Map-Informed Trajectory Recovery With Adaptive Spatio-Temporal Autoencoder

Yongchao Ye, *Student Member, IEEE*, Ao Wang, Adnan Zeb, Shiyao Zhang, *Member, IEEE*, and James Jianqiao Yu, *Senior Member, IEEE*

***Abstract*— The recovery of coarsely sampled trajectories considering the road network topology characteristics is a crucial task for many downstream applications in intelligent transportation systems. Existing approaches in this domain primarily focus on extracting spatio-temporal correlations for the observed trajectory points but neglect the critical role of road network topology characteristics in making the recovery results more accurate and realistic. In addition, too many road segments in cities undermine the model inference performance. To address these challenges, we propose a novel Map-informed Adaptive Spatio-Temporal Autoencoder, which follows an encoder-decoder architecture for trajectory recovery. Specifically, we utilize a pre-trained attributed network embedding module to incorporate the road segment characteristics into the input data to make it easier for the model to extract the spatio-temporal dependencies from coarse trajectories. Furthermore, we construct a novel adaptive mask inference module that contains a distance-based mask matrix and a learnable adaptive mask matrix to assist the model in making segment inferences by weighting each candidate segment adaptively in the recovery process. To evaluate the performance of the proposed model, we conduct a series of comprehensive case studies on two representative real-world trajectory datasets. The experimental results demonstrate that the proposed model consistently outperforms state-of-the-art approaches.**

***Index Terms*— Trajectory recovery, missing data, road network, spatio-temporal modeling, deep learning.**

I. Introduction

With the development of global positioning systems (GPS) devices and the advancement of urbanization, increasing trajectory data based on urban transportation networks are generated and collected every day. Massive trajectory data has greatly contributed to the development of downstream applications, such as travel time estimation [1], traffic forecasting [2], transportation mode identification [3], etc. However, due to factors such as device energy consumption and communication errors, trajectory data is often coarsely sampled, therefore discarding detailed movement information and raising uncertainty in the recorded trajectories [4], [5]. Such uncertainty poses a challenge to the development of downstream intelligent transportation applications.

In light of this practical challenge, recovering the collected low-sampling-rate (LSR) trajectory data is a critical task for the development of downstream trajectory-based applications. This issue has been widely explored as a significant research topic in the field of intelligent transportation systems (ITS) for several decades [6], [7]. A straightforward and preliminary approach to recovering the trajectory is by assuming a constant speed of the object and using linear interpolation to fill the missing points [8]. However, this method fails to consider personalized travel patterns and dynamic changes in the trajectory, which can result in inferior recovery performance. Further research is needed to develop more sophisticated methods to improve the accuracy and reliability of trajectory recovery.

In recent years, the emergence of deep learning has provided a new approach to trajectory recovery. Existing studies have demonstrated the significant advantages of deep learning-based approaches in processing traffic data, such as traffic data imputation and prediction [9], [10]. Considering the complex temporal relationship between different geo-locations among a trajectory, Wu et al. [11] utilized Recurrent Neural Network (RNN) to model trajectories. Wang et al. [12] proposed a DHTR module, which integrates a sequence-to-sequence model with Kalman filter (KF) to recover the LSR trajectory. The module represents each trajectory coordinate with discrete units to reduce the inference complexity. In the meantime, considering the strong correlation between the distribution of trajectories in cities and the topology of traffic networks, a map-matching process is commonly appended to make the recovery results more realistic [13].

Although map-matching can make the recovered trajectory more realistic, the two-stage pipeline (namely, inference and map-matching) is inefficient and also raises the error accumulation issue. Ren et al. circumvented this issue by inferring the road segment ID and moving ratio simultaneously [13]. However, the neglect of the traffic network

Received 20 September 2023; revised 7 March 2024 and 9 September 2024; accepted 15 October 2024. This work was supported by the Stable Support Plan Program of Shenzhen Natural Science Fund under Grant 20220815111111002. The Associate Editor for this article was F. Xia. *(Corresponding authors: Shiyao Zhang; James Jianqiao Yu.)*

Yongchao Ye is with the Department of Data Science, City University of Hong Kong, Hong Kong, SAR, China (e-mail: yongchao.ye@my.cityu.edu.hk).

Ao Wang is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: 12132359@mail.sustech.edu.cn).

Adnan Zeb is with the School of Intelligent Manufacturing and Smart Transportation, Suzhou City University, Suzhou 215104, China (e-mail: adnanzeb@szcu.edu.cn).

Shiyao Zhang is with the Research Institute for Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: zhangsy@sustech.edu.cn).

James Jianqiao Yu is with the Department of Computer Science, University of York, YO10 5DD York, U.K. (e-mail: jqyu@ieee.org).

Digital Object Identifier 10.1109/TITS.2024.3483941

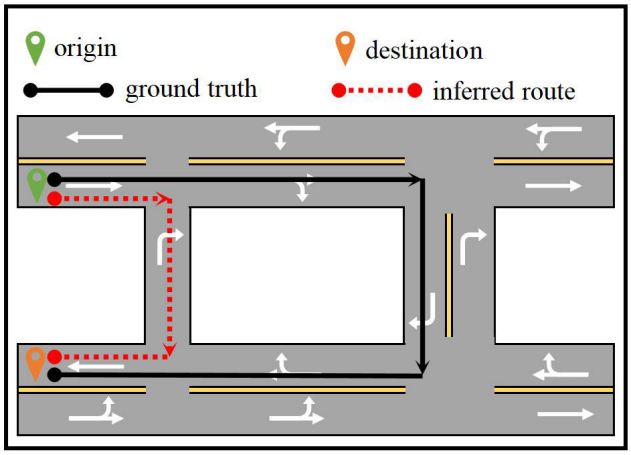


Fig. 1. Trajectory recovery based on the urban network. The black curve denotes the real trajectory. Due to the lack of road features, the model can easily infer the red curve as the recovery result, which is apparently unreasonable.

topology information still adversely influences the recovery results.

While the aforementioned approaches have achieved notable progress in the context of trajectory recovery, these solutions have two significant limitations. First, trajectory data, being a type of traffic data, exhibits complex spatio-temporal correlations, i.e., the previous trajectory route implies the direction of the subsequent trajectory coordinates, while the urban traffic networks further restrict possible locations in a trajectory. Existing approaches primarily consider modeling the spatio-temporal correlations within the observed trajectory data and neglecting the topological features behind the trajectory in the recovery [11], [12], [13], [14], [15]. While segment ID representation [11], [13], [14], [15] presents a more effective means of encoding road segment information compared to GPS or distinct-cell representations [12], it remains limited in its ability to account for topological characteristics among distinct numbers due to inadequate processing techniques. This limitation hinders the extraction of sufficient spatio-temporal correlations needed for accurate inferences. Fig. 1 illustrates an LSR trajectory recovery process based on an urban road network with only the origin and destination of the route provided. It is apparent that only one route (the solid black line) is valid, subject to the road network. However, since previous studies produce the shortest path instead (the red dashed line), they only consider the spatio-temporal correlations between trajectory points while ignoring the topological characteristics of urban traffic networks, rendering an invalid trajectory. In addition, although Ren et al. improved recovery performance by formulating trajectories as segment ID and moving ratio to circumvent the subsequent map-matching, the huge quantity of road segments in a city induces a dimensionality explosion, thereupon undermines the model inference performance [13], [14]. For example, the number of candidate segments in each step of inference may exceed 10 000 (see Table I for details), obstructing fast model training and online inferences. Furthermore, the accuracy of the model inference is exacerbated by the lack of consideration of the road topology characteristics, e.g., the topological and length characteristics of the road segment are essential for inferring the ID and ratio respectively.

To jointly tackle the aforementioned challenges, we propose a novel deep learning model Map-informed Adaptive Spatio-Temporal Autoencoder (MASTA), that considers the road network topology for trajectory recovery. We introduce a pre-trained attributed network embedding (ANE) [16] module to embed the road segment characteristics into the input data, thus helping the model better extract the spatio-temporal correlations within trajectories. Furthermore, inspired by the well-known Graph WaveNet [10], we propose a novel adaptive mask inference module that integrates a distance-based and an adaptive mask matrix to help segment inferences via weighting each candidate segment adaptively. The main contributions of this work are summarized as follows:

- We embed the road topology features into trajectory data via a pre-trained ANE module, allowing more road topology characteristics to be used in extracting the spatio-temporal dependencies from coarse trajectories.
- We propose a novel adaptive mask inference module that combines a distance-based mask and an adaptive mask to assist the model in making accurate road segment inferences in the recovery process.
- We conduct comprehensive case studies on two real-world trajectory datasets and compare the performance of the proposed model with state of the arts. The results show that our model consistently outperforms existing ones.

The rest of the paper is structured as follows. Sec. II reviews related research on trajectory recovery. Sec. III formalizes the trajectory recovery problem. Sec. IV gives a detailed description of the proposed model MASTA and training strategy. Sec. V presents a series of case studies on two real-world trajectory datasets and discusses the results. Finally, the paper is concluded in Sec. VI.

II. Related Work

Trajectory recovery is a crucial task in various trajectory-based applications and has been extensively studied over the years. Previous research in this area can be broadly classified into two categories: history-based and data-driven approaches.

A. Naïve History-Based Approaches

History-based approaches recover trajectories by domain knowledge-driven simple rules and statistics. An intuitive and straightforward approach to recovering an LSR trajectory between two consecutive points is to use historical data to find the Most Popular Route (MPR) between them and take it as the recovery result [17], [18], [19], [20]. As an alternative to MPR, other studies recovered LSR trajectories by searching the top-k routes in the topology [21], [22]. Subsequently, researchers attempted to incorporate transition probabilities into the recovery process. Banerjee et al. [23] utilized Gibbs sampling to learn Network Mobility Model (NMM) from a large-scale historical trajectory dataset. Similarly, Wu et al. [24] developed

a fully probabilistic approach that accounts for both temporal and spatial dependencies.

