TERI: An Effective Framework for Trajectory Recovery with Irregular Time Intervals

Yile Chen
Nanyang Technological University
yile001@e.ntu.edu.sg

Gao Cong
Nanyang Technological University
gaocong@ntu.edu.sg

Cuauhtemoc Anda
DataSpark
temo.anda@dsanalytics.com

ABSTRACT

The proliferation of trajectory data has facilitated various applications in urban spaces, such as travel time estimation, traffic monitoring, and flow prediction. These applications require a substantial volume of high-quality trajectories as the prerequisite to achieve effective performance. Unfortunately, a large number of real-world trajectories are inevitably collected in unsatisfactory quality due to device constraints. To address this issue, previous studies have proposed numerous trajectory recovery methods to augment the quality of such trajectories, thereby ensuring the performance of related applications. However, these methods all assume the awareness of the recovery positions in advance, which is a condition not always available in practice. In this paper, we discard this strong assumption and focus on trajectory recovery with irregular time intervals as a more prevalent setting in downstream scenarios. We propose a novel framework, called TERI, to tackle trajectory recovery without prior information in a two-stage process, where recovery positions are first detected, followed by the imputation of the missing data points. In each stage, TERI framework deploys a model named RETE, which is based on Transformer encoder architecture enhanced by novel designs to boost the performance for the new problem setting. Specifically, RETE features a learnable Fourier encoding module to better model spatial and temporal correlations, and integrates collective transition pattern learning and trajectory contrastive learning to effectively capture sequential transition patterns. Extensive experiments on three real-world datasets demonstrate that TERI consistently outperforms all the baselines by a significant large margin.

PVLDB Reference Format:
Yile Chen, Gao Cong, Cuauhtemoc Anda. TERI: An Effective Framework for Trajectory Recovery with Irregular Time Intervals. PVLDB, 17(3): 414 - 426, 2023.
doi:10.14778/3632093.3632105

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/yileccc/TERI.

1 INTRODUCTION

With the rapid development of geopositioning technologies, trajectories are being collected at an unprecedented speed to enhance urban intelligence in a range of applications, including travel time estimation [19], traffic monitoring [5, 14], and flow prediction [42]. Serving as a vital data source for these applications, trajectories are expected to be dense and accurate to represent the faithful and detailed underlying movements of objects. Otherwise, the uncertainty between consecutive records within trajectories could increase, thus greatly impacting the effectiveness of these applications [27]. Unfortunately, a large quantity of trajectories are inevitably sparse due to device malfunctions or cost-saving measures that involve utilizing low sampling rates. Therefore, it is essential to develop trajectory recovery methods that produce trajectories with higher qualities to benefit the performance of related applications.

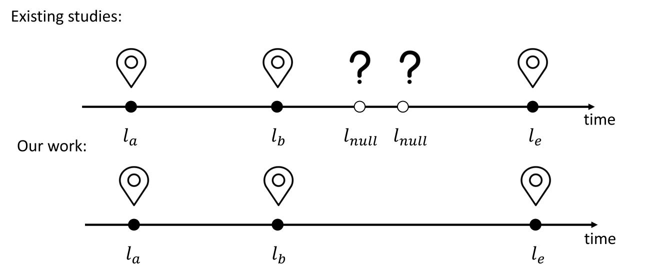
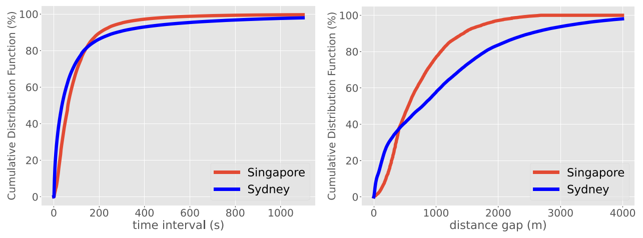


Figure 1: Difference between existing studies and our work.

Extensive studies have been proposed to address the trajectory recovery problem, and they can be classified into two categories based on whether topological constraints (i.e., road networks) are considered. In the first category, data points are map-matched onto road networks, and then the recovery task is transformed into completing the missing route between two distant road segments [7, 13, 18, 27]. In contrast, trajectory recovery in the second category is performed in the way that such structural knowledge is not utilized, and thus called **network-free trajectory recovery** [29, 31, 35, 36]. In this paper, we focus on the latter problem setting without applying additional map-matching operations for data points, as it represents a more general use case in practice.

Despite the difference in problem settings, existing studies on trajectory recovery share a common underlying assumption which can be abstracted in Figure 1: *the recovery positions are provided as prior information.* On one hand, studies [13, 18] on road networks aim to complete the missing parts between two disconnected road segments, and these gaps naturally serve as the recovery positions. On the other hand, other studies either have the knowledge about the positions of the unobserved locations [29, 35, 36], or target at recovering regularly sampled trajectories (e.g., 30 seconds per data point) [7, 27, 31]. In all these instances, the recovery positions can be transformed into placeholders which are either explicitly given or implicitly identified by calculating the time interval between consecutive data points, respectively. Taking Figure 1 as

Proceedings of the VLDB Endowment, Vol. 17, No. 3 ISSN 2150-8097.
doi:10.14778/3632093.3632105



(a) Time interval. **(b) Distance interval.**

Figure 2: Illustration of irregular time and distance intervals.

an example, the question marks could be viewed as placeholders employed in previous studies. They indicate the existence of two unknown locations following l_b to be recovered. In contrast, such prior knowledge is not provided in our study.

We argue that the assumption employed in existing studies has great limitations. Due to the factors such as reducing energy consumption (e.g., logistics truck) or accommodating communication characteristic for data sampling (e.g., cellular data), time intervals between two consecutive data points are usually irregular (i.e., with unfixed sampling rate). For example, Figure 2a illustrates the irregular interval patterns in real-world cellular trajectories from two cities utilized in the experiments. These two datasets are obtained from a leading telecommunication company in Singapore. Considering the advanced mobile technology and well-developed telecommunication infrastructure in urban areas, cellular trajectories can be more extensively and easily collected on a larger scale. This creates the expectation for the company to derive more accurate and granular trajectories, thus enabling the analysis of urban mobility at a greater level of detail. Consequently, the recovery of low-quality trajectories with irregular time intervals becomes a fundamental requirement to benefit the company's services in diverse business scenarios, such as mobility pattern analysis, urban infrastructure optimization, and location-based marketing. For example, by studying recovered trajectories with more details, the company can have a deeper understanding of urban mobility patterns to enhance traffic monitoring, and event management.

However, unlike the setting in regularly sampled trajectories, time intervals in these trajectories cannot be regarded as an indicator to infer recovery positions. Furthermore, in reality we have no means to determine in advance whether any locations are missing for a given trajectory. Therefore, the awareness of the positions for missing locations, is not feasible in practical situations.

To address the limitations in existing studies, we consider a new setting that does not rely on the awareness of recovery positions based on explicitly or implicitly predetermined placeholders. Instead, we do not impose any prior information on the positions where trajectory recovery should be performed. In other words, we aim to not only discover and but also recover the missing data points from trajectories with irregular intervals. However, this problem is non-trivial due to the following challenges.

$\underline{\text{(Challenge 1)}}$ Given that the recovery positions are not provided in advance, the Sequence-to-sequence (Seq2seq) model variants in previous studies [7, 27, 31], which formulate the recovery as a sequence generation process, are not adequate for the new problem setting. In the context of our problem where the prior information for placeholders are not available, each data point in the original sparse trajectory is expected to be produced together with the missing data points in the decoding phase. Unfortunately, it is usually the case that some data points in the original trajectory are not successfully reconstructed in Seq2seq models [12], which could lead to *ambiguous recovery results*. Taking Figure 1 as an example, given the original trajectory (l_a, l_b, l_e), the Seq2seq model might generate a prediction (l_a, l_c, l_e) in the decoder while failing to copy l_b. In this case, it is unclear whether l_b should be placed after l_a or l_c, as both are valid options.

