlated.The input of the decoder is the concatenation of the embedding of the road segment ID e vector and moving ratio r. This is because road segment ID e is a categorical value which cannot directly be fed into the neural network. Comparing with the one-hot encoding [8], the embedding method can effectively reduce the input dimension and learn the semantic meanings for road segment ID e.

The top left in Fig. 3 demonstrates detailed information of the multi-task block, which predicts road segment ID e_j and moving ratio r_j simultaneously. After applying the GRU layer to get the output vector, we first use a dense layer with soft-max function to predict the road segment ID e_j. Then, the embedding of predicted e_j and the output vector from GRU layer are concatenated to go through the fully connected layers with a sigmoid function to predict moving ratio r_j. Since the moving ratio r_j highly depends on the related road segment ID e_j, we design a "series connection" mechanism to predict r_j. Note that although r_j depends on e_j, e_j and r_j have an influence on e_{j+1} and r_{j+1}. Thus, both road segment ID e_j and moving ratio r_j will be used as the input of the decoder.

To this end, for generating a missing point, our decoder derives the hidden state $\mathbf{h}_j$ as:

$$\mathbf{h}_j = GRU(\mathbf{h}_{j-1}, e_{j-1}, r_{j-1}). \tag{4}$$

Once we obtain the the hidden state $\mathbf{h}_j$ from the decoder, we further apply multi-task block to predict road segment ID e_j with a softmax function and to infer moving ratio r_j with a sigmoid function.

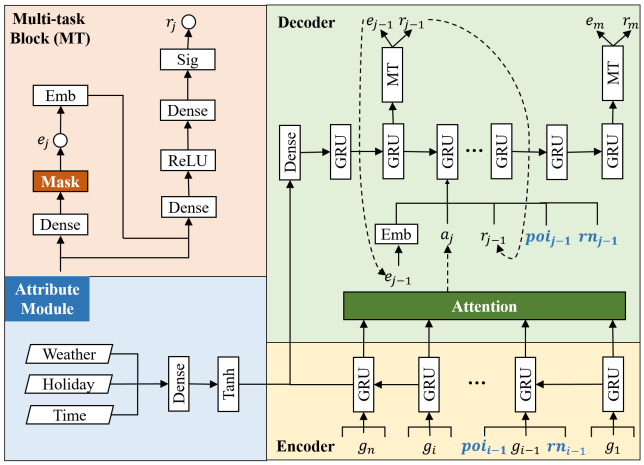


Figure 5: Structure of MTrajRec

The structure mentioned above can guarantee the recovered trajectories constraint on the road network. However, challenges of inaccurate discrete units and diverse complex factors affect the effectiveness of the trajectory recovery. In the following sections, we introduce three components to improve the performance: constraint mask layer, attention mechanism and attribute module. The overall structure of MTrajRec is illustrated in Fig. 5.

3.2 Constraint Mask Layer

As we mentioned in challenge (2), most of the previous studies [7, 24] convert the numerical coordinates into discrete units to mining trajectories since it can reduce training complexity. However, mapping the GPS coordinates into grid cells makes the precise information lost, which brings difficulties to the fine-grained MM trajectory recovery. Thus, there is a trade-off between complexity and accuracy by using discrete units or not. To tackle this challenge, we devise a *constraint mask layer* [20, 31]. With the discrete units in the encoder and the constraint mask layer in the multi-task block, we can benefit both from less complexity and higher accuracy.

In our model, we first define a *distance weight function* f [31], which represents the influence of point p_i to candidate map matched point a'_ℓ based on the Euclidean distance between them, where p_i is the i-th sample from low-sampling-rate trajectory τ and a'_ℓ is the map-matched point on road segment ID e_ℓ. Note that such distance is caused by the sensing errors of GPS and theoretically p_i can be projected onto any road segment in the road network. As shown in Fig. 4(b), the blue points indicate the original GPS points in low-sampling-rate trajectory τ, while the red points are generated by projecting the original points to road segments. Let $d_{i,\ell}$ be the Euclidean distance of raw GPS point p_i and candidate point a'_ℓ, then $f(d_{i,\ell})$ denotes the impact of distance $d_{i,\ell}$ on the original point p_i. Following [31], we use an exponential function to capture the influence of distance, i.e., $f(d) = e^{-\frac{d^2}{\beta^2}}$, where β is a parameter with respect to the road network ($\beta = 15$ in our work). In practice, we only consider matching the blue points that are not far away from road segments [20]. We set 0 probability of any distance from a road segment that is more than 50 meters away from p_i. Note that we only calculate the distance weight for points existing in the low-sampling-rate trajectories, so that we set 1 probability of all road segments for missing values. Finally we combine such function with softmax in multi-task block as the *constraint mask layer* which is formally defined as:

$$\mathbf{c}_{j,\ell} = \begin{cases} f(d_{j,\ell}), & \text{if } j \in \{g_i.tid | i \in [1,n]\} \text{ and } d_{j,\ell} < 50 \\ 0, & \text{if } j \in \{g_i.tid | i \in [1,n]\} \text{ and } d_{i,\ell} \geq 50 \\ 1, & \text{otherwise} \end{cases},$$

$$P(\hat{e}_j | \mathbf{h}_j) = \frac{\exp(\mathbf{h}_j^\top \cdot \mathbf{w}_c) \odot \mathbf{c}_j}{\sum_{c' \in C} \exp(\mathbf{h}_j^\top \cdot \mathbf{w}_{c'}) \odot \mathbf{c}_j}, \tag{5}$$

where $\mathbf{w}_c$ is the c-th column vector from a trainable parameter matrix $\mathbf{W}^C$. Note that we use *argmax* to get the final prediction of road segment ID $\hat{e}_j$. With the constraint mask layer, points in low-sampling-rate trajectory τ would be projected onto limited road segments instead of the whole road segments space.