The previous approaches are limited by a strong assumption that frequent historical trajectories will appear in the future with high probability and rely on sub-optimal methods to extract complex features among traffic data, rendering inferior results than recent solutions as follows.

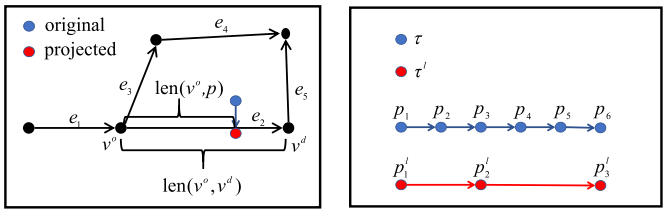
B. Deep Learning Approaches

Compared to the naïve history-based approaches, deep learning methods have shown their superiority in trajectory modeling in recent years. Recurrent Neural Networks (RNNs), together with their modern variants such as Long Short-Term Memory [25] and Gated Recurrent Unit (GRU) [26] are known for their outstanding capability of extracting dynamic features from time series and are widely used in modeling trajectory data [27], [28], [29], [30], [31]. Considering the similarity between trajectory prediction and recovery, prediction models based on extracting temporal features are applied to the recovery tasks [32], [33]. Wu et al. utilized RNN to model trajectory data and attempted to address the road network constraints for the first time in [11]. Feng et al. proposed an attentional recurrent network that captures multi-level periodicity in historical data to predict the next location [14]. However, these models often require complete historical data to support the process of extracting complex spatio-temporal features in order to make one-step predictions accurate and stable. Thus, they may not perform well in trajectory recovery with high missing rates. Tailor-made model design for feature extraction and recovery from traffic data in the presence of missing values is critical. Wang et al. applied the seq2seq model with KF to model and recover LSR trajectory data [12]. Xi et al. developed Bi-STDDP, which combines bi-directional spatio-temporal correlations and user preferences for recovery [34]. Leveraging the superior performance of attention models originating from natural language processing [35], Xia et al. adopted the fully attentional neural network to extract periodic regularities from historical data [15]. Ren et al. used a seq2seq model that simultaneously infers road segment IDs and rates, eliminated the need for subsequent map-matching processes and achieved state-of-the-art performance [13].

Despite that the aforementioned deep learning models have achieved better results than history-based approaches in trajectory recovery, they have not adequately taken into account the road topology characteristics associated with the trajectory data. Additionally, the problem of having too many candidate segments during inference, possibly leading to inaccurate recoveries, has not been fully addressed. Motivated by these research gaps, we propose a novel deep learning model MASTA based on the simple-yet-effective auto-encoder architecture for trajectory recovery considering road network topology characteristics.

III. Preliminaries

In this section, we introduce the preliminaries and the problem formulation of trajectory recovery considering the road network topology.



(a) Map-matching (b) LSR map-matched trajectory

Fig. 2. Examples of preliminaries concepts.

Definition 1 (Raw Trajectory): A raw trajectory τ^* is presented as a sequence of GPS points, denoted as $\tau^* = \{c_1, c_2, \cdots, c_n\}$, where each GPS point refers to the latitude, longitude coordinates, and the timestamp of the observation, respectively.

Definition 2 (Road Network): The urban road networks can be represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of intersections and $\mathcal{E}$ represents the set of straight road segments. It is noteworthy that curved road segments may be discretized into a series of interconnected straight sub-segments. For $e_i \in \mathcal{E}$, each road segment has an origin intersection $v^{\text{o}}_{e_i} = \langle \text{lat}^{\text{o}}_{e_i}, \text{lng}^{\text{o}}_{e_i} \rangle \in \mathcal{V}$, a destination $v^{\text{d}}_{e_i} = \langle \text{lat}^{\text{d}}_{e_i}, \text{lng}^{\text{d}}_{e_i} \rangle \in \mathcal{V}$, and additional features such as road classification, neighboring points of interest, etc.

Definition 3 (Map-matched Trajectory): To accurately determine the route taken by a trajectory, a map-matching process is required. Each point c_i in $\tau^* = \{c_1, c_2, \cdots, c_n\}$ is projected onto the road network via map-matching to produce a projected point $p_i = \langle e_i, r_i, t_i \rangle$. e_i denotes the road segment ID, and r_i refers to the moving ratio, which represents the proportion of the moving distance over the total length of segment e_i. t_i is the corresponding time step. Fig. 2a shows an example of the map-matching process, where the original blue point is projected onto its nearest segment to generate a projected red point. Particularly, $r = \text{len}(v^{\text{o}}, p)/\text{len}(v^{\text{o}}, v^{\text{d}})$, where $\text{len}(v^{\text{o}}, p)$ means the moving distance between road origin intersection v^{o} and point p, and $\text{len}(v^{\text{o}}, v^{\text{d}})$ means the total length of road. In addition, since the road sections are all straight, the position $\rho = \langle \text{lat}, \text{lng} \rangle$ of point p is fixed and calculated by the following rules:

$$\rho = v^{\text{o}} + r \times (v^{\text{d}} - v^{\text{o}}). \tag{1}$$

Definition 4 (Map-matched ϵ-Sampled Trajectory): A map-matched trajectory is considered to be ϵ-sampled if the time interval between any two consecutive points is equal to ϵ, i.e., for consecutive points $p_i = \langle e_i, r_i, t_i \rangle$ and p_{i+1}, $t_{i+1} - t_i = \epsilon, \forall i, 1 \leq i < (n-1)$.

Definition 5 (Map-matched k%-Keeping Rate Trajectory): Within a map-matched ϵ-sampled trajectory $\tau = \{p_1, p_2, \cdots, p_n\}$, arbitrary points are missing. The remaining LSR map-matched trajectory $\tau^l = \{p^l_1, p^l_2, \cdots, p^l_m\}$ has less points compared to the original trajectory τ, and its keeping rate is defined as $k\%$ if $\frac{m}{n} \times 100\% = k\%$. Consequently, the time intervals between consecutive points in τ^l are not constant and do not follow ϵ. Fig. 2b presents an example of the LSR trajectory τ^l (decorated by red color) and the

corresponding ϵ-sampled trajectory τ (blue one). In this example, the keeping rate of τ^l is $\frac{3}{6} \times 100\% = 50\%$.

Problem 1 (Trajectory Recovery): Given a LSR raw trajectory $\tau^{l*} = \{c_1, c_2, \cdots, c_m\}$, trajectory recovery aims to recover a ϵ-sampled map-matched trajectory $\tau^l = \{p_1^l, p_2^l, \cdots, p_n^l\}$. $\frac{m}{n} \times 100\% = k\%$ represents the degree to which the input data is missing, and ϵ means the time interval in the map-matched trajectory. The LSR raw trajectory τ^{l*} is first projected onto the road network using map-matching algorithms, resulting in LSR map-matched trajectory $\tau^l = \{p_1, p_2, \cdots, p_m\}$. Then, a neural network model is designed to recover the map-matched ϵ-sampled trajectory $\hat{\tau} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n\}$. The recovered trajectory $\hat{\tau}$ should resemble the original one τ, and $|\hat{\tau}| = |\tau|$.

IV. Methodology

In this section, we elaborate on the proposed MASTA. We start by presenting the overview of the proposed model. Then we introduce each constituting module and the model training method.

A. Overview

Fig. 3a illustrates the overview of the proposed MASTA. The model is based on the encoder-decoder structure, which has shown state-of-the-art performance on sequence-to-sequence tasks [36]. The urban traffic networks have a non-negligible effect on the distribution of trajectory points, e.g., the coordinates of most moving objects are on road segments. This means that the road topology characteristics is crucial for the model to extract the spatio-temporal correlations from trajectory data and then infer the missing trajectory points precisely. To achieve this, each trajectory point consisting of the road segment ID, moving ratio, and time of sample is processed by the pre-trained ANE module to embed road topology characteristics into the trajectory. After the processing of the pre-trained ANE module, the encoder module utilizes GRU layer to extract the complex temporal dependencies from LSR trajectories. The decoder module contains the pre-trained ANE, the attention mechanism, GRU, and an adaptive mask inference layer. It iteratively infers the road segment ID and moving ratio and generates a high-sampling-rate trajectory. By inferring the road segment ID and moving ratio rather than the coordinate, the model ensures that the inferred points are on road segments, thus eliminating the need for subsequent map-matching processes and minimizing the error accumulation problem on the recovery result.

B. Encoder

The urban trajectory contains complex spatio-temporal correlations, within which earlier trajectory points reflect the later trend and the road network topology further constrains the precise locations. Nevertheless, existing trajectory representation methods, namely, GPS and distinct-cell, discard the latter and undermine the recovery performance. In this paper, we employ the map-matched k%-keeping rate trajectory $\tau^l = \{p_1^l, p_2^l, \ldots, p_m^l\}$ as the model input, which utilizes the road segment ID and moving ratio to fix each point. Furthermore, we apply the pre-trained ANE module to embed road topology characteristics and make it easier for the model to extract the spatio-temporal dependencies from coarse trajectories.

1) Attributed Network Embedding: Even though combining the segment ID and moving ratio can determine the geographic coordinates of each point, the lack of topological characteristics makes it difficult for the model to extract spatio-temporal correlations and make subsequent inferences. Driven by the outstanding performance of network representation learning on various network tasks (e.g., link prediction and node classification [37], [38]), we introduce a pre-trained ANE module to integrate the road segment features into the trajectory and assist the model in extracting the complex spatio-temporal dependencies from the LSR trajectory. However, the ANE focuses on learning low-dimensional node embeddings, whereas we aim to extract topological features of road segments. It is imperative to reinterpret urban road segments as fundamental nodes while establishing the interconnections between these segments as the edges delineating the network structure. Thus, in the subsequent introduction about ANE, we shall adopt the term "node" to signify the urban road segments.