$\underline{\text{(Challenge 2)}}$ Trajectories with irregular intervals exhibit *complex spatial and temporal correlations*, and therefore we cannot easily determine the recovery positions based solely on large time or distance intervals. As shown in Figure 2, for all the data points in the Sydney dataset, approximately 25% have time interval exceeding 100 seconds, and 40% have distance interval greater than 1000 meters. If we heuristically choose a threshold that is too large (small) to identify recovery positions, the recall (precision) would significantly decrease. Moreover, although we may split the time or distance into different bins (e.g., 10 mins or 100 meters) to create discrete embedding vectors, it is difficult to manually select an appropriate number and interval of bins. Even worse, this approach neither effectively models fine-grained spatial and temporal correlations, as different intervals within the same bin are considered identical (e.g., intervals of 10 mins and 18 mins are indistinguishable with a 10-20 minute bin), nor it is capable of handling unseen intervals (e.g., very long intervals) that may exist in testing examples.

$\underline{\text{(Challenge 3)}}$ Given a sparse trajectory with unobserved missing locations, it is critical to capture *sequential transition patterns* to offer invaluable insights for the trajectory recovery task. Specifically, it's important to ensure that the recovered trajectory makes logical sense considering its full context, and that the recovered locations align with typical movement patterns hidden in the data. For instance, in the challenge presented, an adequate model should identify the unsmooth and anomalous sequential transition between l_b and l_e, thereby indicating the high likelihood of missing locations in between. Capturing such patterns could be achieved using sequential models such as recurrent neural networks (RNN) [15] or Transformer [30], as applied in [29, 36]. However, these models struggle to handle the scenarios when trajectories become short and contain limited context information. Therefore, dedicated designs are required to mitigate the issue.

To address these challenges in both tackling the new problem setting and the unique data characteristics inherent in cellular trajectoies, we propose a novel framework, named TERI, for **T**rajectory r**E**covery with i**R**regular time **I**ntervals. To eliminate the ambiguous recovery results (Challenge 1) produced by the sequence generation of Seq2seq models, we divide the task into two stages, namely the detection stage and the recovery stage. Specifically, the detection stage aims to identify the recovery positions for given trajectories. Then the recovery stage generates the predictions for those missing data points at each recovery position. The framework is built on the Transformer encoder, which formulates the recovery process as a pipeline of classification followed by imputation, thus avoiding the ambiguity by ensuring all the data points in the original trajectory are preserved in the final prediction. To better adapt the Transformer encoder to our problem setting, we employ a novel time and distance encoding module based on learnable Fourier features [20, 37], which can model continuous intervals without pre-defining any embedding vectors, while also being naturally inductive to handle arbitrary intervals. Consequently, complex spatial and temporal correlations in irregular intervals (Challenge 2) are encoded into the model. Moreover, to better capture sequential transition patterns (Challenge 3), we propose to harness the collective information from trajectory databases by constructing a global transition graph. We then adopt graph neural networks (GNN) [6, 34] to encode spatial proximity in location representations, as the representations are influenced by precedents during message passing operations in the global transition graph. In addition, since trajectories with different underlying routes have distinct sequential transition patterns, we develop trajectory contrastive learning to further enrich the module by distinguishing trajectory variants with the same underlying route from those on different routes. These proposed techniques are integrated to obtain a Recovery Enhanced version of Transformer Encoder (RETE). Finally, RETE serves as a unified backbone model, which requires only minimal modifications in the last layer to fulfill the function for each stage effectively. Utilizing the proposed pipeline coupled with these specifically designed modules, TERI is tailored to seamlessly tackle the new trajectory recovery scenario, and effectively overcome the issues unresolved by previous studies.

The contributions of our work are summarized as follows:

- We identify the limitations in existing studies and introduce a two-stage framework, named TERI, to address trajectory recovery with irregular intervals. To the best of our knowledge, TERI is the first attempt to tackle this problem setting, which has more practical and general real-world use cases.
- We propose a novel backbone model, called RETE, to accommodate the new problem setting. RETE incorporates three new techniques, namely learnable Fourier features for interval encoding to consider complex spatial and temporal correlations, collective transition pattern learning module and trajectory contrastive learning to effectively capture sequential transition patterns.
- We conduct extensive experiments on three real-world datasets with irregular intervals to evaluate the effectiveness of the proposed framework. Experimental results demonstrate that the TERI significantly outperforms existing solutions.

2 RELATED WORK

2.1 Road-Network-based Trajectory Recovery

Trajectory recovery on road networks aims to infer the most likely route between two non-adjacent road segments. In this setting, road network structure is used as prior knowledge or constraints to address the problem. Early studies [1, 44] utilize Markov-based models with road network constraints to derive spatial transition probabilities for road segment pairs. Several studies propose search algorithms in a data-driven manner to obtain the recovery route based on dynamic programming [23], inverse reinforcement learning [33] or heuristic cost measures [13]. Recently, deep learning based models have been adopted to model the complex factors in trajectory recovery on road networks. Li et al. [18] propose a deep generative model which explains the generation of a route conditioning on the past route sequence, destination and real-time traffic. Wu et al. [32] propose to improve A^* search algorithms in trajectory recovery using neural networks to automatically learn the cost functions without handcrafting search strategies. Moreover, Ren et al. [27] and Chen et al. [7] integrate trajectory recovery into map-matching [25, 39], which aims to produce map-matched records with a high sampling rate. They propose a Seq2seq model with multi-task learning to recover fine-grained data points directly on road networks from GPS trajectories. In addition, some methods tackle this problem from another perspective, and propose to utilize traffic camera videos to identify and track vehicles for trajectory recovery [22, 41]. While these methods have achieved promising results, they essentially rely on modeling the graph structure of road network, and are therefore not applicable to our problem setting.

2.2 Network-free Trajectory Recovery

Different from the above studies, another line of research, referred to as network-free trajectory recovery, aims to work with fixed locations rather than road network structure. Early studies propose to predict a sequence of traversed locations between a pair of origin and destination locations by formulating it as an optimization problem [11], utilizing pre-defined anchor points for calibration [28], or adopting Markov model [3, 8]. However, these approaches simply consider low-order transition patterns in a trajectory, failing to capture spatial and temporal context with long-term dependencies. To overcome this issue, sequential models based on RNN [9, 15] or Transformer [30] have been applied to the problem. For example, Xi et al. [35] propose to model the location correlations with a fixed window size using embedding techniques, which can only capture the local location correlations. To further improve the performance, Wang et al. [31] employ a RNN-based Seq2seq framework equipped with spatial and temporal attentions to capture transition patterns, and introduce a calibration component based on the Kalman filter to reduce the predictive uncertainty. Xia et al. [36] utilize historical trajectories of users, and adopt self-attention mechanisms both within and across trajectories to recover unobserved locations given as placeholders. Building on this model, Sun et al. [29] further incorporate GNN to model each individual trajectory, which helps better capture the transition patterns. Moreover, Elshrif et al. [10] propose a heuristic search algorithm to impute locations between two consecutive but distant records within a trajectory. As discussed in Section 1, these methods share the same underlying assumption that the recovery positions are provided as prior information. However, such oracle information is usually not available in practice. In contrast, our framework does not require the recovery positions, making it more versatile and applicable to real-world scenarios.

3 PROBLEM FORMULATION

In this section, we present some definitions used throughout this paper, and then formulate the new problem setting of trajectory recovery with irregular intervals.