3.3 Attention Mechanism

As above mentioned, we only consider local region constraint, while global correlations of low-sampling-rate trajectories are not modeled. Inspired by the attention mechanism that is widely used in

natural language translation [1, 26], we introduce the attention mechanism [1] into the decoder.

The goal of attention mechanism is to compute the *similarity* between query vector (i.e., current hidden state in the decoder) and key vectors (i.e., outputs from the encoder) to generate a context vector $\mathbf{a}$. Thus, the hidden state $\mathbf{h}_j$ in the decoder is updated to:

$$\mathbf{h}_j = GRU(\mathbf{h}_{j-1}, e_{j-1}, r_{j-1}, \mathbf{a}_j), \tag{6}$$

where the context vector $\mathbf{a}_j$ is computed by a weighted sum of all the output vectors $\mathbf{s}$ from the encoder, where $\mathbf{a}_j$ is formulated as:

$$\begin{aligned} \mathbf{a}_j &= \sum_{i=1}^{n} \alpha_{j,i}\mathbf{s}_i, \\ \alpha_{j,i} &= \frac{\exp(u_{j,i})}{\sum_{i'=1}^{n}\exp(u_{j,i'})}, \\ u_{j,i} &= \mathbf{v}^\top \cdot \tanh(\mathbf{W_h}\mathbf{h}_j + \mathbf{W_s}\mathbf{s}_i), \end{aligned} \tag{7}$$

where $\mathbf{v}$, $\mathbf{W_h}$ and $\mathbf{W_s}$ are the learnable parameters, and $\mathbf{h}_j$ denotes the current mobility status from the decoder.

3.4 Attribute Module

The recovery accuracy is also influenced by traffic conditions since vehicles are not moving at a constant speed in the real world. The traffic speed is affected by spatial context, temporal dependencies and external factors, such as weather conditions, time, POI distribution and etc [15]. To tackle our challenge (3), diverse complex factors, we incorporate such information to devise an attribute module in our model. The attribute module includes two types of factors:

- The environmental context features $\mathbf{f_e}$. As shown in Fig. 5, we incorporate the attributes of weather conditions (sunny, rainy, cloudy, etc), holiday (whether today is a holiday or not), and time (hour of the day) to extract the effect of the environmental context features. We attempt to implement the environmental context features as an independent block. Note that these factors are categorical values which cannot be directly fed into the neural network. We use one-hot encoding to represent them since the dimension of each factor is not large. After concatenating the one-hot encoding results, a fully connected network (FCN) is implemented to learn the embedding of all the features, which is fused with the last hidden state of the encoder as the initial hidden layer of the decoder, i.e. $\mathbf{h}_0 = FCN(\mathbf{s}_n, Emb(\mathbf{f_e}))$. The intuition is that the environmental context features do not change significantly with trajectories mobility. Therefore, we combine the embedding vector with the context vector of the encoder and input them at the beginning of the decoder.
- The spatial context features $\mathbf{f_s}$. Different from the stable environmental context features, the spatial context features, such as POI and road network, change rapidly as vehicles move. Thus, we input such features for each time step in the encoder and the decoder respectively. We use the POIs density of different categories as POIs features, and extract the properties of road network as network features, i.e., number of intersections and level of road segments. Thus, the hidden state $\mathbf{s}_i$ of the encoder and the hidden state $\mathbf{h}_j$ are renewed as,

$$\begin{aligned} \mathbf{s}_i &= GRU(\mathbf{s}_{i-1}, \mathbf{g}_{i-1}, \mathbf{f_s}_{j-1}), \\ \mathbf{h}_j &= GRU(\mathbf{h}_{j-1}, e_{j-1}, r_{j-1}, \mathbf{a}_j, \mathbf{f_s}_{j-1}). \end{aligned} \tag{8}$$

3.5 Algorithm Training

We finally elaborate the training procedure of our model which is trained end to end. Recall that during the training phrase, we predict road segment ID and moving ratio simultaneously. For the road segment ID prediction, we adopt the cross entropy as the loss function:

$$\begin{aligned} \mathcal{L}_1(\theta) = -\sum_{(\tau,\tilde{\tau})\in\mathcal{D}_{sub}}\sum_{j=1}^{|\tilde{\tau}|}\sum_{\ell=1}^{L}\mathbf{1}\{a_j.e = e_\ell\}\log(P_\theta(\hat{a}_j.e = e_\ell|\mathbf{d}_{1:j-1})), \\ \text{s.t. } \mathbf{d}_{j-1} = (\tau, \tilde{\tau}_{1:j-1}, \mathbf{f_s}_{j-1}, \mathbf{f_e}), \end{aligned} \tag{9}$$

where τ is the low-sampling-rate trajectory, $\tilde{\tau}$ is the target ϵ-MM trajectory, $|\tilde{\tau}|$ is the length of the ϵ-MM trajectory, L is the size of road segments, $a_j.e$ is the ground truth of road segment ID, $\hat{a}_j$ is the prediction, P_θ represents the neural network for predicting road segments, $\mathbf{f_s}$ and $\mathbf{f_e}$ are external factors vectors. $\mathcal{D}_{sub}$ means the dataset consisting of low-sampling-rate trajectories and ϵ-MM trajectories, which is the subset of the training set $\mathcal{D}$.