Compared to previous Network Embedding methods [39], [40], ANE generates the node embeddings through the network topology and the node attribute features. Specifically, it first utilizes the latter to construct the latent relationship among nodes and then employs the DeepWalk [39] method on the reconstructed network to generate the node embeddings. For constructing the hidden relationship, we utilize three kinds of edge features, namely, edge length, neighbor count, and highway classification (e.g., `primary`, `trunk_link`, `living_street`, etc.). Given the nodes feature $F_{\text{node}} \in \mathbb{R}^{|\mathcal{E}| \times 3}$, we utilize the following cosine similarity function to calculate the similarity between nodes:

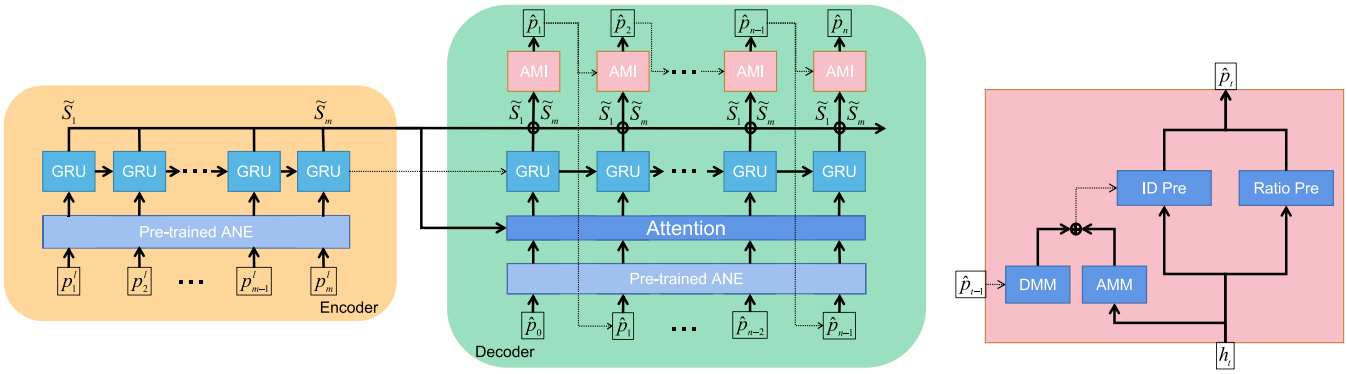
$$\cos(F_{\text{node}}^i, F_{\text{node}}^j) = 1 - \frac{||F_{\text{node}}^i - F_{\text{node}}^j||_2}{2}, \tag{2}$$

where $||\cdot||_2$ is L2 norm of a vector. In order to make the hidden relationship matrix sparse and thus to be able to distinguish the strength of the relationship between different nodes, we establish connections with the top-q[1] most similar node pairs. Consequently, we derive the sparse hidden relationship matrix $\mathcal{A}_{\text{hid}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$, containing $q|\mathcal{E}|$ 1-valued entries representing the connected nodes. Finally, based on the hidden relationship matrix $\mathcal{A}_{\text{hid}}$ and original network topology $\mathcal{A}_{\text{geo}}$, the reconstructed relationship matrix $\mathcal{A}$ is processed by the following rules:

$$\mathcal{A}^{[i,j]} = \begin{cases} 0, & \text{if both } \mathcal{A}_{\text{hid}}^{[i,j]} \text{ and } \mathcal{A}_{\text{geo}}^{[i,j]} \text{ are } 0 \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Considering that the reconstructed matrix $\mathcal{A}$ will be applied for the DeepWalk process, it is sufficient for matrix $\mathcal{A}$ to simply record whether nodes are related to each other or not (0,1). It is noteworthy that the reconstructed matrix $\mathcal{A}$ is processed by the network topology $\mathcal{A}_{\text{geo}}$ and the node attribute features F_{node}. Thus, it is inherently information-richer than $\mathcal{A}_{\text{geo}}$.

[1] In the case studies, we empirically set $q = 30$.



(a) Overview of MASTA (b) AMI

Fig. 3. The architecture of the proposed MASTA. (a) The model consists of an encoder and a decoder. The encoder consists of the pre-trained ANE and GRU module, the decoder consists of pre-trained ANE, attention, GRU, and adaptive mask inference layer. (b) The adaptive mask inference layer is composed of the ID and moving ratio inference sublayers. The ID inference layer consists of the *distance-based mask matrix* and *adaptive mask matrix* modules.

Subsequently, the DeepWalk method is applied to derive latent representations of different nodes based on the reconstructed matrix $\mathcal{A}$. It walks from each node and generates length L node sequence for T times. After the random walk process, it generates $T \times |\mathcal{E}|$ node sequences to form the training dataset. For each node sequence, we use a window of length $2w+1$ (empirically set to 10 in case studies [16]) to slide over this sequence and generate a set of training pairs Set.

For the node embedding method, the Skip-Gram model [41] is proposed. Its training objective is to learn node representations in favor of predicting the nearby nodes in a node sequence. Thus, the Skip-Gram model maximizes the following function:

$$\sum_{(e_i,e_j)\in Set} = log\,P(e_j|e_i) = \frac{exp(E_i \cdot E_j)}{\sum_{k=1}^{|\mathcal{E}|}(exp(E_i \cdot E_k))}, \tag{4}$$

where (e_i, e_j) means node pair in training dataset Set, and E denotes the dense embeddings of the corresponding node. It is clear that Eq. (4) is related to the number of nodes $|\mathcal{E}|$, so the training efficiency of Skip-Gram is unsatisfactory. In order to accelerate the training process, the Skip-Gram Negative Sampling (SGNS) [42] method is proposed, which maximizes the following function:

$$\log\sigma(E_i \cdot E_j) + \sum_{neg} \mathbf{E}_{k\sim P_D}\left[\log\sigma(-E_i \cdot E_k)\right], \tag{5}$$

where E_k is the embedding of a negative sample from the unigram distribution P_D. Maximizing Eq. (5) will make the embeddings of E_i and E_j (in one node sequence) close to each other while the embeddings of E_i and E_k away from each other. This means that the embeddings of nodes closely presented in one node sequence are similar. After the training process of the ANE module, it can be utilized to integrate the complex topological characteristics into trajectories and assist in extracting the spatio-temporal dependencies and making inferences from the original trajectory. Thus, through the pre-trained ANE layer, the road segment e_i of point p_i^l in input trajectory τ^l is first projected to $E_i \in \mathbb{R}^{1\times D}$. As mentioned in Sec. III, a projected point consists of the road segment ID e_i, the moving ratio r_i, and the corresponding time step t_i. We then get the embedding vector of point p_i^l by concatenating the moving ratio r_i and time step t_i to ANE module's output.

$$\tilde{E}_i = \text{ANE}(e_i) \oplus [r_i, t_i]. \tag{6}$$

Note that, t_i is a relative number based on the trajectory's starting point, i.e., 0,1,2,3,.... Thus, through the pre-trained ANE module, the input trajectory data $\tau^l = \{p_1^l, p_2^l, \ldots, p_m^l\} \in \mathbb{R}^{m\times D}$ which contains the information of road segment ID, moving ratio, and time step is transformed to $Emb = \{\tilde{E}_1, \tilde{E}_2, \ldots, \tilde{E}_m\} \in \mathbb{R}^{m\times(D+2)}$.

2) Gated Recurrent Unit: After integrating the road topology characteristics through the pre-trained ANE layer, we have eased the way for the model to extract the hidden spatio-temporal dependencies in trajectory data. Considering the dynamic temporal features contained in the trajectory data and the limitation of RNNs in processing long-term sequential data, we employ GRU to extract the temporal dependency [26]. By utilizing reset gate and update gate, GRU regulate the flow of information while avoiding the gradient problems. For time step t, GRU processes the input trajectory data $\tilde{E}_t$ and generates the hidden unit $\tilde{S}_t$ based on the following propagation rules:

$$\begin{aligned} u_t &= \sigma(W_z[\tilde{E}_t, \tilde{S}_{t-1}] + b_u), \\ r_t &= \sigma(W_r[\tilde{E}_t, \tilde{S}_{t-1}] + b_r), \\ c_t &= \tanh(W[\tilde{E}_t, r_t \odot \tilde{S}_{t-1}] + b_c), \\ \tilde{S}_t &= u_t \odot \tilde{S}_{t-1} + (1-u_t) \odot c_t, \end{aligned} \tag{7}$$

where u_t and r_t are the update gate and reset gate, and $W \in \mathbb{R}^{(D+2+F)\times F}$ and $b \in \mathbb{R}^{1\times F}$ are trainable model parameters. $\odot$ denotes element-wise multiplication. After Emb goes through L layers of GRU module, we get a hidden state sequence $\tilde{S} = \{\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_m\} \in \mathbb{R}^{m\times F}$. By sequentially processing the data, the GRU model is capable of extracting the temporal dependency from the trajectory representation Emb, and the last hidden state $\tilde{S}$ is subsequently passed to the decoder module.

C. Decoder

Similar to the encoder model, the decoder module also contains the pre-trained ANE module, which first projects the inferred trajectory point of the previous step $\hat{p}_{i-1} = \langle \hat{e}_{i-1}, \hat{r}_{i-1}, t_{i-1} \rangle$ to the dense representation $\hat{E}_{i-1} \in \mathbb{R}^{1\times(D+2)}$. To initialize the decoder, we set the $\hat{E}_0 = 0$.