Definition 3.1. (**Trajectory**). A trajectory T of length n is defined as a trip within a city, comprising of a sequence of chronologically ordered data points, denoted as $T = (l_1, l_2, ..., l_n)$. Each l_i is a location associated with a coordinate c_i and a timestamp t_i to indicate the arrival time of the location.

Unlike the studies [27, 31] which work on trajectories with constant intervals for consecutive data points, i.e., $t_{i+1} - t_i = \epsilon$ for $1 \leq i \leq n-1$ where ϵ is a fixed time interval, to accommodate various sampling characteristics across different data sources and facilitate more generic use cases, we do not make such an assumption in our setting.

Definition 3.2. (**Sparse Trajectory**). A sparse trajectory $\tilde{T} = (l_{a_1}, l_{a_2}, ..., l_{a_k})$ is a part of T of length k ($k < n$), in which some data points are omitted. Each a_i corresponds to an index of a data point in T, i.e., $a_i \in \mathbb{N}$: $a_i \in [1, n]$, and $1 \leq a_1 < ... < a_i < a_{i+1} < ... < a_k \leq n$.

We note that the definition of sparse trajectory in this paper differs from those in [29, 35, 36], where locations are masked with null placeholders (e.g., the question marks in Figure 1) without actually being dropped, i.e., $k = n$. This implies that the recovery positions are provided in advance as prior knowledge to the model, which contradicts many practical scenarios. Then we formulate the problem setting of trajectory recovery as follows.
Problem Statement. Given a sparse trajectory $\tilde{T}$, the objective is to recover its corresponding T. In other words, we aim to recover the missing locations that are not present in $\tilde{T}$, i.e., $l_i \in T$ and $l_i \notin \tilde{T}$.

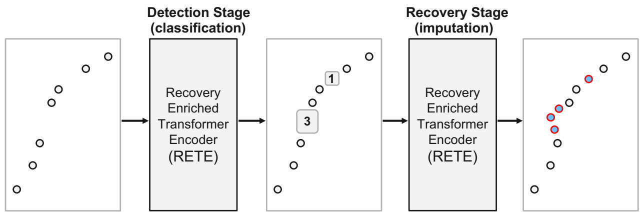
4 METHODOLOGY

In this section, we first provide an overview of TERI framework. Then we briefly introduce the Transformer encoder as the main building block of our RETE model. Next, we present the details of the modules specifically designed for the RETE model, including time and distance encoding module, collective transition pattern learning module and trajectory contrastive learning module. Finally, we introduce the training procedure of TERI framework.

4.1 Framework Overview

The pipeline of the proposed TERI framework is illustrated in Figure 3. To address the recovery ambiguity issue in Seq2seq models, we propose to rephrase the problem into two stages. First, classification is performed to determine the number of missing locations following each observed data point, thereby identifying both the recovery positions and their lengths (detection stage). Next, given the results from the detection stage, we insert the corresponding number of placeholders into the original trajectory and perform imputation to replace them with predicted locations as the final recovery results (recovery stage). In contrast to decoding phase in the Seq2seq framework, the proposed recovery process is structured into the manner that predicted locations are inserted into positions identified as potentially incomplete, thus successfully avoiding the recovery ambiguity issue in Challenge 1.

To achieve this, we propose a unified model, called Recovery Enhanced Transformer Encoder (RETE), to serve as the backbone for each stage. The overview of RETE model is shown in Figure 4. RETE is built on Transformer encoder [30] which treats the trajectory as input sequence and utilizes self-attention mechanism at each location. Moreover, RETE is further enhanced with novel designs to better adapt to our problem setting. First, since irregular intervals are crucial features to be considered, we adopt an encoding method based on learnable Fourier features [20, 37] to handle the continuous properties of distance and time intervals (Section 4.3). In



Detection Stage (classification)
Recovery Stage (imputation)
Recovery Enriched Transformer Encoder (RETE)
Recovery Enriched Transformer Encoder (RETE)
1
3

Figure 3: The pipeline of TERI framework.

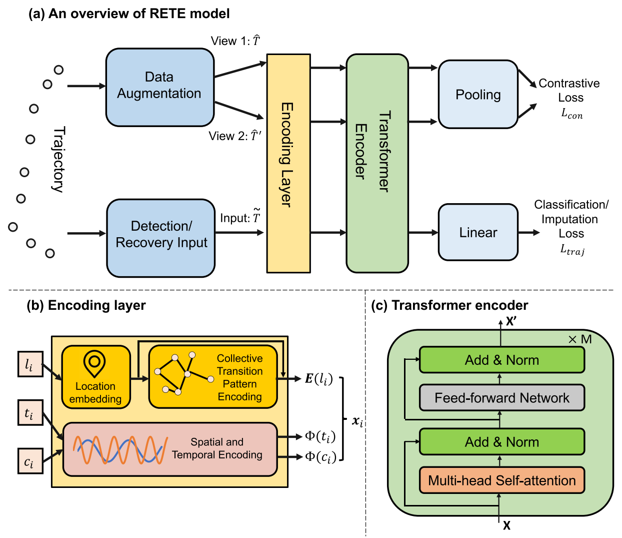


Figure 4: The model architecture of RETE.

contrast to existing studies [29, 36] which manually select intervals to produce embedding vectors and regard different times or distances belonging to the same interval bin as the same, this method retains fine-grained interval information, and seamlessly integrates complex spatial and temporal correlations in a trajectory to the self-attention mechanism, thus addressing Challenge 2. Next, the capture of sequential transition patterns provides informative signals for trajectory recovery in both stages. Although Transformer is capable of capturing such patterns, the effectiveness declines for short and sparse trajectories due to their limited spatial and temporal contexts. To this end, we propose to enhance the model by harnessing the collective information from trajectory databases (Section 4.4). Specifically, we construct a global transition graph based on transition frequencies for consecutive data points, and apply GNN to obtain the global contextual location representations, which explicitly model the influences among different locations through GNN messaging passing. Moreover, we integrate a trajectory contrastive learning module into the model. The module aims to distinguish trajectory variants with the same underlying route from those with different routes (Section 4.5). By doing so, trajectories with more similar sequential transition patterns could be identified, which further enhances the effectiveness of the model. These two modules are combined together to address Challenge 3. Finally, RETE is flexible to be applied in both the detection stage and the recovery stage with only minor adjustments to the training objectives (Section 4.6), thus reducing the efforts to design respective components for the pipeline of trajectory recovery.

4.2 Transformer Encoder

Before presenting the details of the RETE model, we first introduce some preliminaries regarding Transformer encoder [30], which has become the default choice for sequence modeling across various tasks. As shown in Figure 4(c), Transformer encoder is composed of several identical self-attention layers, each of which includes two sub-layers: multi-head self-attention module followed by position-wise Feed-foward Network. Then we describe the details of these two components.

Multi-head Self-attention Module. Self-attention mechanism allows the model to selectively focus on correlated parts of the input sequence. It can be described as mapping the representations of input sequence to output representations through scaled dot-product function, which is defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K and V represent the query, key, and value matrix respectively derived from linear projection on the representations of the input trajectory, and d_k is the vector dimension, which is typically set to be the same for all the three matrices.