We also implement the mean squared error as the loss function for the moving ratio prediction:

$$\begin{aligned} \mathcal{L}_2(\theta) = -\sum_{(\tau',\tilde{\tau})\in\mathcal{D}_{sub}}\sum_{j=1}^{|\tilde{\tau}|}(a_j.r - R_\theta(\mathbf{d}_{j-1}))^2, \\ \text{s.t. } \mathbf{d}_{j-1} = (\tau', \tilde{\tau}_{1:j-1}, \mathbf{f_s}_{j-1}, \mathbf{f_e}), \end{aligned} \tag{10}$$

where $a_j.r$ is the ground truth of real moving ration and R_θ represents the neural network for predicting moving ratio and other parameters are the same as above.

Overall, the final model optimization function is weighted combined with these two loss functions and formulated as:

$$\mathcal{L}_t = \mathcal{L}_1(\theta) + \lambda\mathcal{L}_2(\theta) \tag{11}$$

where λ is a tunable parameter that linearly balances the trade-off between two tasks in our work.

Algorithm 1 illustrates the training process of the MTrajRec model. During the training process, we apply the gradient descent approach to update parameters $\boldsymbol{\theta}$, with learning rate η and a predefined $epochs_{max}$. We first extract attributes including external factors $\mathbf{f_e}$ and spatial context features $\mathbf{f_c}$. Then, we select one pair of trajectories τ and $\tilde{\tau}$. Lastly, we update MTrajRec parameters θ by using Eq 11, with η as the step size (Line 5).

Algorithm 1 MTrajRec Training

Require: Trajectories $\mathcal{T}$, ϵ-MM Trajectories $\tilde{\mathcal{T}}$, features $\mathbf{f_e}$, $\mathbf{f_c}$, initialized parameters $\boldsymbol{\theta}$, learning rate lr, max iteration $epochs_{max}$.
Ensure: A well trained MTrajRec with parameters $\boldsymbol{\theta}$.
1: Extract attributes.
2: Sample a pair of trajectories τ and $\tilde{\tau}$.
3: **while** epoch < $epochs_{max}$ **do**
4: Calculate gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$ using Eq 11.
5: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla\mathcal{L}(\boldsymbol{\theta})$.
6: **end while**

4 EXPERIMENTAL EVALUATION

4.1 Experimental Settings

4.1.1 Data Description. We validate the effectiveness of our model on a real-world taxi trajectory dataset and a road network from OpenStreetMap [1]. The dataset contains trajectories of 122, 390 drivers and 6.20 million GPS records in Jinan, Shandong over a period of 1 month, in September 2017. All the trajectories are completed sampled every 15 seconds. It covers a rectangular area from (36.6456, 116.9854) to (36.6858, 117.0692) which is around 7.47 km long and 4.47 km wide. There are 2, 571 road segments in the area.

We split the dataset into training set, validation set and test set with a splitting ratio of 7: 2: 1. Since the dataset is completely sampled, we generate low-sampling-rate trajectories by randomly sampling points from high-sampling-rate trajectories with a keep ratio $kr\%$. According to [31], taxis usually report their GPS positions with a low sampling rate to save communication and energy cost. More than 60% trajectory data is sampled every $2 \sim 6$ mins. Thus, we further vary the keep ratio of $kr\%$ in the set $\{6.25\%, 12.5\%, 25\%\}$ to evaluate the robustness of our proposed model. Since the original trajectories are sampled every 15 seconds, the generated low-sampling-rate trajectories with $kr\% = 6.25\%, 12.5\%, 25\%$ can be considered as the average time interval of such trajectories is 4 mins, 2 mins and 1 min respectively. A smaller keep ratio indicates to a larger number of missing points in the low sampling rate trajectories. For the high-sampling-rate trajectories, we utilize HMM algorithm on the original trajectories [20] to get ground truth, since the map matching accuracy can reach as high as 99% with a sampling interval around $10 \sim 15$ seconds [20].

4.1.2 Evaluation Metrics. The purpose of our defined problem is to recover low-sampling-rate trajectories from free space to high-sampling-rate trajectories constrained onto the road network. Thus, we adopt both accuracy of road segments recovery and distance error of location inference to show the performance of our model and baseline methods as follows:

MAE & RMSE. We adopt two distance measurements to evaluate the location recovery performance. *MAE* is the mean absolute error and *RMSE* is the root mean squared error between ground truth values and predicted values. However, as shown in Fig. 6, if we directly use earth distance to compute the error, the prediction of pre_2 is better than pre_1, while pre_2 cannot reach the ground truth through the dash line in the real world. Thus, we should calculate the distance error based on road network. We update earth distance to road network constrained distance to evaluate the distance error. The smaller values of *MAE* and *RMSE* are, the better performance the model represents. *MAE* and *RMSE* are formulated as:

$$\begin{aligned} MAE &= \frac{1}{m}\sum_{j=1}^{m}|dis(a_j, \hat{a}_j)|, \\ RMSE &= \sqrt{\frac{1}{m}\sum_{j=1}^{m}(dis(a_j, \hat{a}_j))^2}, \\ \text{s.t. } dis(a_j, \hat{a}_j) &= min(rn\_dis(a_j, \hat{a}_j), rn\_dis(\hat{a}_j, a_j)), \end{aligned} \tag{12}$$

where a_j is the ground truth location, $\hat{a}_j$ is the predicted map matched trajectory point and $rn\_dis(a_j, \hat{a}_j)$ is the distance of shortest path between prediction and ground truth. Since the road network is a directed graph, the road network based distance from a_j to $\hat{a}_j$ is not equal to the distance from $\hat{a}_j$ to a_j, thus we use the minimal distance as the final error.

Recall & Precision. We use recall and precision to evaluate the performance of route recovery by comparing the recovered road segments $\mathcal{E}_R$ to the ground truth $\mathcal{E}_G$. Following previous work [4, 12], *Recall* is defined as $recall = \frac{|\mathcal{E}_R \cap \mathcal{E}_G|}{|\mathcal{E}_G|}$, and *Precision* is denoted as $precision = \frac{|\mathcal{E}_R \cap \mathcal{E}_G|}{|\mathcal{E}_R|}$. The larger values of *Recall* and *Precision* indicate that the methods predict road segments more accurately.

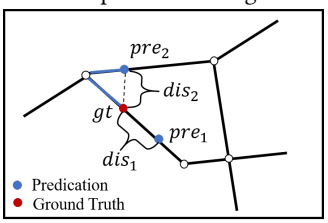


Figure 6: Examples of Distance Error

4.1.3 Baseline Algorithms. We compare our MTrajRec with several representative baselines. To our best knowledge, there is no existing solution that can recover low-sampling-rate trajectories to high-sampling-rate trajectories and map match onto the road network simultaneously. Therefore, we design the following two-stage pipelines for comparison:

- **Linear** [10] + **HMM** [20]: It recovers the locations by assuming trajectories are moving straightly and uniformly and then matches trajectories onto the road network. Here we implement HMM as the map matching algorithm due to its high accuracy in high-sampling-rate trajectories map matching [20].
- **DHTR** [28] + **HMM** [20]: It devises a subseq2seq model with Kalman Filter to recover trajectories in free space, which is the state-of-the-art method in the field of trajectory recovery. After getting the recovered high-sampling-rate trajectories, we introduce HMM to match them onto the road network.
- **DeepMove** [7] + **Rule**: It incorporates multiple factors with recurrent neural network to predict human mobility for next-step trajectory prediction. We adapt it to this task by consecutively predicting each missing road segment and then use the centric location as the final prediction.

4.1.4 Variants. To evaluate each component of our proposed model, we also compare it with different variants of MTrajRec:

- **MTrajRec-noCons**: We remove constraint mask layer in multi-task block to evaluate the relevance of this part.
- **MTrajRec-noAttn**: We remove the attention mechanism to detect the importance of attention mechanism.
- **MTrajRec-noAtts**: We remove the attribute module from our model to reveal the significance of this component.

4.1.5 Implementations. We train the deep neural network with machine learning library Pytorch, version 1.7.1. Our experiments run on a GPU server with 64 GB memory and Tesla V100 GPU. Detailed information is provided to support the reproducibility of the results in Appendix A.

[1] http://www.openstreetmap.org/

Table 1: Overall performance comparison. The best result for each evaluation metric is in bold. A smaller keep ratio indicates a larger number of missing points in low-sampling-rate trajectories. Note that *MAE* and *RMSE* are in *km*.

Methods	6.25%				12.5%				25%			
	Recall	Precision	MAE	RMSE	Recall	Precision	MAE	RMSE	Recall	Precision	MAE	RMSE
Linear+HMM	0.4019	0.3388	0.662	1.107	0.5122	0.4592	0.535	1.030	0.6291	0.6065	0.398	0.820
DHTR+HMM	0.5110	0.3601	0.637	1.099	0.5984	0.4379	0.485	0.958	0.6581	0.4916	0.368	0.844
DeepMove+Rule	0.6409	0.7754	0.339	0.696	0.6572	0.7924	0.311	0.666	0.6762	0.8085	0.279	0.628
MTrajRec-NoCons	0.6472	0.7835	0.303	0.648	0.6748	0.7998	0.260	0.596	0.7001	0.8066	0.227	0.575
MTrajRec-NoAttn	0.6837	0.7938	0.290	0.631	0.7105	0.8066	0.245	0.594	0.7308	0.8144	0.209	0.566
MTrajRec-NoAtts	0.6925	0.7981	0.278	0.628	0.7121	0.8073	0.238	0.603	0.7437	0.8116	0.199	0.567
MTrajRec	**0.6972**	**0.8018**	**0.270**	**0.610**	**0.7235**	**0.8137**	**0.229**	**0.590**	**0.7498**	**0.8198**	**0.194**	**0.564**

4.2 Results

4.2.1 Overall Performance. We compare our MTrajRec with the baseline models in terms of *Recall*, *Precision*, *MAE* and *RMSE*. The performance of different approaches with different keep ratios for trajectory recovery is presented in Table 1. We have the following observations:

(1) The first two pipelines interpolate missing values for low-sampling-rate trajectories in free space and then map them onto the road network. Comparing these two models, we can see that DHTR + HMM is better than Linear + HMM, especially when the keep ratio is small. This is because DHTR is designed with advanced sequential neural networks, which is able to utilize the spatio-temporal information in low-sampling-rate trajectories, while linear interpolation cannot model complex mobility regularity.
(2) DHTR + HMM and DeepMove + Rule are deep learning based method, but the performance of DeepMove + Rule is better than the former one. As the keep ratio decreases, i.e., the number of missing points increases as well as the uncertainty between two consecutive points grows, the performance of DHTR + HMM is worse than DeepMove + Rule. This is because DeepMove is directly trained to predict road segments, which can better capture the constraint of the road network compared with DHTR.
(3) It is clear that our approach attains the best performance across all of the evaluation metrics with different keep ratios. Compared with other baselines, the improvement of ratio is the most significant at the lowest keep ratio. Specifically, *Recall* and *Precision* of MTrajRec outperform the best baseline by 8.7% and 3.4% when the keep ratio is 6.25%. *MAE* and *RMSE* are reduced by 20.4% and 14.1% respectively. This proves the effectiveness of our MTrajRec in trajectory recovery. The fundamental reasons for such improvement lie in two aspects. First, we devise a multi-task Seq2Seq model to predict road segments and moving ratio simultaneously which is able to reduce error compared with two-stage pipelines. Second, we introduce several components such as the constraint mask layer, the attention mechanism and the attributes module to extract more information, which will be elaborated in next section.

4.2.2 Running Efficiency. To evaluate the efficiency of MTrajRec, we compare the running time of our algorithm with other baseline models. As can be seen in Fig. 7, MTrajRec and DeepMove+Linear run much faster than other baseline models. This is because both of MTrajRec and DeepMove+Linear only need to make a forward computation of neural network to recover trajectories, which only requires $O(n)$ computation complexity. On the contrary, other two approaches rely on heavy computations of dynamic programming to do map matching task, with a computation complexity of $O(n^2)$. With the keep ratio increases, the running time of MTrajRec and DeepMove+Linear decreases, since less missing points need to be predicted. However, the running time of other two baseline methods increases since more and more points need to be matched by HMM. Although MTrajRec is slightly slower than DeepMove+Linear, the accuracy is much higher. Thus, such difference can be ignored.

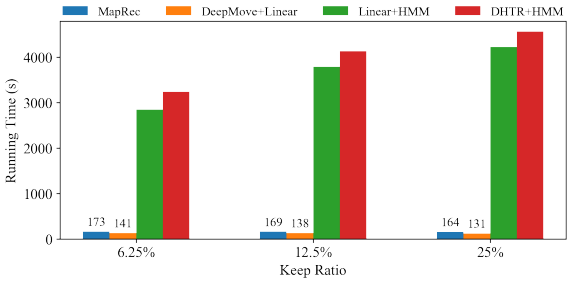
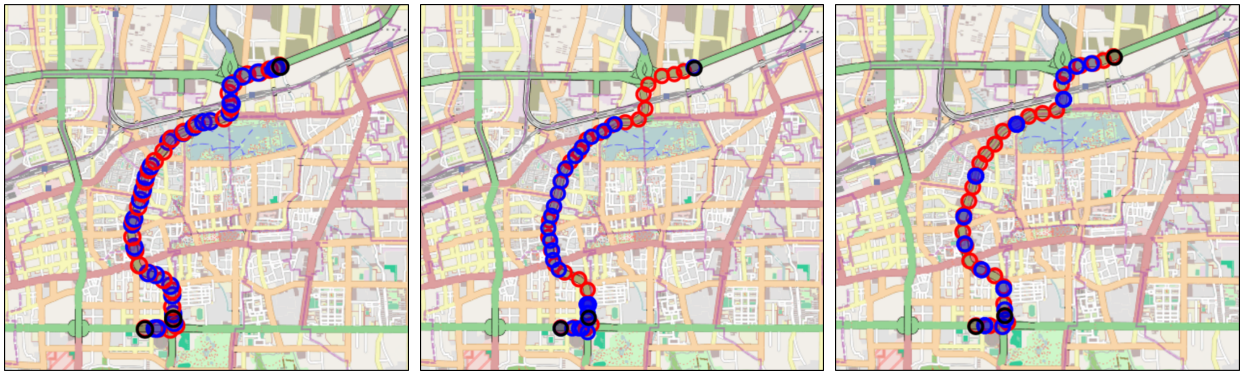


Figure 7: Running Efficiency

4.2.3 Importance of the Constraint Mask. We introduce a constraint mask layer into MTrajRec to overcome the limits of converting coordinates to grid cells. To evaluate the importance of constraint mask layer, we compare *MTrajRec-noCons* to MTrajRec. As can be seen from Table. 1, both MTrajRec and MTrajRec-noCons get a better performance as the keep ratio increases. But when the constraint mask layer is removed, the performance declines significantly. Especially, it causes *MAE* to grow 25% and *Recall* to shrink 8% with 6.25% keep ratio. This is because the constraint masks introduce prior knowledge for the existing points, which enhances the missing information by utilizing grid cells.