1) Attention: In addition to the dynamic regularities presentation, the trajectory also exhibits complex spatial correlations, where each coordinate is influenced by other points with varying weights. These weights are highly dynamic and depend on the geographical location and time step of each point. To capture these dynamic relationships between trajectory points, we draw inspiration from the attention mechanism in natural language translation tasks [35], [43]. We employ attention to learn the changing correlations between the different trajectory points over time. Specifically, the attention mechanism assigns different weights to each point in the LSR trajectory based on the previously inferred coordinate, generating a dense representation that captures the corresponding spatial correlations and facilitates the inference of the next coordinate. Given the dense representation $\hat{E}_{i-1}$ and the output state of the encoder module $\tilde{S}$, the weighted sum is calculated as follows:

$$a_{i-1} = \sum_{j=1}^{m} \alpha_{i-1,j} \tilde{S}_j, \tag{8a}$$

$$\alpha_{i-1,j} = \frac{\exp(u_{i-1,j})}{\sum_{k=1}^{m} \exp(u_{i-1,k})}, \tag{8b}$$

$$u_{i-1,j} = v^T \tanh(W_E \hat{E}_{i-1} + W_S \tilde{S}_j), \tag{8c}$$

where $\alpha_{i-1,j}$ is the attention score between dense representation $\hat{E}_{i-1}$ and the encoder output $\tilde{S}_j$, and $\sum_{j=1}^{m} \alpha_{i-1,j} = 1$. Symbol $u_{i-1,j}$ is the similarity between $\hat{E}_{i-1}$ and $\tilde{S}_j$. Further, $W_E \in \mathbb{R}^{(D+2)\times F_A}$, $W_S \in \mathbb{R}^{F\times F_A}$, and $v \in \mathbb{R}^{F_A\times 1}$ are weight matrices. The attention mechanism can model the highly dynamic weights at each point, which is an effective tool for analyzing the complex spatial correlations in trajectory data. Besides, since we introduce the pre-trained ANE module to integrate topological characteristics into the trajectory data, it enable the attention mechanism to extract more meaningful hidden spatial correlations. Finally, we combine the attention-weighted sum a_{i-1} with the previous inferred coordinate representation $\hat{E}_{i-1}$ to get the output of the attention mechanism: $A_{i-1} = [a_{i-1}, \hat{E}_{i-1}] \in \mathbb{R}^{1\times(F+D+2)}$.

Next, the output of attention mechanism A_{i-1} is processed by the L layers of GRU module to extract the temporal dependency. The hidden state is updated as follows:

$$S_i = \mathrm{GRU}(A_{i-1}, S_{i-1}), \tag{9}$$

where GRU represents the process of Eq. (7). To initialize the GRU module in the decoder, we set the $S_0 = \tilde{S}_m$.

2) Adaptive Mask Inference: Compared with predicting the GPS coordinates, inferring discrete segment ID and moving ratio can reduce the complexity of model training and ensure that the inferred points are on the road network to omit the subsequent map-matching process. Nevertheless, inferring the road segment IDs where the missing point locates is also tricky because there is typically a large number of candidate road segments in a city. To tackle this problem, we propose a novel adaptive mask inference module (AMI) consisting of a distance-based mask layer and an adaptive mask layer to assist the model in making road segment inferences.

For the time interval ϵ, the moving distance of an object is in the range of $[0, V_{\max} \times \epsilon]$. Thus, to infer the segment IDs of time step t on the trajectory τ^l, we can utilize the inferred point $\hat{p}_{t-1}$ of the previous time step and the movable range threshold to build a distance-based mask matrix so as to narrow the range of candidate segments. Since the movement of an object is uncertain and is affected by many factors (traffic congestion, weather conditions, etc. [44]) we can only build a coarse-grained mask matrix based on distance. Furthermore, since RNNs have the problem of error accumulation in processing sequence data [26] (the inference of the previous trajectory point is incorrect), the distance threshold shan't be too small. Thus, given the inferred point of the previous time step $\hat{p}_{t-1} = \langle \hat{e}_{t-1}, \hat{r}_{t-1}, t-1 \rangle$, the *distance-based mask matrix* (DMM) $M^t_{\mathrm{dis}} \in \mathbb{R}^{|\mathcal{E}|\times 1}$ is generated based on the following rules:

$$M^t_{\mathrm{dis}}[i] = \begin{cases} 1, & \text{if } \mathrm{len}(\hat{p}_{t-1}, e_i) \le \gamma \\ 0, & \text{if } \mathrm{len}(\hat{p}_{t-1}, e_i) > \gamma, \end{cases} \tag{10}$$

where $\gamma = V_{\max} \times \epsilon$ is the distance threshold to narrow the range of candidate segments. $\mathrm{len}(\hat{p}_{t-1}, e_i)$ denotes the shortest path distance between $\hat{p}_{t-1}$ and e_i on the urban network, which can be calculated by Dijkstra algorithm [45].

Although DMM makes restrictions on the candidate road segments, its coarse-grained nature may result in a relatively weak function during the inferring process. To address this limitation, we propose an *adaptive mask matrix* (AMM) $M^t_{\mathrm{adp}} \in \mathbb{R}^{|\mathcal{E}|\times 1}$. Building upon the superior performance of the self-adaptive adjacency matrix in Graph Neural Networks that learn hidden spatial dependencies to extract comprehensive spatial correlations [10], [46], AMM assigns heterogeneous weights to candidate road segments. This approach enhances the segment inference process by offering more nuanced and context-specific weighting to better capture the underlying spatial relationships between road segments. Considering the current feature information and the overall correlations of the trajectory are crucial to constructing the AMM [15], [47], we first concatenate the hidden state of GRU module S_t, the first and last GRU output of encoder module $\tilde{S}_1$, $\tilde{S}_m$ to get the input vector $h_t = [S_t, \tilde{S}_1, \tilde{S}_m] \in \mathbb{R}^{1\times 3F}$. Furthermore, the purpose of the model to infer segment ID and moving ratio makes the road network characteristics non-negligible. Thus, given the trajectory information h_t and the urban road segment feature set $E_{\mathrm{set}} = \{E_0, E_1, \ldots, E_{|\mathcal{E}|-1}\} \in \mathbb{R}^{|\mathcal{E}|\times D}$, the M^t_{adp} is generated by the following rule:

$$M^t_{\mathrm{adp}} = \sigma((E_{\mathrm{set}} W_E) \times (h_t W_h)^T), \tag{11}$$

where $W_h \in \mathbb{R}^{3F\times C}$ and $W_E \in \mathbb{R}^{D\times C}$ are learnable parameters, and $\sigma(\cdot)$ is the sigmoid activation function to restrict the weight within $(0, 1)$.

Compared to DMM, AMM provides finer-grained differentiation on candidate road segments for the inference process. Nevertheless, DMM is also indispensable because inaccurate

AMM may have an adverse impact on the corresponding inference process (e.g., the next trajectory point is on segment e_i, but $M^t_{\text{adp}}[i]$ is incorrectly set to 0). This issue is particularly common in the early stages of model training when the correct mapping between trajectory spatio-temporal features and the appropriate AMM has not yet been learned. Therefore, considering the model training stability, we integrate M^t_{dis} and M^t_{adp} masks together to generate the final *mask matrix* M^t as follows:

$$M^t = \beta M^t_{\text{dis}} + (1-\beta) M^t_{\text{adp}}, \tag{12}$$

where β (set to 0.5) is a tuneable parameter to control the information weight from the DMM and AMM.

Finally, based on the *mask matrix* $M^t \in \mathbb{R}^{|\mathcal{E}|\times 1}$ and the hidden state S_t, the road segment inference is generated by the fully connected (FC) layer with softmax activation as follows:

$$P(j) = \frac{\exp(S_t \cdot W_j + B_j) \odot M^t_j}{\sum_{k=1}^{|\mathcal{E}|} \exp(S_t \cdot W_k + B_k) \odot M^t_k}, \tag{13}$$

where $P(j)$ means the probability that the next trajectory point is located on road e_j, and $W \in \mathbb{R}^{F\times|\mathcal{E}|}$ and $B \in \mathbb{R}^{1\times|\mathcal{E}|}$ are learnable parameters. M^t is integrated into the softmax function to assist the model in making inferences by narrowing the candidate set and weighting each candidate segment adaptively. This approach improves the precision and reliability of the inferred road segment ID, which is determined by selecting the highest probability P among all candidates. The inferred moving ratio $\hat{r}_t$ is processed by the following rule:

$$\hat{r}_t = \sigma(W_r S_t + b_r), \tag{14}$$

where $W_r \in \mathbb{R}^{F\times 1}$ and b_r are weights, and $\sigma(\cdot)$ is the sigmoid function.

3) Training Scheme: By recursively processing the inference n times, we can generate the final recovered map-matched ϵ-sampling rate trajectory $\hat{\tau} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n\}$, where $\hat{p}_i$ contains the inferred road segment $\hat{e}_i$ and corresponding moving ratio $\hat{r}_i$. Because we infer the road segment ID and moving ratio simultaneously, the model is trained with the following losses:

$$L = L_e + \eta L_r, \tag{15a}$$

$$L_e = -\sum_{i=1}^{n} e_i \log(P(\hat{e}_i)), \tag{15b}$$

$$L_r = \sum_{i=1}^{n} ||r_i - \hat{r}_i||^2, \tag{15c}$$

where L is the loss function which consists of the road segment ID inference loss L_e and moving ratio inference loss L_r. Coefficient η (empirically set to 10 in case studies [13]) controls the weighting of these two components. L_e is the cross entropy loss function to minimize the difference between the inferred road segment ID and the real one. L_r is the mean squared error to minimize the prediction error on the inferred moving ratio.