In our work, we adopt multi-head self-attention to model trajectory sequences. Specifically, the representations of input trajectory are projected into h sets of different queries, keys, and values to perform self-attention mechanism, which has been shown to achieve better performance. Given the input representations $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N] \in \mathbb{R}^{N \times d_{in}}$ with length N where $\mathbf{x}_i$ is the representation of the i-th location in the trajectory after the encoding layer, the output representations $\mathbf{Z} \in \mathbb{R}^{N \times d_{out}}$ are produced as follows:

$$\begin{aligned} \mathbf{Z} &= \text{MH-Attn}(\mathbf{X}) = [head_1, \dots, head_h] \cdot \mathbf{W}^O \\ head_i &= \text{Attention}\left(\mathbf{X}\mathbf{W}_i^Q, \mathbf{X}\mathbf{W}_i^K, \mathbf{X}\mathbf{W}_i^V\right) \end{aligned} \tag{2}$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d_{in} \times d_{in}/h}$ are projection matrices for each head, and $\mathbf{W}^O \in \mathbb{R}^{d_{in} \times d_{out}}$ is learnable parameters.

Position-wise Feed-forward Network. After the multi-head self-attention module, the output representations are passed through a two-layer fully connected feed-forward network (FFN) with ReLU activation function:

$$\text{FFN}(\mathbf{Z}) = \text{ReLU}\left(\mathbf{Z}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2 \tag{3}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are parameters of the network.

Model Stacking. In practice, it is usually beneficial to stack multiple layers to obtain more effective embeddings for downstream tasks. To alleviate possible training difficulty caused by the increasing depth of more stacked layers, we follow [30] to employ the residual connection and layer normalization on each of the two sub-layers. It can be expressed as follows:

$$\begin{aligned} \mathbf{Z}' &= \text{LayerNorm}(\mathbf{X} + \text{MH-Attn}(\mathbf{X})) \\ \mathbf{X}' &= \text{LayerNorm}(\mathbf{Z}' + \text{FFN}(\mathbf{Z}')) \end{aligned} \tag{4}$$

where LayerNorm denotes layer normalization and $\mathbf{Z}'$ denotes the final output representations which are passed as the input to the subsequent layer of Transformer encoder.

4.3 Time and Distance Encoding

Considering the crucial role of spatial and temporal correlations in identifying missing locations, it is essential to effectively encode these complex correlations into the model. The original Transformer encoder is limited to only encoding the ordering information in the input trajectory through the use of pre-defined or learned positional embeddings. To further incorporate spatial and temporal correlations, previous methods on trajectory recovery [29, 31, 36] utilize discrete embeddings to represent specific intervals, with each embedding associated with a particular time bin. For example, every 5-minute interval can be treated as a bin for embedding. However, this approach has significant drawbacks. It is a non-trivial task to select appropriate intervals to obtain discrete embeddings in the setting of irregular intervals, and the method also fails to model the fine-grained correlations for two different intervals that fall in the same bin.

In light of this, we adopt a novel spatial and temporal encoding method based on learnable Fourier features to address the limitations. Specifically, we propose to learn an encoding function $\Phi(x) : \mathbb{R}^m \rightarrow \mathbb{R}^d$, which maps the original feature to a d-dimensional vector. The function is defined as follows:

$$\Phi(x) = \frac{1}{\sqrt{d}}\left[\cos x\mathbf{W}_r \| \sin x\mathbf{W}_r\right] \tag{5}$$

where $\|$ denotes vector concatenation, $\mathbf{W}_r \in \mathbb{R}^{m \times d/2}$ is the learnable parameter which defines the orientation and the wavelength of Fourier features. Here, the feature x could either represent a timestamp (temporal information) or location coordinates (spatial information). Then the dot-product between two features $\Phi(x)$ and $\Phi(y)$ is calculated as:

$$\Phi(x) \cdot \Phi(y) = \mathcal{K}(x, y) = \frac{1}{d}\sum(\cos(x - y)\mathbf{W}_r) \tag{6}$$

This means that the mapping function in Equation 5 corresponds to a translation-invariant kernel (i.e., $\mathcal{K}(x, y) = \mathcal{K}(x + c, y + c)$) parameterized by $\mathbf{W}_r$. The proposed encoding paradigm has several advantages. First, the translation invariant property allows the model to focus on the gap information for temporal interval or geographical distance, which is more desired in trajectory recovery task, as opposed to relying on absolute values of spatial and temporal features. Second, it endeavors to model the correlations directly on continuous features without manually selecting the intervals for discrete embeddings. This reduces the information loss associated with discrete embeddings. Third, it is applicable to unseen features that might not be present in training examples. Furthermore, for every data point l_i in a trajectory, such a dot-product formulation can seamlessly integrate the correlations into the self-attention mechanism by modifying the input embeddings to the Transformer encoder as follows:

$$\mathbf{x}_i = \mathbf{E}(l_i) \| \Phi(t_i) \| \Phi(c_i) \tag{7}$$

where $\Phi(t_i)$ and $\Phi(c_i)$ denote the temporal encoding on timestamp and the spatial encoding on location coordinates respectively, and $\mathbf{E}(l_i)$ is the enhanced location representation for l_i, which will be detailed in the next subsection. The process is illustrated in shown in Figure 4(b). To facilitate model training, we initialize the parameter from a normal distribution $\mathbf{W}_r \sim \mathcal{N}(0, \sigma^{-2})$. By doing this, Equation 6 approximates the Gaussian kernel (i.e., $\Phi(x)\cdot\Phi(y) \approx \exp(-\|x-y\|^2/\sigma^2)$) over its original feature differences [26], and this introduces a useful inductive bias of Euclidean distances as the start point in the model.

4.4 Collective Transition Pattern Learning

While Transformer encoder demonstrates its ability to capture sequential transition patterns, it still struggles when dealing with short and sparse trajectories due to their limited context information, which in turn provide less informative knowledge about transition patterns. To mitigate this issue, we leverage the collective information from trajectory databases to explicitly integrate such knowledge into the model. Specifically, we propose to construct a global transition graph, denoted as $\mathcal{G} = (\mathcal{L}, \mathcal{E}, \mathbf{A})$, to provide a comprehensive view of global movement patterns. Here, $\mathcal{L}$, $\mathcal{E}$ and $\mathbf{A}$ denote a set of unique locations across all trajectories, a set of edges, and the adjacency matrix, respectively. To represent the spatial proximity, we define each entry $\mathbf{A}_{ij}$ to be the frequency of co-occurrences for consecutive data points. Given the global transition graph, we adopt graph convolutional networks (GCN) [17], a commonly used GNN variant, to derive contextual location representations. Specifically, we first derive the normalized adjacency matrix for a non-symmetric $\tilde{A} \in \mathbb{R}^{L\times L}$ as follows:

$$\tilde{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-1}\hat{\mathbf{A}} \tag{8}$$

where $\mathbf{D}$ is the degree matrix, $\mathbf{I}$ is the identity matrix, and $\hat{\mathbf{A}} = \mathbf{A}+\mathbf{I}$ is the adjacency matrix augmented with self-hoop. Then the message passing in GCN is performed as follows:

$$\mathcal{F}(\mathbf{H}^{(k-1)}) = \mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)} + \mathbf{b}^{(k)}\right) \tag{9}$$

where $\mathbf{H}^{(k-1)}$ is the embeddings for the k-th layer, $\mathbf{W}^k$ and $\mathbf{b}^k$ are the parameters in the k-th layer, and σ is the activation function. From the perspective of an individual location l_i, this process can be reformulated as:

$$\mathbf{h}_i^{(k)} = \sigma\left(\sum_{j\in N(i)} \frac{1}{\sqrt{|N(i)||N(j)|}}\mathbf{h}_j^{(k-1)}\mathbf{W}^{(k)} + \mathbf{b}^{(k)}\right) \tag{10}$$

where $N(i)$ denotes the set of neighbors for node l_i. This illustrates that for each location, the message passing operation updates the location representation by aggregating information from its neighbors in the global transition graph (i.e., precedents in collective trajectories). In this way, the model explicitly models the influences among different locations, thus enriching the knowledge about sequential transition patterns.