4.2.4 Importance of the Attention Mechanism. In this section, we remove the attention mechanism from MTrajRec to test its contribution. Similar as above, the performance of both MTrajRec and MTrajRec-noAttn increase as the keep ratio increases because the missing points are reduced, making the recovery task easier. As illustrated in Table. 1, the results of MTrajRec-noAttn falls obviously comparing with MTrajRec. Particularly, *MAE* increases 7% and *Precision* decreases 2% after removing the attention mechanism at the keep ratio 6.25%. A possible reason is that attention mechanism can effectively strengthen the spatial constraints for the missing locations.

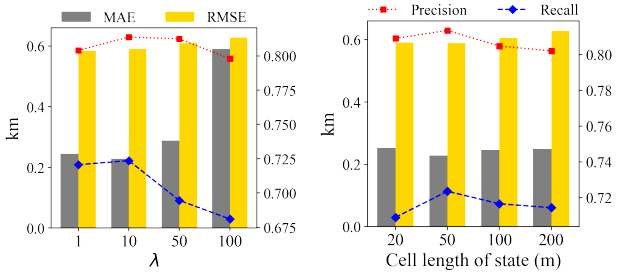
4.2.5 Importance of the Attribute Module. We also evaluate the relevance of the attribute module by removing all the external factors. Table. 1 shows that the results of MTrajRec-noAtts drops



(a) MTrajRec (b) DHTR+HMM (c) DeepMove+Rule

Figure 8: Screenshots of Trajectory Recovery. Black points represent the low-sampling-rate trajectory, points in red are ground truth coordinates of 15*s*-MM trajectory and the blue points stand for predicted locations of 15*s*-MM trajectory.

slightly when taking out the attribute module. The contribution of attribute module is not as significant as the constraint mask and the attention mechanism, i.e., the *MAE* of MTrajRec only drops 3% and the *Precision* of MTrajRec only increases 0.7% when adding this component. Probably because the previous two provide enough constraints in the model.



(a) Varying of λ (b) Varying of cell length

Figure 9: Parameters Tuning

4.2.6 Parameter Tuning. Apart from evaluating the components of our proposed model MTrajRec, there are two important parameters to tune in our model.

Weight of multi-task λ. To show the effectiveness of the multi-task learning component, we first evaluate our model under different combinations λ in range of 1, 10, 50 and 100. As shown in Fig. 9(a), the gray and yellow bars show results of *MAE* and *RMSE* and the red and blue lines present *Precision* and *Recall* respectively. We get the best performance when $\lambda = 10$, while the performance gets worse as the λ increases more or decreases. It indicates that our MTrajRec benefits from a good balance of this two tasks.

Cell length. As mentioned in Section 3, we map numerical coordinates into discrete units to extract spatial dependencies and reduce the computation complexity. When a smaller cell length is used, we can obtain more accurate spatial information but the number of grid cells increases leading to complex modeling problem and vice versa. Thus, there is a trade-off between accuracy and complexity with the setting of cell length. We vary the cell length to 20, 50, 100 and 200 meters to tune the parameter. Fig. 9(b) shows the varying results of different cell lengths. As can be seen, the performance achieves the best when the cell length is 50 meters. As the cell length increases to 100 and 200 meters, the *Precision* and *Recall* increase and *MAE* and *RMSE* decrease, probably some of the spatial information is lost when mapping to grid cells. However, the performance also gets worse when the cell length decreases to 20 meters since the large number of grid cells increase the complexity of training.

4.3 Qualitative Analysis

Fig. 8 presents screenshots of the visualized recovery results of our MTrajRec compared with two baseline models (DHTR + HMM, DeepMove + Rule) on the same low-sampling-rate trajectory data at a keep ratio 6.25%. Black points represent the low-sampling-rate trajectory. Points in red are ground truth coordinates of 15*s*-MM trajectory and the blue points stand for predicted locations of 15*s*-MM trajectory. It is clear that our MTrajRec finds the right route and the recovered positions are more reliable and adaptive than the other two baselines. Fig. 8(b) shows the recovery results of DHTR+HMM, which recovers the low-sampling-rate trajectory in free space and then implements a map matching algorithm to match the trajectory onto the road network. Although it predicts the right path, the locations are not correctly recovered especially no correct points are predicted in the top right area. A possible reason is that there are a large number of missing values to be predicted, while DHTR cannot recover locations with such low sampling rate. The keep ratio is only 6.25% which is about 4 times lower than the smallest keep ratio in DHTR [28]. Fig. 8(c) illustrates the recovery results of DeepMove + Rule, which employs a deep learning model to match the low-sampling-rate trajectory onto the road network and then uses the centric location as the final prediction. With the constraint of road network, DeepMove + Rule finds the right path as the left two methods. However, the recovered points move discontinuously and several blue points in the middle area fail to move forward. This is because DeepMove + Rule can only predict the possible road segments but it does not have the ability to infer moving speed of the vehicle.