TABLE I
DATASET STATISTICS

Dataset	Region	#Trajectories	#Edges
CD	(30.659 104.042) (30.697, 104.110)	264948	6122
XA	(34.207, 108.907) (34.282, 108.994)	255129	5009

V. EXPERIMENT

In this section, a series of comprehensive case studies are conducted on two datasets to evaluate the trajectory recovery performance of MASTA. We employ four evaluation metrics to validate the model's performance under different keeping rates and time intervals and compare the results with state-of-the-art baselines. Additionally, we conduct an ablation test to verify the effectiveness of each sub-module. Furthermore, we analyze the sensitivity of the model performance to the choice of different hyperparameters. Finally, we present the visualization results of MASTA and compare them with a state of the art.

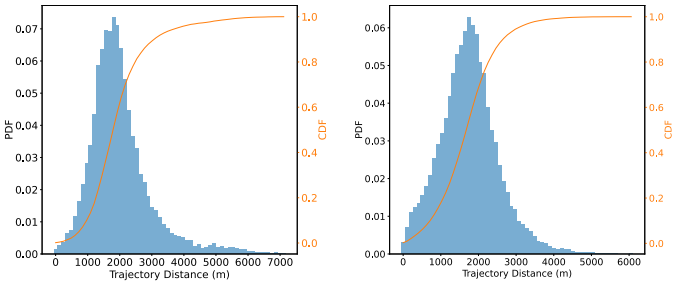
A. Dataset and Configurations

We evaluate the performance of MASTA on two real-world taxi trajectory datasets collected in two major representative cities of China, namely, Chengdu (**CD**) and Xi'an (**XA**). The datasets are provided by the Didi Chuxing GAIA Project[2] and are collected from October 1st to October 31st, 2016 and from November 1st to November 30th, 2016, respectively. The corresponding road network topology is provided by OpenStreetMap.[3] The statistics of these two datasets are presented in Table I. The amount of data is sufficient for subsequent experiments. In addition, as megacities in China, Chengdu and Xi'an have complex urban transportation structures. Thus, the types of related trajectory data are diverse. The sampling interval of GPS points in trajectories is 3 s. However, due to energy consumption concerns, mainstream GPS-based applications do not collect location information at such high frequencies. The sampling interval of car coordinates is typically between 15 and 30 seconds (e.g., taxi trajectory dataset Porto dataset[4]). In addition, a short sampling interval leads to reduced uncertainty between trajectory points, reducing the difficulty of the trajectory imputation task. It makes high-performance imputation models unimpressive in terms of metrics. To address these issues and make the model meaningful, we have resampled the trajectory data and increased the sampling interval to 30 s. This adjustment provides a wider range over which trajectories can be moved and also makes trajectory recovery more realistic. After resampling the trajectory data, any short trajectories with less than 10 GPS points and those with missing points are removed. The remaining trajectory data is processed by the fast map-matching (FMM) algorithm [48] to generate the ground truth map-matched ϵ-sampled trajectories with $\epsilon = 30$ s. Due to discrepancies between the road network data and trajectory information on OSM, trajectories that failed

[2]These datasets can be downloaded (following approval of access request) at https://outreach.didichuxing.com/

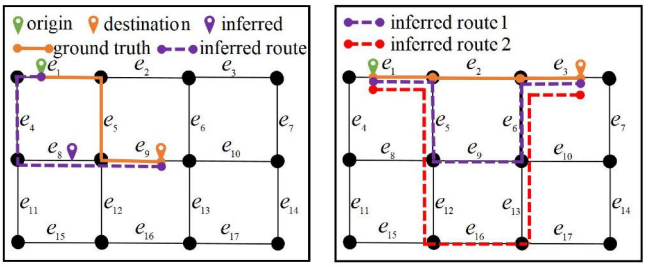
[3]https://www.openstreetmap.org/

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



(a) **CD** dataset (b) **XA** dataset

Fig. 4. Probability distribution function (PDF) and cumulative distribution function (CDF) of trajectory distance.



(a) Example of trajectory route (b) Illustration of *Hit Rate*

Fig. 5. Examples of evaluation metrics.

during map-matching were excluded prior to training. For the remaining trajectories, the FMM produced high-quality matches, with the average distance from each original point to its corresponding matched point being 3.10 m and 3.54 m for CD and XA, respectively. These lane-level errors are negligible for the purposes of this study. For the processed trajectory data, Fig. 4a and 4b shows the Probability distribution function (PDF) and Cumulative distribution function (CDF) of trajectory distance. The distribution of trajectory distances in the **CD** and **XA** datasets is similar. The trajectory distance ranges of **CD** and **XA** datasets are [0, 7113] and [0, 6102], respectively. Most trajectories are short-distance trajectories with distances between 1000 and 3000 meters.

Based on the 30 s-sampling-rate trajectory data, the LSR trajectory data is generated by randomly keeping $k\%$ of the original data. In this study, we investigate the performance of the model under different keeping rates, namely, $k \in \{6.25, 12.5, 25\}$ following the practice in [13]. It means that the average time interval of the generated LSR trajectory is 8 min, 4 min, and 2 min, respectively. For cross-validation, each dataset is split into non-overlapping training, validation, and testing subsets with a ratio of $3:1:1$.

All case studies are conducted on a server with an nVidia GTX 2080 Ti GPU. The default values for the hyperparameters of MASTA are as follows. Dimension of the ANE module $D = 16$. Output dimension of the GRU module and attention mechanism $F = 256$. Number of GRU layers in the encoder and decoder module $L = 2$, and the number of hidden layer neurons of the AMP module in Eq. (11) is set to $C = 128$. To infer the DMM with trajectory points of the previous time step, we set $V_{\max} = 120\,\text{km/h}$, which is large enough for the traffic speed in the city. The Adam optimizer [49] with an initial learning rate of 0.001 is used for training, which decays by 0.7 every five epochs. The batch size is 128. Early stopping with a patience of 10 is applied to prevent overfitting.

Given the recovered high-sampling-rate trajectory $\hat{\tau} = \{\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_n\}$ and the ground truth τ, we adopt the Recall ($\frac{|\hat{\tau} \cap \tau|}{|\tau|}$) and the Precision ($\frac{|\hat{\tau} \cap \tau|}{|\hat{\tau}|}$) as the recovery performance metrices for each inferred road segment ID. Besides, since our goal is to recover the LSR trajectory points as well as to ensure that the trajectory route inferred from the recovered trajectory points aligns with the ground truth, we also consider the accuracy of the inferred trajectory route. In particular, we use ϕ to denote the set of all segments through which the trajectory τ traverses. For the recovered trajectory $\hat{\tau}$, we utilize the Dijkstra algorithm to calculate the list of road segment IDs required to traverse all points, represented by $\hat{\phi}$. The *Hit Rate* indicating the level of consistency between the inferred trajectory route and the real one is therefore calculated as follows:

$$\text{Hit Rate} = \frac{|\hat{\phi} \cap \phi|}{|\phi|}. \tag{16}$$

Fig. 5a presents an illustration of this metric. Based on the inferred (purple) points, we can get the trajectory route (purple) which passes through edges $\{e_1, e_4, e_8, e_9\}$. Considering the ground truth route (orange) on $\phi = \{e_1, e_5, e_9\}$, Hit Rate = 66.6%. Nevertheless, the Hit Rate does not fully indicate how the recovered trajectory $\hat{\tau}$ aligns with the ground truth. As shown in Fig. 5b, the Hit Rate of the inferred route 1 (purple) and route 2 (red) are both 66.6%, but it is evident that route 1 is closer to the ground truth. Therefore, we also calculate the Length Difference between the recovered route segment list $\hat{\phi}$ and label ϕ to assist the metric Hit Rate in evaluating the performance. Length Difference is calculated as follows:

$$\text{Length Difference} = |\text{len}(\hat{\phi}) - \text{len}(\phi)|, \tag{17}$$

where $len(\phi)$ means the total distance that traversed by ϕ. Thus, evaluation metrics Hit Rate and Length Difference are required to work together to measure the performance of the model (a higher value of Hit Rate and a lower value of Length Difference represent better recovery performance). In conclusion, Recall and Precision are primarily oriented towards assessing the inferred segments at each trajectory point. Conversely, the metrics of Hit Rate and Length Difference take a broader perspective by evaluating the segment information across the entirety of the inferred trajectory.

B. Recovery Performance

In this section, we compare MASTA with several trajectory recovery baselines. For those methods that infer trajectory points in plane coordinates, we splice the FMM method to ensure that the trajectory points are aligned with the road network.

- Linear [8] + FMM [48]: This baseline approach recovers the trajectory by linear interpolation. The inferred points are then projected onto the road networks using FMM.

- TrImpute [50] + FMM [48]: This baseline approach generates several candidate points based on historical trajectory data to fill in the gaps in the LSR path, without using any prior traffic network information. The inferred points are then projected onto the road networks using FMM.
- LPIRNN [11]: This baseline approach utilizes two RNN models to predict the destination's road segment by modeling trajectory data and addressing the network structure constraints, respectively. The missing points in the trajectory data are inferred consecutively using this approach.
- DHTR [12] + FMM [48]: This baseline approach integrates the sequence-to-sequence model with the Kalman filter to recover the LSR trajectory in plane coordinates. The inferred points are then projected onto the road networks using FMM.
- DeepMove [14]: This baseline approach proposes an attentional RNN model that takes into account multiple factors. It infers each missing point autoregressively.
- MTrajRec [13]: This baseline approach uses a multi-task learning model based on sequence-to-sequence structure to recover LSR trajectories while simultaneously inferring the corresponding road segment ID and moving ratio.
- AttnMove [15]: This baseline approach introduces a novel attentional neural network model, which is capable of modeling trajectory data and extracting periodic patterns from individual historical trajectories.
- TrajBERT [51] + FMM [48]: This baseline introduces a novel BERT-based neural network model, which focuses on implicit trajectories. The inferred points are then projected onto the road networks using FMM.