We can stack multiple layers to derive the final location representations that preserve the influences among locations with higher-order, as this could encode more complex contextual information in trajectories. In our model, we take the results at the t-th layer and apply residual connection to obtain the enhanced location representations as depicted in Equation 7:

$$\mathbf{E} = \mathcal{F}^t(\mathbf{L}) + \mathbf{L} \tag{11}$$

where $\mathbf{L}$ represents the initial location embedding matrix for all the locations at the start of the encoding layer in RETE.

4.5 Trajectory Contrastive Learning

Given that missing data points can occur in any segment of a trajectory, the resulting sparse trajectories may exhibit substantial variation with respect to the original locations. However, these trajectories essentially share the same underlying route, which implies a high similarity in their sequential transitions patterns. Since these sparse trajectories can provide complementary information for trajectory recovery, it is beneficial to effectively discriminate them from other trajectories to further enrich the sequential transition patterns. Such an objective naturally suggests the use of contrastive learning, which is designed to draw semantically similar positive sample pairs closer while pushing unrelated negative samples farther apart [4]. To achieve this, for a given trajectory T, we apply data augmentation to generate two trajectory variants, denoted as $\hat{T}$ and $\hat{T}'$, as two views of T. Analogous to data augmentation operations utilized in images, the idea of data augmentation for trajectories aims to generate alternative views of a given trajectory with additional variations, while preserving the semantic characteristics (i.e., the same underlying route). Apart from the positive trajectory pair, we also randomly sample other $(M-1)$ negative trajectories and apply the same procedure to create a batch of $2M$ trajectories. These trajectories are processed through our RETE model followed by a pooling layer to produce representations for all the trajectories. Then for each trajectory, it is trained to align with the other view of itself (positive sample) among the $(2M-1)$ instances. Formally, given $\mathbf{z}_i$ and $\mathbf{z}_j$ as a positive pair from two views of T generated through data augmentation, the contrastive loss for this pair is defined as follows:

$$\ell_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right)/\tau\right)}{\sum_{k=1}^{2M} \mathbf{1}_{[k\neq i]} \exp\left(\operatorname{sim}\left(\mathbf{z}_i, \mathbf{z}_k\right)/\tau\right)} \tag{12}$$

where sim denotes the cosine similarity, and τ is the temperature hyperparameter. We average the loss for each pair within the batch as the final trajectory contrastive learning loss.

To tailor contrastive learning for trajectories, we devise two kinds of data augmentation strategies, namely trajectory cropping and segment masking. They are designed to emulate the trajectory characteristics inherent in the two-stage pipeline.

- **Trajectory Cropping.** Given a trajectory T, trajectory cropping involves removing a certain number of segments to generate $\hat{T}$. Specifically, we randomly sample the number of dropped segments from 1-3 and evenly remove locations to a total ratio of ρ_c. The resulting $\hat{T}$ can be represented as $\hat{T} = [l_{n_1}, l_{n_2}, ..., l_{n_{|\hat{T}|}}]$, where $|\hat{T}| = \lfloor |T| \times (1-\rho_c) \rfloor$.
- **Segment Masking.** Given a trajectory T, segment masking involves applying masking for a certain number of segments to generate $\hat{T}$. Specifically, we randomly sample the number of masked segments from 1-3 and evenly mask locations to a total ratio of ρ_c. The resulting $\hat{T}$ can be represented as $\hat{T} = [l_1, l_2, ..., l_{|T|}]$ with masked positions replaced by [MASK] tokens.

Notably, trajectory cropping produces trajectories that resemble the inputs in the detection stage in the detection stage for the identification of recovery positions, while segment masking produces trajectories that resemble the inputs in the recovery stage for the imputation of missing locations. The adoption of these strategies allows for the effective incorporation of contrastive learning into the trajectory recovery task, and thus facilitating the capture of sequential transition patterns and contributing to improved performance of the model. Note that we have additionally experimented with several other trajectory augmentation strategies [2], such as point shifting, temporal jittering, and trajectory pooling. However, we do not observe performance improvement.

4.6 Model Training

Recall that RETE model is utilized for both the detection stage and the recovery stage. We train a distinct model for each stage based on specific data characteristics required for each model input.

In the detection stage, a classification task is conducted for each data point in a trajectory to identify the number of missing locations succeeding this data point. Since the number of classes could become quite large if each potential number of missing locations is assigned a separate class, we propose to reduce the class number by partitioning the numbers into the non-overlapping bins (e.g., 0, 1-4, 5-8, etc). We then transform the initial label for each data point into the label of its corresponding bin and train the model to minimize classification error. This approach offers two primary benefits. First, it indicates that our focus is not to determine the exact number of locations, but to identify the approximate number within the interval. This not only aligns with the intuition that minor deviations in numbers tend to have little effect on the overall result, but also ensures to balance the difficulty of tasks between the first and the second stage. Second, adopting this paradigm greatly reduces the number of classes associated with all possible location numbers, which makes the the training of the model more feasible and tractable.

In the recovery stage, based on the predicted results for the bin class, we first insert [MASK] tokens with maximum possible number for the designated bin after this data point. For example, if the predicted bin corresponds to 1-4, we insert four [MASK] tokens. Subsequently, we train the model to accurately impute the missing location for each [MASK] token. When the actual number of missing locations is less than the maximum number for the specified bin, we train the model to fill in special [NULL] tokens that represent vacancies for those additional positions.

To further enhance the model performance, we incorporate the trajectory contrastive learning loss into the training process. Specifically, we employ trajectory cropping and segment masking strategies in the detection and recovery stages, respectively. The trajectory contrastive learning loss is jointly optimized with the respective loss for RETE model in each stage.

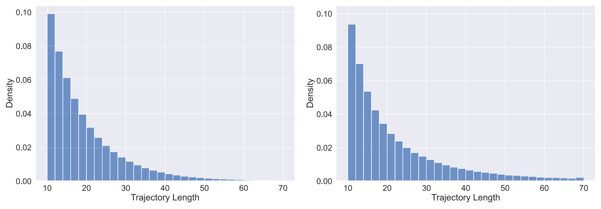
5 EXPERIMENTS

5.1 Experimental Settings

5.1.1 Datasets. We conduct the experiments using real-world trajectory datasets from three cities, namely Singapore (SG), Sydney (SYD), and Beijing (BJ). The Singapore and Sydney datasets, which include cellular trajectories, are provided by the largest telecommunication company in Singapore. Cellular trajectories record mobile connections to base stations from anonymous users. In this scenario, users automatically switch their cellular connections to another base station upon entering an area where that station provides mobile signals. Such switching events result in cellular trajectories with irregular intervals. As cellular trajectories can effectively reflect user mobility, we treat base stations as locations in the paper. For the Beijing dataset, we utilize the public T-drive dataset [43] which tracks trajectories of taxis with irregular intervals in Beijing. Since the records comprise continuous coordinates, we partition the city into 300m×300m grids to serve as locations and filter out the grids with fewer than 10 visits. For all the datasets, we retain trajectories that have a length of at least 10. Table 1 presents the statistics of the three datasets, and Figure 5 shows the distribution of trajectory length on the Singapore and Sydney datasets.

Table 1: Statistics of the datasets

Dataset	**SG**	**SYD**	**BJ**
Collection time	Apr 2022	Nov 2022	Feb 2008
# Trajectories	2,512,584	625,255	131,218
# Locations	11,633	18,426	9,936
Avg Traj. Length	18.98	21.54	49.68



(a) SG **(b) SYD**

Figure 5: Distribution of trajectory length.