5 RELATED WORK

Trajectory Data Mining. Trajectory data mining [33] discovers various knowledge from massive trajectory data, namely a few, traffic condition prediction [16], travel time estimation [9, 22] and driver behavior learning [5, 23]. In particular, Hong et al. [9] leveraged heterogeneous information graph to solve estimated time of

arrival task based on road network and high sampling rate vehicle trajectories. Dong et al. [5] proposed a deep-learning framework to analyze driving behavior based on trajectory data. Their empirical studies revealed that the performance of any approaches can become poor when the sampling rate is lower than 10 seconds. Such work relies on high-sampling-rate road network constrained trajectories, which can benefit from our work.

Trajectory Recovery. Recovering low-sampling-rate trajectory is an important problem to get more information and reduce uncertainty [28–30]. In particular, Wang et al. [28] recovered a high-sampling-rate trajectory from a irregular low-sampling-rate trajectory by integrating the subseq2seq with a calibration component of Kalman Filter. Xia et al. [30] proposed an attentional neural network model to densify individual trajectory by recovering unobserved locations based on historical trajectories. Xi et al. [29] proposed a Bi-directional Spatial and Temporal Dependence and users' Dynamic Preferences model to identify missing POI check-in. Apart from these recovery methods, some works related to next-step or short-term location prediction can also be adopted for recovery [7, 14, 19]. However, such frameworks are implemented on free space, as opposed to our problem which aims to recover trajectories onto road network. Besides, most of the existing works model the sequence of location IDs rather than the numerical coordinate information.

Sequence-to-sequence Models. The sequence-to-sequence model (Seq2Seq) [25] is an architecture for domain translation, which has been widely used in trajectory data mining, namely but a few, trajectory generation, trajectory similarity learning, anomaly detection, etc [13, 17, 21, 28]. Park et al. [21] generated the future trajectory sequence of surrounding vehicles via Seq2Seq model. Li et al. [13] proposed a Seq2Seq framework to learn representations of trajectories to support trajectory similarity computation. However, to our best knowledge, we are the first one to employ multi-task learning into Seq2Seq model to recover map constrained trajectories.

6 CONCLUSION

In this paper, we propose a novel end-to-end deep learning model MTrajRec for recovering low-sampling-rate trajectories to high-sampling-rate map-matched trajectories. We introduce multi-task learning into Seq2Seq model to ensure the generated trajectories are map matched onto the road network. The *constraint mask, attention mechanism and attribute module* are implemented to improve the performance. The experimental results illustrate that MTrajRec outperforms state-of-the-art works by reducing recover error around 20.4% with the keep ratio at 6.25% on a real world dataset. As future work, we plan to extend the proposed model by incorporating more user preference, e.g., user identification information.

7 ACKNOWLEDGEMENTS

This work was supported by the National Key R&D Program of China (2019YFB2101801). The research of Huimin Ren and Yanhua Li was supported in part by NSF grants IIS-1942680 (CAREER), CNS-1952085, CMMI1831140, and DGE-2021871.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.
[2] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.
[4] G. Cui, J. Luo, and X. Wang. Personalized travel route recommendation using collaborative filtering based on gps trajectories. *International journal of digital earth*, 11(3):284–307, 2018.
[5] W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang. Characterizing driving styles with deep learning. *arXiv:1607.03611*, 2016.
[6] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang. Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. In *Proc. of the 26th ACM SIGKDD*, pages 2697–2705, 2020.
[7] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proc. of the 2018 WWW conference*, pages 1459–1468, 2018.
[8] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. *arXiv:1512.05287*, 2015.
[9] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye. Heteta: Heterogeneous information network embedding for estimating time of arrival. In *Proc. of the 26th ACM SIGKDD*, pages 2444–2454, 2020.
[10] S. Hoteit, S. Secci, S. Sobolevsky, C. Ratti, and G. Pujolle. Estimating human trajectories and hotspots through mobile phone data. *Computer Networks*, 64:296–307, 2014.
[11] R. R. Joshi. A new approach to map matching for in-vehicle navigation systems: the rotational variation metric. In *Proc. of ITSC 2001*, pages 33–38. IEEE, 2001.
[12] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proc. of the 19th ACM CIKM*, pages 579–588, 2010.
[13] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. Deep representation learning for trajectory similarity computation. In *Proc. of IEEE 34th ICDE*, pages 617–628. IEEE, 2018.
[14] D. Lian, Y. Wu, Y. Ge, X. Xie, and E. Chen. Geography-aware sequential location recommendation. In *Proc. of the 26th ACM SIGKDD*, pages 2009–2019, 2020.
[15] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI*, pages 3428–3434, 2018.
[16] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, and F. Wu. Deep sequence learning with auxiliary information for traffic prediction. In *Proc. of the 24th ACM SIGKDD*, pages 537–546, 2018.
[17] Y. Liu, K. Zhao, G. Cong, and Z. Bao. Online anomalous trajectory detection with deep generative sequence modeling. In *36th ICDE*, pages 949–960. IEEE, 2020.
[18] Y. Lou, C. Zhang, X. Xie, Y. Zheng, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proc. of 18th ACM SIGSPATIAL*, 2009.
[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NeurIPS*, pages 3111–3119, 2013.
[20] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proc. of the 17th ACM SIGSPATIAL*, pages 336–343, 2009.
[21] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE IV*, pages 1672–1678. IEEE, 2018.
[22] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. Route travel time estimation using low-frequency floating car data. In *16th International IEEE ITSC*, pages 2292–2297. IEEE, 2013.
[23] H. Ren, M. Pan, Y. Li, X. Zhou, and J. Luo. St-siamesenet: Spatio-temporal siamese networks for human mobility signature identification. In *Proc. of the 26th ACM SIGKDD*, pages 1306–1315, 2020.
[24] H. Su, K. Zheng, J. Huang, H. Wang, and X. Zhou. Calibrating trajectory data for spatio-temporal similarity analysis. *The VLDB Journal*, 24(1):93–116, 2015.
[25] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in NeurIPS*, 27:3104–3112, 2014.
[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2017.
[27] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng. When will you arrive? estimating travel time based on deep neural networks. In *Proc. of the AAAI*, volume 18, pages 1–8, 2018.
[28] J. Wang, N. Wu, X. Lu, X. Zhao, and K. Feng. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE TKDE*, 2019.
[29] D. Xi, F. Zhuang, Y. Liu, J. Gu, H. Xiong, and Q. He. Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In *Proc. of the AAAI*, volume 33, pages 5458–5465, 2019.
[30] T. Xia, Y. Qi, J. Feng, F. Xu, F. Sun, D. Guo, and Y. Li. Attnmove: History enhanced trajectory recovery via attentional network. *arXiv:2101.00646*, 2021.
[31] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An interactive-voting based map matching algorithm. In *Proc. of 11th MDM*, pages 43–52. IEEE, 2010.
[32] S. Zheng, Y. Yue, and J. Hobbs. Generating long-term trajectories using deep hierarchical networks. *Advances in NeurIPS*, 29:1543–1551, 2016.
[33] Y. Zheng. Trajectory data mining: an overview. *ACM TIST*, 6(3):1–41, 2015.