Table II presents a comparison of MASTA with selected baselines on the **CD** and **XA** datasets. To begin with, the MASTA outperforms all baselines for all keeping rates $k \in \{6.25, 12.5, 25\}$ and shows the most significant improvement in recovery performance for $k = 6.25$. Specifically, for CD with $k = 6.25$, the MASTA outperforms the best-performing baseline (MTrajRec) by 11.6%, 9.15%, and 6.83% on Recall, Precision, and Hit Rate, respectively. Furthermore, the proposed model reduces the Length Difference evaluation metric by 69.5%. The superior performance of the proposed model can be attributed to three factors: (i) the autoencoder structure and its sub-modules capture the dynamic temporal dependency in the trajectory data; (ii) the pre-trained ANE module incorporates road topology characteristics into the trajectory data and an attention mechanism extracts spatial correlations; and (iii) the novel adaptive mask inference module facilitates adaptive segment inference by avoiding an excessive number of candidate segments.

Additionally, it is worth noting that the performance of all methods on dataset **CD** is inferior to that on dataset **XA**, suggesting that the network topology of **CD** is more complex than that of **XA**, which is consistent with the number of edges presented in Table I. For different trajectory distances, the model's performance decreases slightly by about 9% on the long-distance (larger than 3000 m) trajectory recovery compared to the short-distance trajectories, but remains stable. The

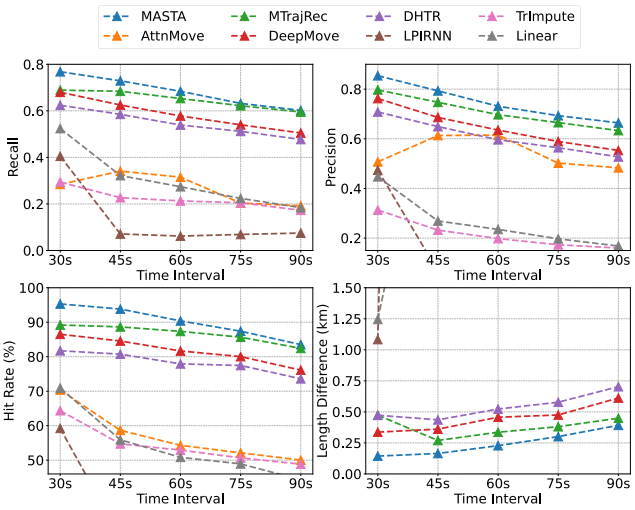


Fig. 6. Performance comparison of different approaches on different time intervals.

rule-based approaches, including Linear and TrImpute, exhibit unsatisfactory recovery performance due to their strong prior assumptions and inability to capture the dynamic trend of the trajectory. Notably, TrImpute shows an abnormal behavior in which all four metrics increase as the keeping rate rises, contrary to the expected decrease in the Length Difference metric. Among the deep learning methods, AttnMove suffers from overfitting problems, potentially due to the inclusion of personal preference features in the ride-trajectories. TrajBERT's trajectory recovery results are not satisfactory, probably due to its focus on implicit trajectory recovery. LPIRNN's recovery performance is inferior, as it solely relies on the RNN module to process the trajectory data, making it vulnerable to missing data. In contrast, DHTR, DeepMove, and MTrajRec exhibit stable and satisfactory performance. DHTR's performance is limited due to its use of discrete grid representation, which introduces noise into the trajectory data. On the other hand, MTrajRec achieves the best results by simultaneously inferring segment ID and ratio, effectively avoiding the problem of error accumulation.

In addition to different keeping rates, we also investigate the model sensitivity at different time intervals. We select a wide range of time intervals from 30 s to 90 s with a 15 s step. Fig. 6 shows the simulation result on **CD** dataset with 12.5% keeping rate. It is clear that MASTA outperforms all baselines at every time interval, which demonstrates the generality and robustness of the proposed model. The performance gap shrinks with the increasing time interval, which suggests that the trajectory inference process becomes more challenging with the increased time interval and heightened uncertainty.

For the missing types of trajectory data, we also try to explore the MASTA's imputation performance under the non-random missing type (where the missingness of the data is continuous). Fig. 7 shows the simulation result on the **XA** dataset with 12.5% keeping rate. In order to make the results clear, we have chosen to present evaluation matrices Hit Rate and Length Difference. We remove AttnMove by default due to its overfitting problem on the dataset **XA**.

TABLE II

PERFORMANCE COMPARISON WITH DIFFERENT KEEPING RATE (6.25% / 12.5% / 25%) ON **CD** AND **XA** DATASETS WITH RECALL, PRECISION, HIT RATE, AND LENGTH DIFFERENCE. HIT RATE IS REPORTED IN PERCENTAGE (%) AND LENGTH DIFFERENCE IN (km)

Dateset	Model	Keeping Rate (%)		
		6.25	12.5	25
CD	Linear	0.367 / 0.301 / 57.08 / 1.680	0.524 / 0.447 / 70.93 / 1.245	0.620 / 0.548 / 78.06 / 0.945
	TrImpute	0.233 / 0.262 / 50.85 / 2.168	0.294 / 0.313 / 64.35 / 2.414	0.348 / 0.313 / 69.94 / 3.170
	LPIRNN	0.361 / 0.399 / 54.82 / 1.218	0.405 / 0.474 / 59.23 / 1.083	0.471 / 0.529 / 62.29 / 0.980
	DHTR	0.574 / 0.680 / 78.63 / 0.686	0.624 / 0.708 / 81.73 / 0.503	0.659 / 0.723 / 92.48 / 0.473
	DeepMove	0.627 / 0.737 / 83.20 / 0.483	0.680 / 0.762 / 86.50 / 0.337	0.718 / 0.774 / 88.80 / 0.314
	MTrajRec	0.647 / 0.783 / 86.51 / 0.527	0.689 / 0.797 / 89.20 / 0.473	0.760 / 0.822 / 92.48 / 0.358
	AttnMove	0.148 / 0.499 / 61.54 / 9.474	0.284 / 0.507 / 70.35 / 18.901	0.421 / 0.534 / 80.75 / 13.618
	TrajBERT	0.372 / 0.420 / 58.36 / 1.204	0.435 / 0.482 / 64.03 / 0.985	0.467 / 0.633 / 63.26 / 0.855
	MASTA	**0.718 / 0.840 / 92.16 / 0.195**	**0.769 / 0.870 / 95.30 / 0.144**	**0.826 / 0.884 / 96.90 / 0.104**
XA	Linear	0.470 / 0.396 / 62.35 / 1.436	0.593 / 0.503 / 79.62 / 0.895	0.685 / 0.652 / 84.22 / 0.825
	TrImpute	0.303 / 0.331 / 58.25 / 2.020	0.375 / 0.389 / 71.61 / 2.146	0.435 / 0.384 / 76.40 / 2.797
	LPIRNN	0.432 / 0.512 / 57.69 / 1.013	0.492 / 0.543 / 77.63 / 0.773	0.583 / 0.642 / 80.22 / 0.702
	DHTR	0.629 / 0.731 / 82.95 / 0.621	0.674 / 0.762 / 84.53 / 0.433	0.709 / 0.774 / 92.75 / 0.353
	DeepMove	0.672 / 0.780 / 86.01 / 0.480	0.714 / 0.802 / 87.18 / 0.318	0.745 / 0.805 / 89.68 / 0.285
	MTrajRec	0.626 / 0.722 / 90.45 / 0.272	0.781 / 0.848 / 93.25 / 0.200	0.816 / 0.859 / 94.67 / 0.167
	AttnMove	0.162 / 0.499 / 67.13 / 11.820	0.303 / 0.496 / 75.62 / 22.213	0.436 / 0.502 / 80.82 / 15.824
	TrajBERT	0.464 / 0.598 / 66.23 / 0.824	0.506 / 0.572 / 79.66 / 0.785	0.626 / 0.696 / 82.26 / 0.695
	MASTA	**0.753 / 0.868 / 94.59 / 0.168**	**0.800 / 0.887 / 96.46 / 0.123**	**0.846 / 0.904 / 97.70 / 0.118**

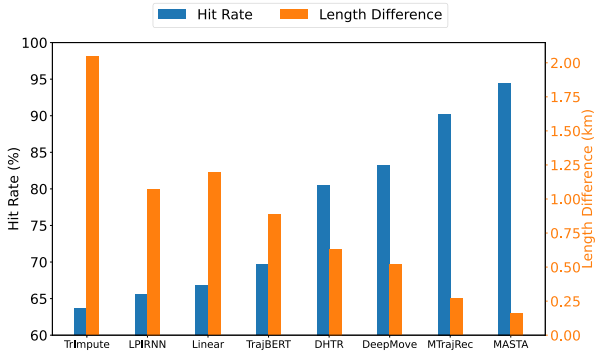


Fig. 7. Performance comparison of different approaches on **XA** dataset with non-random missing type, 12.5% keeping rate and 30 s time-interval.

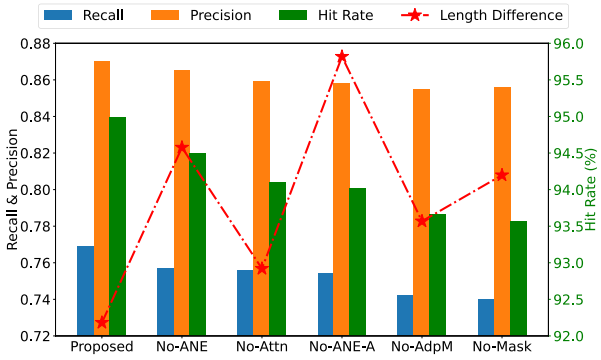


Fig. 8. Performance comparison of ablation models.

Clearly, the MASTA model outperforms all baselines under the non-random missing type, demonstrating the proposed model's robustness. The model's imputation performance with non-random missing types is slightly degraded compared to random missing. The reason is that non-random missing is more complex than random, making it more difficult for the model to extract spatio-temporal correlations in low-sampled trajectories.