5.1.2 Baseline Methods. As mentioned in Section 1, existing methods [29, 31, 36] assume that the recovery positions are provided in advance. Specifically, these methods employ masking on some locations with null placeholders and then aim to recover them, and thus they are not applicable when recovery positions are unavailable due to the constraints of their model designs. Therefore, given the more practical and realistic problem in this paper, we only compare with models that are directly applicable or can be suitably adjusted to our setting. Nevertheless, to illustrate the superiority of our framework, we also evaluate the performance against these previous methods in the setting where the oracle recovery positions are provided beforehand (Section 5.4). Specifically, we include three types of methods as baseline models, namely heuristic methods (LI and TraImpute), statistical learning methods (Markov), and Seq2seq models (LSTM, Transformer and SAGCopy) as follows:

- **LI**: it applies linear interpolation for distance consecutive data points, and selects locations within a specified distance from the interpolated line as the prediction.
- **TraImpute** [10]: it is a heuristic search algorithm which decides the imputed locations based on the frequencies of each location in the historical trajectories.
- **Markov** [24]: it is a statistical learning method that regards each location as a state and constructs a transition matrix for these states. We impute locations between consecutive data points if this increases the transition probability, and then we use the results with the highest probability as the prediction.
- **LSTM** [15]: it is a classical Seq2seq model which takes the original trajectory as input to the encoder and outputs the recovered trajectory in the decoder.

- **Transformer** [30]: it is a widely utilized Seq2seq model equipped with self-attention mechanism. The recovered trajectory is also produced in the decoder.
- **SAGCopy** [38]: it incorporates copy mechanism [12] to directly copy the input to the output result, and this method can alleviate the issue that the decoder fails to reconstruct the input trajectory.

5.1.3 Evaluation Metrics. Considering the fact that input trajectories are not guaranteed to be successfully reconstructed in Seq2seq models, we propose to design the metrics to evaluate different methods from two aspects: a macroscopic view and a microscopic view. For the macroscopic view, we evaluate how the recovered full trajectory $\overline{T}$ resembles the ground truth trajectory T. This implies that the model is required to accurately predict the missing locations while retaining the original locations of the input trajectory to avoid the ambiguous recovery results. Accordingly, we define two metrics: precision and recall for the full trajectory, denoted as Precision$_F$ (Prec.$_F$) and Recall$_F$ (Rec.$_F$), as follows: Precision$_F$ = $\frac{|T \cap \overline{T}|}{|\overline{T}|}$, Recall$_F$ = $\frac{|T \cap \overline{T}|}{|T|}$.

For the microscopic view, we are only interested in how the missing locations are recovered. Specifically, given a sparse trajectory $\tilde{T}$, and its associated ground truth full trajectory T and its predicted full trajectory $\overline{T}$ respectively, we denote the ground truth missing locations as $P = T - \tilde{T}$, and the predicted missing locations as $\overline{P} = \overline{T} - \tilde{T}$. To focus specifically on these missing locations, we define two additional metrics Precision$_M$ and Recall$_M$ to consider the missing locations as follows: Precision$_M$ = $\frac{|P \cap \overline{P}|}{|\overline{P}|}$, Recall$_M$ = $\frac{|P \cap \overline{P}|}{|P|}$. Based on the above metrics, we also report the $F1$ score for both micro and macro views defined as: $F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

5.1.4 Parameter Settings. In our experiments, we randomly sample 70%/10%/20% trajectories as training/validation/test examples for all the datasets. We keep the architecture of RETE model consistent for each stage, and it consists of four self-attention layers. For all the layers, the hidden size and the number of heads are set to be 128 and 4 respectively. We apply trajectory cropping and segment masking for the detection stage and recovery stage respectively. We set the number of GNN layers to be 3 in the collective transition pattern learning module. We set the ratio ρ in data augmentation to be 0.7, and size M to be 512 for the SG and SYD datasets and 128 for the BJ dataset in trajectory contrastive learning. We adopt Adam optimizer to train the model with a batch size 512 for the SG and SYD datasets and 128 for the BJ dataset, and the weight for the trajectory contrastive loss is set to 0.2 while the weight for classification and imputation task is set to 1. To ensure a fair comparison, all compared Seq2seq models use the same layer number and hidden size as defined in our framework.

5.2 Performance Comparison

Unlike the previous studies which adopt a fixed dropping ratio for all the trajectories, we embrace a more realistic setting that better reflects real-world scenarios where both the number of dropped segments and the number of missing locations may vary. Specifically, for each trajectory, the total dropping ratio is randomly selected in $[0.2, 0.3, 0.4, 0.5, 0.6]$, and the number of dropped segments is selected from 1-4. For example, if the dropping ratio is 0.4 and the number of dropped segments is 2, it indicates that two segments from a given trajectory are dropped, and each segment contains 20% of the data points in the original trajectory (i.e., 40% in total). This design principle aligns with our problem setting: we have no prior knowledge about the characteristics of sparse trajectories in practice, and thus no fixed parameters should not be predetermined. As a result, the model is required to handle a diverse range of trajectory recovery cases, rather than developing individual models for each dropping ratio.

5.2.1 Overall Performance. The overall results of all the compared methods are presented in Table 2. The improvement column indicates the performance of TERI relative to the second best models. Based on the results, we can make several observations.

First, Seq2seq models demonstrate lower performance in terms of Precision$_F$ and Recall$_F$ in most cases compared to heuristic and statistical learning methods. This is because Seq2seq models might not successfully reconstruct the whole original input trajectories in the decoding phase. The absence of some locations from the input could lead to lower metrics in the macroscopic view.

Second, despite the potential imperfect reconstruction issue, Seq2seq models outperform other baselines in terms of Precision$_M$ and Recall$_M$ in most cases. It indicates that Seq2seq models possess a superior capability to predict missing locations, as they are better equipped to capture sequential transition patterns owing to more powerful model capability to tackle sequential data than other baselines. Moreover, Transformer model is better than LSTM due to its enhanced expressive power for modeling sequential data, and SAGCopy alleviates the issue of imperfect reconstruction on two datasets. However, no single baseline model can consistently deliver the best performance.

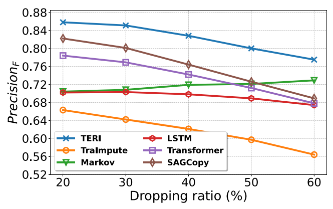
Third, out TERI framework achieves the best performance across all the metrics on these three datasets. The framework effectively combines the strengths of the two types of baselines: it avoids the issue of imperfect reconstruction in Seq2seq models while demonstrating the capability to model complex spatial and temporal correlations as well as capture complex sequential transition patterns.

5.2.2 Impact of Dropping Ratio. We examine the model performance on different dropping ratios by computing the average results on each metric for recovered trajectories. Figure 6– 7 show the four metrics of all the compared methods across different dropping ratios on the SG and SYD dataset respectively. Then we can make the following observations based on the results.

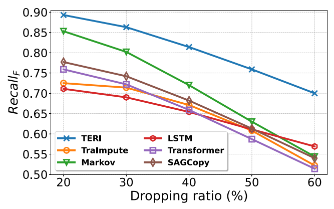
Regarding the macroscopic perspective, both Precision$_F$ and Recall$_F$ metrics demonstrate a downward trend as the dropping ratio increases. This is due to the increasing difficulty of the trajectory recovery problem with a higher level of sparsity. One exception is that Precision$_F$ for Markov method shows a slightly gradual increase, as Markov method tends to keep the original trajectory unchanged with a high dropping ratio. In general, heuristic methods are better than Seq2seq models for most of the dropping ratios. Regarding the microscope perspective, it is typically observed that Precision$_M$ increases with a higher dropping ratio, while Recall$_M$ conversely decreases. However, the decline in Recall$_M$ may be due to the increased challenge of recovering more detailed missing locations given the limited context information available in trajectories with higher dropping ratios.