A APPENDIX FOR REPRODUCIBILITY

In this section, we provide detailed information to support the reproducibility of the results in this paper. To support the reproducibility of the results in this paper, we have released our code at Github: https://github.com/huiminren/MTrajRec.

A.1 Details of Data Preprocessing

In this stage, we employ the road network to clean up trajectories, down sample trajectories and map match trajectories to get the training, validation and test dataset for the model.

Due to sensor noises and other factors, the trajectories generated by GPS device usually contain noise points. We removed noise points by implementing a heuristics-based outlier detection method [33]. They first calculate the travel speed of each point in a trajectory based on its precursor and itself. The current estimated point will be filtered out from the trajectory if the speed is larger than a threshold. In this work, we set the speed threshold to 120 km/h and replace the noise with the average location of its precursor and successor.

Table 2: Description of Dataset

	Training	Validation	Test
# of Trajectories	160, 617	45, 916	22, 825

To keep enough information, we filter out the trajectory which duration is less than 5 mins. Table. 2 gives the overview about our dataset. Based on the cleaned trajectories, we randomly generate low sampling rate trajectories using a keep ratio $kr\%$ as trajectories $\mathcal{T}$. In other words, for each cleaned trajectory, we only keep $kr\%$ points from it randomly. To get ground truth, we utilize HMM map matching algorithm [20] to map the cleaned trajectories onto the road network. This algorithm shows that with a sampling interval of 15 seconds, the map matching accuracy can reach as high as 99% [20], indicating that the matched results are enough to be the ground truth.

A.2 Detailed Settings of MTrajRec

Parameters of training MTrajRec. We train our model on Jinan taxi dataset as a multi-task learning problem. To minimize the loss function Eq.(11), we implement the back-propagation on feed-forward networks by the adaptive moment optimizer (Adam) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The mini-batch size is 128 and the learning rate is 0.001. We trained network for 20 epochs to get the converged results.

Settings of MTrajRec. Since we design a multi-task learning model, the best λ in Eq.(11) is 10 in this work. The number of hidden states in the encoder and the decoder is 512. To prevent overfitting, we set dropout ratio as 0.5 in GRUs. In the multi-task block, we implement a dense layer with 2571 units to predict road segment ID via the constraint mask layer. The moving ratio is predicted by a fully-connected network with hidden units in [640, 512, 1]. We use ReLU activation function for all hidden layers and sigmoid activation function at the output layer. In the attention mechanism, we use two dense layers with units in [1024, 514] to learn the context vector α. In the attribute module, features are embedded by a dense layer with 30 hidden units and a Tanh activation function.

A.3 Detailed Settings of Baselines

- **Linear + HMM** There is no parameters to learn in linear interpolation. And three parameters in the HMM algorithm should be defined. First, it sets a search distance D to query candidate road segments. Second one is a transition probability $f_t(\beta)$ to indicate the probability of a vehicle moving between the candidate road matches at these two times. And the last one is the emission probability $f_e(\sigma)$ to show the likelihood that a measurement resulted from a give state. In this paper, we set $D = 50$, $\beta = 2$ and $\sigma = 5$.
- **DHTR + HMM** We follow the same hyperparameters setting of DHTR [28]. The DHTR is trained using Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a learning rate of 0.001 for 20 epochs with a batch size 128. The setting of HMM is the same as above mentioned.
- **DeepMove + Rule** Instead of predicting grid cells, we predict road segments and then use the centric location as final prediction to compare with our method. DeepMove is trained with two one-layer LSTMs as well as an attention mechanism. The number of hidden states in the encoder and the decoder is 256. And the setting of hyperparameters of the attention mechanism is the same as our MTrajRec. The other training paraemters are the same as DHTR.