C. Ablation Test

In order to investigate the contribution of each sub-module to the overall recovery performance of MASTA, we carry out comprehensive ablation tests with the following ablated variants:

- No-ANE: The pre-trained ANE module is removed.
- No-Attn: The attention mechanism in the decoder module is replaced by a linear layer.
- No-ANE-A: Both the pre-trained ANE module and attention mechanism are removed.
- No-AdpM: The *adaptive mask matrix* M_{adp} in the AMI layer is removed.
- No-Mask: Both the *distance-based mask matrix* M_{dis} and *adaptive mask matrix* M_{adp} in the AMI layer are removed. The *mask matrix* M is filled with 1.

All variants are trained and tested on **CD** dataset with a 12.5% keeping rate. The simulation results are shown in Fig. 8. Overall, the proposed MASTA demonstrated superior performance compared to the other variations, which confirms the contribution of each sub-module to the overall performance.

Out of the variations, the AMI layer shows the most significant contribution to the recovery performance. This highlights the crucial role of AMI in narrowing down the range of candidate segments. By comparing models No-AdpM and No-Mask, we can conclude that only utilizing M_{dis} has a minor effect on the recovery performance, which is consistent with its function of providing a coarse-grained restriction. Additionally, the No-ANE module shows a significant drop in Length Difference. This can be attributed to the removal of the pre-trained ANE module, which makes it challenging for the model to incorporate road characteristics. As a result, the inference process, particularly the moving ratio inference, is affected. Moreover, the recovery outcomes of

TABLE III
PERFORMANCE OF MASTA WITH FOUR HYPERPARAMETERS ON **CD** DATASET. HIT RATE IS REPORTED IN PERCENTAGE (%) AND LENGTH DIFFERENCE IN (km)

Hyperparameter	Recall	Precision	Hit Rate	Length Difference
D=8	0.751	0.862	94.18	0.263
D=16 (default)	0.769	**0.870**	**95.30**	**0.144**
D=32	**0.776**	0.870	95.28	0.149
D=64	0.774	0.868	95.19	0.145
F=64	0.772	0.865	93.51	0.257
F=128	0.769	0.870	94.98	0.170
F=256 (default)	0.769	**0.870**	95.30	**0.144**
F=512	**0.777**	0.868	**95.49**	0.156
L=1	**0.770**	**0.875**	94.20	0.203
L=2 (default)	0.769	0.870	**95.30**	**0.144**
L=3	0.760	0.853	94.11	0.183
L=4	0.762	0.854	94.23	0.188
C=32	0.774	0.862	95.10	0.172
C=64	0.777	0.865	95.19	0.154
C=128 (default)	0.769	**0.870**	95.30	**0.144**
C=256	**0.778**	0.864	**95.39**	0.145
β=0	0.742	0.855	93.66	0.217
β=0.25	0.756	0.862	94.19	0.163
β=0.5 (default)	0.769	**0.870**	95.30	**0.144**
β=0.75	0.771	0.863	**95.33**	0.152
β=1	**0.774**	0.869	95.26	0.148
q=0	0.752	0.859	93.93	0.206
q=15	0.760	0.865	94.73	0.170
q=30 (default)	**0.769**	**0.870**	**95.30**	**0.144**
q=60	0.762	0.860	94.98	0.167

variants No-Attn and No-ANE-A not only emphasize the critical role of spatial feature extraction in trajectory recovery but also demonstrate the contribution of the road topology characteristics integrated by the pre-trained ANE module in the extraction process.

D. Hyperparameter Test

The hyperparameters setting of the proposed model is critical to the recovery performance. In this section, we investigate the sensitivity of six parameters of MASTA, namely, the dimension of ANE module D, the GRU output dimension F, the number of GRU layers L, the hidden dimension of AMI module in Eq. (11) C, the information weight β in Eq. (12), and the number of neighboring nodes q of $\mathcal{A}_{\text{hid}}$ in this section. We set $D \in \{8, 16, 32, 64\}$, $F \in \{64, 128, 256, 512\}$, $L \in \{1, 2, 3, 4\}$, $C \in \{32, 64, 128, 256\}$, $\beta \in \{0, 0.25, 0.5, 0.75, 1\}$, $q \in \{1, 15, 30, 60\}$ for simulation on **CD** dataset with a 12.5% keeping rate.

Table III shows the simulation results. In terms of performance changes, all hyperparameters have the same characteristics: the model's recovery performance increases significantly with an increase in parameter values and the model capacity. Nevertheless, the excessive parameter search space caused by overly large parameter values can lead to the overfitting issue [52]. Taking the hyperparameter D as an example, the model's recovery performance improves notably as the parameter D increases from 8 to 16. However, the model's performance decreases slightly as D increases from 16 to 64. This phenomenon is consistent with the information on the number of candidate segments in the dataset we used. Since the number of candidate roads in **CD** and **XA** datasets does not exceed 2^{16}, it is sufficient to use a 16-dimensional vector to distinguish different roads. The hyperparameter β controls the information weight from the DMM and AMM in Eq. (12). When the value of β is small ($\beta = 0, 0.25$), the model does not fully utilize the critical role that AMM can provide with fine-grained differentiation on candidate road segments, making the model perform poorly. When β is too large ($\beta = 0.75, 1$), it causes the model to ignore the vital role that DMM can play in stabilizing the training process. This results in a longer training process (the average number of training iterations increases from 26 to 42) for the model with negligible improvement in recovery performance. The hyperparameter q controls the sparsity of the hidden adjacency matrix $\mathcal{A}_{\text{hid}}$, which affects whether the process of generating node embedding can take into account knowledge of node attributes. When $q = 0$, the Attribute Network Embedding process degenerates to the Network Embedding process, resulting in poor performance. When q is too large ($q = 60$), the edges of the matrix $\mathcal{A}_{\text{hid}}$ are too dense, which makes it difficult to distinguish the strength of the relationship between nodes and affects the model performance.

Furthermore, we observe that the evaluation metrics Recall and Hit Rate perform inconsistently in some cases. Compared with the performance of $L = 1$, $L = 2$ is better in metrics Hit Rate but worse in Recall. This is due to the fact that the focus of these evaluation metrics is different. Recall and Precision focus on the recovery of each point, while Hit Rate and Length Difference focus on the accuracy of the entire recovered trajectory. Since the accuracy of the whole trajectory is more vital than the precision of each trajectory point in this task, we choose 2 as the default parameter of L instead of 1.

E. Qualitative Analysis

To better investigate the recovery performance of MASTA, we conduct a visualization case study in this section. Fig. 9 shows the visualization of recovered trajectories of MASTA and MTrajRec on the **XA** and **CD** dataset with a 12.5% keeping rate and 30 s time interval. The black points and curve in Fig. 9a represent LSR coordinates and the real trajectory route, respectively. The grey points in Fig. 9a denote missing points in the 30 s time interval trajectory. The green points in Fig. 9b show the recovery results of MASTA. It can be seen that the trajectory points inferred by MASTA are all on the correct road segments, ensuring that the trajectory route generated by the inferred coordinates is consistent with the ground truth. It demonstrates the superior performance in extracting the complex spatio-temporal correlations in the trajectory data and making precise coordinate inferences of MASTA. In contrast, Fig. 9c illustrates the recovery result of MTrajRec, which infers the road segment ID and ratio simultaneously to eliminate error accumulation. One of the trajectory points (the seventh point with red color) is incorrectly inferred in this sample. While the inferred road segment is geographically close to the ground truth, the trajectory must bypass a redundant circle to correct this mistake and return to the

(a) Ground truth on **XA** (b) MASTA on **XA** (c) MTrajRec on **XA**

(d) Ground truth on **CD** (e) MASTA on **CD** (f) MTrajRec on **CD**

Fig. 9. Visualization of recovered trajectories of MASTA and MTrajRec on the **XA** and **CD** dataset with 12.5% keeping rate and 30 s time-interval.

TABLE IV
COMPARISON OF MODEL SIZE AND EPOCH TRAINING TIME WITH DIFFERENT KEEPING RATE

Model	# Parameters (M)	Epoch time (min) with different keeping rate		
		0.0625%	0.125%	0.25%
DHTR	7.95	102	99	99
DeepMove	7.42	103	101	97
MTrajRec	7.35	69	67	67
MASTA	11.6	101	99	95

correct road segment due to the limitations of the urban road network. The reason for this phenomenon is that MTrajRec considers the characteristics and relationships of segments incompletely, making the inference of road segments more complex, which affects the recovery performance. In addition, Fig. 9d to 9f show the visualization of recovered trajectories on the **CD** dataset. Similarly, Fig. 9f shows that the MTrajRec infer incorrectly for the tenth missing trajectory point because of the lack of consideration of road segment characteristics.

F. Computational Cost

To investigate the computational efficiency of the proposed MASTA, we present a comparison of the model size and epoch training time for different keeping rates across four models with competitive recovery performance: DHTR, DeepMove, MTrajRec, and MASTA. Table IV shows that while MASTA has the highest number of parameters (11.6 million), its epoch training time is competitive, particularly at higher keeping rates. For example, at a 0.25% keeping rate, MASTA is faster than both DHTR and DeepMove, despite having more parameters. This observation demonstrates that the adaptive mask inference mechanism in MASTA effectively manages computational cost with selecting candidate road segments. In contrast, MTrajRec has the fastest training speed, but this comes at the cost of a smaller model with less recovery accuracy compared to MASTA. These results suggest that MASTA strikes an effective balance between model size and computational efficiency, offering higher recovery quality while maintaining competitive training times.