Table 2: Performance comparison of different methods on trajectory recovery with irregular intervals. Bold scores denote the best in method group, and underlined scores denote the second best.

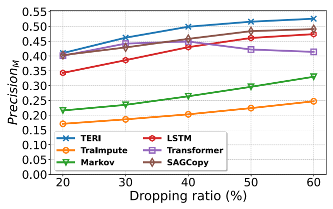
Methods	SG						SYD						BJ					
	Prec.$_F$	Rec.$_F$	F1$_F$	Prec.$_M$	Rec.$_M$	F1$_M$	Prec.$_F$	Rec.$_F$	F1$_F$	Prec.$_M$	Rec.$_M$	F1$_M$	Prec.$_F$	Rec.$_F$	F1$_F$	Prec.$_M$	Rec.$_M$	F1$_M$
LI	0.420	0.663	0.514	0.032	0.061	0.042	0.547	0.670	0.602	0.071	0.061	0.066	0.526	0.584	0.553	0.012	0.014	0.013
TraImpute	0.684	0.694	0.689	0.200	0.147	0.169	0.625	0.679	0.651	0.183	0.138	0.157	<u>0.536</u>	0.583	<u>0.559</u>	0.021	0.015	0.018
Markov	0.715	<u>0.709</u>	0.712	0.268	0.195	0.226	<u>0.658</u>	<u>0.706</u>	<u>0.681</u>	0.236	0.186	0.208	0.485	<u>0.591</u>	0.533	<u>0.058</u>	0.076	<u>0.066</u>
LSTM	0.693	0.646	0.669	0.419	<u>0.339</u>	<u>0.375</u>	0.533	0.478	0.504	<u>0.355</u>	0.181	<u>0.240</u>	0.241	0.417	0.305	0.032	0.036	0.034
Transformer	0.737	0.648	0.690	0.426	0.301	0.353	0.654	0.618	0.635	0.317	<u>0.188</u>	0.236	0.442	0.544	0.488	0.045	<u>0.081</u>	0.058
SAGCopy	<u>0.761</u>	0.674	<u>0.715</u>	<u>0.453</u>	0.299	0.360	0.451	0.556	0.498	0.267	0.165	0.204	0.475	0.537	0.504	0.051	0.064	0.057
TERI	**0.820**	**0.807**	**0.813**	**0.481**	**0.452**	**0.466**	**0.791**	**0.746**	**0.768**	**0.373**	**0.278**	**0.319**	**0.618**	**0.627**	**0.622**	**0.115**	**0.114**	**0.114**
Improv.	7.75%	13.82%	13.71%	6.18%	33.33%	24.27%	20.21%	5.67%	12.78%	5.07%	47.87%	32.92%	13.99%	6.09%	11.27	98.28%	40.74%	72.73%



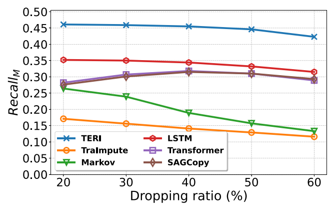
(a) Precision$_F$



(b) Recall$_F$

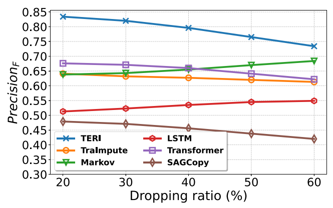


(c) Precision$_M$

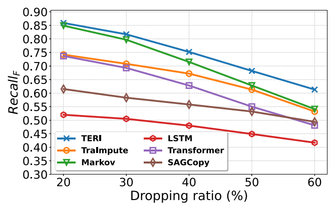


(d) Recall$_M$

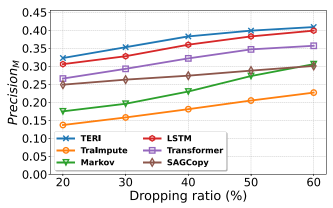
Figure 6: Results for different dropping ratios on SG dataset.



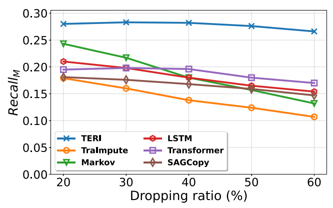
(a) Precision$_F$



(b) Recall$_F$



(c) Precision$_M$



(d) Recall$_M$

Figure 7: Results for different dropping ratios on SYD dataset.

Among all the methods, TERI consistently achieves the best performance across all the dropping ratios. This demonstrates that our designed modules are effective for various dropping ratios.

5.2.3 Impact of Trajectory Length. We conduct experiments to examine the model performance across various trajectory lengths. The results for F1 scores as comprehensive evaluation metrics are presented in Figure 8.

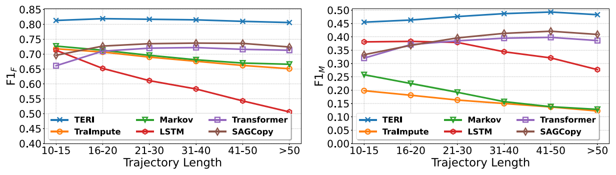
We can observe that distinct model types exhibit varied performance trends as the trajectory length increases. Specifically, the performance of non-learning-based methods and LSTM tends to gradually decrease, while Transformer-based methods (i.e., Transformer, SAGCopy and TERI) maintain a stable or even increased performance across different length ranges. This pattern can be likely attributed to the inherent capability of Transformer architecture to effectively model long sequences. Moreover, our proposed TERI consistently outperforms all the compared methods across all length ranges, thereby validating its superior performance for trajectories irrespective of their dropping ratios or length variations.

5.3 Model Analysis

5.3.1 Ablation Study. We conduct an ablation study by removing different model components to investigate their contributions to the performance. Specifically, we compare the TERI framework with the following variants:

- **TERI-T**: it removes the spatial and temporal encoding technique (Section 4.3) and applies solely the learnable positional embeddings [30] for each input position.
- **TERI-TG**: it further removes the collective transition pattern learning module (Section 4.4).
- **TERI-TGC**: it further removes the module of employing contrastive learning technique to enhance the model (Section 4.5).
- **TERI-B**: it adopts a different recovery process. It first employs binary classification on each location to identify recovery positions, and then uses a Transformer model to only recover the missing locations for each detected position in a Seq2seq manner.

Figure 9 presents the results of all model variants on the SG and SYD datasets. Based on the results, we can observe that the removal of different components from the framework leads to the decline of all the metrics, which demonstrates the contributions of each component to the model performance. In particular, it validates that our framework can effectively model the spatial and temporal correlations and as well as capture the sequential transition patterns. Moreover, as more components are removed from the model (-T, -G and -C), a more substantial decline in performance can be observed. This illustrates that each component has complementary effect to enhance the model performance. In addition, the framework becomes less effective when adopting an alternative form of recovery process (TERI-B) by first detecting the recovery position and then


(a) SG dataset

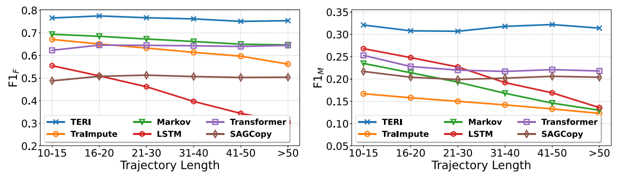
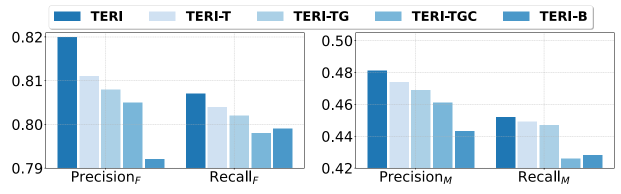

(b) SYD dataset

Figure 8: Results for different trajectory lengths.