VI. CONCLUSION

In this paper, we propose a novel deep learning model MASTA based on the encoder-decoder architecture for trajectory recovery considering the road network topology characteristics. Since the spatial features in the trajectory data are significant but previously ignored in the recovery process, we introduce a pre-trained ANE module to integrate road topology characteristics into the trajectory data, which helps to extract spatio-temporal dependencies more effectively. Subsequently, we employ the attention mechanism and GRU for spatial and temporal correlation modeling, respectively. Furthermore, since the large number of candidate road segments in a city can adversely affect the model's inference process, we propose a novel adaptive mask inference module, consisting of a *distance-based mask matrix* and a learnable *adaptive mask matrix* to assist the model in making segment inference by weighting each candidate segment adaptively.

To evaluate the recovery performance of MASTA, we conduct comprehensive case studies on two real-world trajectory

datasets. Compared with the state-of-the-art baselines, the simulation results demonstrate the superiority and stable performance of the proposed model. An ablation test is conducted to demonstrate the effectiveness of each sub-module in contributing to the overall performance. In addition, we investigate the sensitivity of four hyperparameters on model performance. While MASTA shows strong recovery performance, one limitation is that it may rely on highly accurate map-matching results, which was not fully evaluated in this study. Addressing map-matching errors in LSR trajectories and making the model more robust to such errors is a valuable future direction.

REFERENCES

[1] 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. 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, 2018, pp. 2500–2507.

[2] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data," *Transp. Res. C, Emerg. Technol.*, vol. 112, pp. 62–77, Mar. 2020.

[3] Y. Zhu, Y. Liu, J. J. Q. Yu, and X. Yuan, "Semi-supervised federated learning for travel mode identification from GPS trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2380–2391, Mar. 2022.

[4] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Proc. 11th Int. Conf. Mobile Data Manage.*, Kanas City, MO, USA, May 2010, pp. 43–52.

[5] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, pp. 29:1–29:41, May 2015.

[6] Y. Zhou, Y. Lin, S. Ahn, P. Wang, and X. Wang, "Platoon trajectory completion in a mixed traffic environment under sparse observation," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16217–16226, Sep. 2022.

[7] K. Zhang, Z. Fan, X. Song, and S. Yu, "Enhancing trajectory recovery from gradients via mobility prior knowledge," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5583–5594, Mar. 2023.

[8] S. Hoteit, S. Secci, S. Sobolevsky, C. Ratti, and G. Pujolle, "Estimating human trajectories and hotspots through mobile phone data," *Comput. Netw.*, vol. 64, pp. 296–307, May 2014.

[9] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values," *Transp. Res. C, Emerg. Technol.*, vol. 118, Sep. 2020, Art. no. 102674.

[10] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.

[11] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Melbourne, VIC, Australia, Aug. 2017, pp. 3083–3090.

[12] J. Wang, N. Wu, X. Lu, W. X. Zhao, and K. Feng, "Deep trajectory recovery with fine-grained calibration using Kalman filter," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 921–934, Mar. 2021.

[13] H. Ren et al., "MTrajRec: Map-constrained trajectory recovery via Seq2Seq multi-task learning," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1410–1419.

[14] J. Feng et al., "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. World Wide Web Conf.*, Lyon, France, 2018, pp. 1459–1468.

[15] T. Xia et al., "AttnMove: History enhanced trajectory recovery via attentional network," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4494–4502.

[16] C. Hou, S. He, and K. Tang, "RoSANE: Robust and scalable attributed network embedding for sparse networks," *Neurocomputing*, vol. 409, pp. 231–243, Oct. 2020.

[17] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Hannover, Germany, Apr. 2011, pp. 900–911.

[18] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, Jun. 2013, pp. 713–724.

[19] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Washington, DC, USA, Apr. 2012, pp. 1144–1155.

[20] H.-P. Hsieh, C.-T. Li, and S.-D. Lin, "Exploiting large-scale check-in data to recommend time-sensitive routes," in *Proc. ACM SIGKDD Int. Workshop Urban Comput.*, Beijing, China, Aug. 2012, pp. 55–62.

[21] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Beijing, China, Aug. 2012, pp. 195–203.

[22] H. Liu, L.-Y. Wei, Y. Zheng, M. Schneider, and W.-C. Peng, "Route discovery from mining uncertain trajectories," in *Proc. IEEE 11th Int. Conf. Data Mining Workshops*, Vancouver, BC, Canada, Dec. 2011, pp. 1239–1242.

[23] P. Banerjee, S. Ranu, and S. Raghavan, "Inferring uncertain trajectories from partial observations," in *Proc. IEEE Int. Conf. Data Mining*, Shenzhen, China, Dec. 2014, pp. 30–39.

[24] H. Wu et al., "Probabilistic robust route recovery with spatio-temporal dynamics," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, 2016, pp. 1915–1924.

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

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[27] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017.

[28] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Paris, France, Apr. 2018, pp. 617–628.

[29] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Dallas, TX, USA, 2020, pp. 949–960.

[30] 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 *Proc. IEEE Intell. Vehicles Symp. (IV)*, Suzhou, China, Jun. 2018, pp. 1672–1678.

[31] T.-Y. Fu and W.-C. Lee, "Trembr: Exploring road networks for trajectory representation learning," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 1, pp. 10:1–10:25, 2020.

[32] A. Al-Molegi, M. Jabreel, and B. Ghaleb, "STF-RNN: Space time features-based recurrent neural network for predicting people next location," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Athens, Greece, Dec. 2016, pp. 1–7.

[33] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, 2016, pp. 194–200.

[34] 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. AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, 2019, vol. 33, no. 1, pp. 5458–5465.

[35] K. Chowdhary and K. Chowdhary, "Natural language processing," in *Fundamentals of Artificial Intelligence*, 2020, pp. 603–649.

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 27, Montreal, QC, Canada, Dec. 2014, pp. 3104–3112.

[37] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Phys. A, Stat. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, Mar. 2011.

[38] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," 2011, *arXiv:1101.3291*.

[39] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2014, pp. 701–710.

[40] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

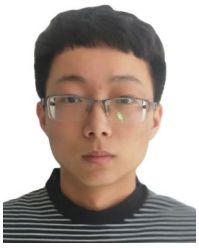
[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. 1st Int. Conf. Learn. Represent. (ICLR)*, Scottsdale, AZ, USA, Y. Bengio and Y. LeCun, Eds., May 2013.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 27th Annu. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., Dec. 2013, pp. 3111–3119.

[43] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inform. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

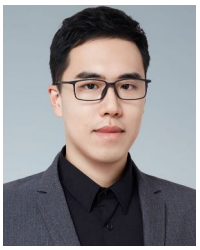
[44] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "GeoMAN: Multi-level attention networks for geo-sensory time series prediction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, Jul. 2018, pp. 3428–3434.
[45] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[46] C. Tian and W. K. Chan, "Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies," *IET Intell. Transp. Syst.*, vol. 15, no. 4, pp. 549–561, 2021.
[47] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. 33rd AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, 2019, pp. 922–929.
[48] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden Markov model with precomputation," *Int. J. Geographical Inf. Sci.*, vol. 32, no. 3, pp. 547–570, 2018.
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[50] M. M. Elshrif, K. Isufaj, and M. F. Mokbel, "Network-less trajectory imputation," in *Proc. 30th Int. Conf. Adv. Geographic Inf. Syst.*, Seattle, WA, USA, Nov. 2022, pp. 8:1–8:10.
[51] J. Si et al., "TrajBERT: BERT-based trajectory recovery with spatial-temporal refinement for implicit sparse trajectories," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4849–4860, May 2024.
[52] J. J. Q. Yu, C. Markos, and S. Zhang, "Long-term urban traffic speed prediction with deep learning on graphs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7359–7370, Jul. 2022.

Yongchao Ye (Student Member, IEEE) received the B.Eng. degree in computer science and technology from Ningbo University, Ningbo, China, in 2020, and the M.Phil. degree from the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, in 2023. He is currently pursuing the Ph.D. degree with the Department of Data Science, College of Computing, City University of Hong Kong. His research interests include human mobility analytics, trajectory generation, and spatio-temporal data mining in intelligent transportation systems.



Ao Wang received the B.Eng. degree in computer science and technology from Southern University of Science and Technology, Shenzhen, China, in 2021, and the M.Phil. degree from the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, in 2024. His research interests include smart city, intelligent transportation systems, and deep learning.

Adnan Zeb is currently a Lecturer with the School of Intelligent Manufacturing and Smart Transportation, Suzhou City University, Suzhou, China. His research interests include deep representation learning, deep graph embedding networks, spatio-temporal data mining, and their applications to intelligent transportation systems.



Shiyao Zhang (Member, IEEE) received the B.S. degree (Hons.) in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2014, the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2016, and the Ph.D. degree from The University of Hong Kong, Hong Kong, SAR, China, in 2020. He was a Post-Doctoral Research Fellow with the Academy for Advanced Interdisciplinary Studies, Southern University of Science and Technology from 2020 to 2022. He is currently a Research Assistant Professor with the Research Institute for Trustworthy Autonomous Systems, Southern University of Science and Technology. His research interests include smart cities, smart energy systems, intelligent transportation systems, optimization theory and algorithms, and deep learning applications.

James Jianqiao Yu (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in electrical and electronic engineering from The University of Hong Kong, Pokfulam, Hong Kong, in 2011 and 2015, respectively. He was a Post-Doctoral Fellow with The University of Hong Kong from 2015 to 2018; and an Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, from 2018 to 2023. He is currently a Lecturer with the Department of Computer Science, University of York, U.K. He has published over 100 academic papers in top international journals and conferences, and representative papers have been selected as ESI highly cited papers. His general research interests are in data mining, embodied artificial intelligence, and intelligent transportation systems. His work is mainly on multi-modal large language model, spatial–temporal data mining, and forecasting and logistics of future transportation systems. He was the World's Top 2% Scientists of single year and career by Stanford University. He is an Editor of *IET Smart Cities*.