(a) SG dataset

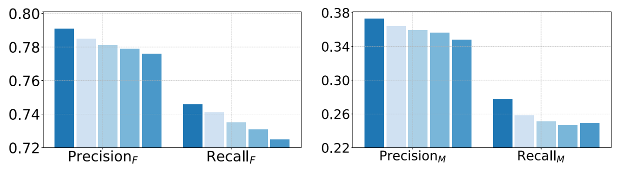

(b) SYD dataset

Figure 9: Results for ablation study.

recovering the missing locations with Seq2seq model. TERI-B is better than TERI-TGC in certain metrics because it incorporates both the spatial and temporal encoding module (-T) and the collective transition pattern learning module (-G), which are beneficial to the model performance. However, its performance still lags behind other model variants. This suggests that Seq2seq model is not the optimal choice for this task even if it can be adjusted to avoid the ambiguous results issue with a two-stage design.

5.3.2 *Time and Distance Encoding Techniques.*

We examine the effectiveness of time and distance encoding method utilized in the paper. Specifically, we replace the learnable Fourier features in TERI with four alternative techniques to encode time and distance information, while maintaining all other components unchanged. The details of these techniques are listed as follows:

- **Emb**: we partition the time and distance intervals into overlapping bins. Each bin is represented as an embedding, which then serves as an input to the Transformer encoder. We find that bins of 30-second interval for time and 200-meter interval for distance produce the best performance.
- **Attn-bias** [16]: we employ the interval-aware self-attention mechanism to transform the intervals to a bias term in attention score calculation.
- **Flashback** [40]: we compute correlation scores by considering time and distance intervals on hidden representations from the Transformer encoder. The scores are used to derive additional context representation at each time step through weighted sum.
- **CPE** [21]: we convert time and distance intervals into kernels, which are used to in convolution operations on location representations to incorporate spatio-temporal information.

Table 3: Results for time and distance encoding techniques.

Dataset	SG		SYD	
Metric	$F1_F$	$F1_M$	$F1_F$	$F1_M$
Emb	0.806	0.458	0.759	0.308
Attn-bias	0.803	0.454	0.762	0.303
Flashback	0.808	0.461	0.761	0.310
CPE	0.745	0.348	0.729	0.250
TERI	**0.813**	**0.466**	**0.768**	**0.319**

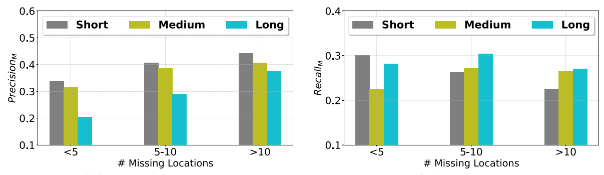

(a) Precision$_M$ (b) Recall$_M$

Figure 10: Results for different number of missing locations.

The results of the learnable Fourier features utilized in TERI in comparison to these methods are shown in Table 3. From the results, we can make the following observations. First, CPE yileds the worst performance, likely due to discrepancy in data format between its application and cellular trajectories. Second, the embedding method exhibits performance comparable to both Attn-bias and Flashback despite its simplicity. Flashback outperforms other baseline techniques across datasets. Third, the proposed technique in this study achieves the best performance, which demonstrates its effectiveness in directly tackling the continuous feature values.

5.3.3 *Missing Locations v.s. Trajectory Length.*

We further examine the performance of TERI on trajectories with varying length ranges, each characterized by distinct number of missing locations. This evaluation sheds light on questions such as the effectiveness on long trajectories with few missing locations. In particular, trajectories are classified based on their length into three groups: short (10-20), medium (20-50) and long (>50). Each group is further tested with three ranges of missing locations: <5, 5-10 and >10. Note that as long trajectories do not produce sparse trajectories with fewer than 10 missing locations, we re-process the test data to include these two ranges. Then the metrics of Precision$_M$ and Recall$_M$ on the SYD dataset are reported in Figure 10.

We observe that trajectories with varying length ranges exhibit distinct patterns. Our framework demonstrates superior performance on short trajectories with few missing locations. However, it tends to make conservative predictions when the proportion of missing locations increases, thus leading to increased Precision$_M$ and decreased Recall$_M$. Trajectories of medium length yield intermediate performance on these metrics. Interestingly, while long trajectories manifest a suboptimal performance on Precision$_M$, the framework produces commendable results on Recall$_M$. Given that the framework has never been trained on trajectories fewer than 10 missing locations, such results manifest its inherent robustness. As such, regardless of the trajectory length, our framework remains relatively stable across diverse ranges of missing locations.

5.3.4 Effect of Location Granularity. To evaluate the model performance on varying levels of location sparsity, we conduct experiments across different location granularities on the Beijing dataset. Specifically, we employ grid partitioning strategies with a range of grid sizes from 150m×150m to 300m×300m. The data processing procedure is the same as the one described in Section 5.1.1, and the results are presented in Table 4. We can observe that all the metrics increase as the granularity becomes more sparse, which is consistent with our intuition. Moreover, our model exhibits a modest decline in performance even when the granularity results in more than a four-time increase in location number. This demonstrates the robustness of TERI framework across different location granularities, and the model's adaptability to diverse datasets.

5.4 Performance with Oracle Recovery Positions

As discussed in Section 1, previous studies on network-free trajectory recovery are limited by the constraints that the recovery positions are provided in advance. Given that such oracle information is not available in real use cases, our problem formulation does not follow this assumption, and hence is not comparable with these methods due to their requirement of predetermined recovery positions. To conduct a more extensive evaluation on the superiority of our proposed framework, we investigate the performance of TERI framework against these methods under conditions where both the positions and the exact number of missing locations are given as prior information. In this specific scenario, our TERI framework reduces to employing only the second stage, and is directly comparable to several baselines, which are listed as follows:

- **AttnMove** [36]: it interprets each missing location as a token in the input, and employs a Transformer-based model with various attention mechanisms to capture the sequential transition patterns and impute the missing locations.
- **PeriodicMove** [29]: it employs a Transformer-based model with self-attention mechanism to model location correlations and periodicity patterns. It also applies graph neural networks for each individual trajectory to enrich transition patterns.
- **DHTR** [31]: it adopts the Seq2seq framework based on RNN to only recover the missing locations in the input during the decoding phase. The attention mechanism is modified to incorporate spatial and temporal information, and Kalman filter is utilized to calibrate the predicted results.
- **TrajFormer** [21]: it employs squeeze function within the Transformer self-attention mechanism to reduce computational complexity, and integrates spatial and temporal context with convolution operations to refine location representations.

It is worth noting that PeriodicMove and AttnMove require the historical trajectories of users to perform cross-attention in their methods. We exclude this module since our datasets do not allow user identification to extract historical trajectories due to privacy concerns. We use the same experimental setting as detailed in Section 5.2 to evaluate all the compared methods.

The results on the SG and SYD datasets are shown in Table 5. We can observe that Periodic achieves the worst performance while AttnMove performs the best among the baselines. This indicates

Table 4: Results for different location granularities

Size (m)	# locations	Metric			
		Precision$_F$	Recall$_F$	Precision$_M$	Recall$_M$
150	25837	0.571	0.527	0.087	0.077
225	15006	0.589	0.529	0.092	0.089
300	9936	0.618	0.627	0.115	0.114
375	7098	0.621	0.633	0.124	0.128
450	5317	0.712	0.692	0.135	0.139

Table 5: Results with oracle recovery positions.

Dataset	SG		SYD	
Metric	Precision$_M$	Recall$_M$	Precision$_M$	Recall$_M$
AttnMove	0.536	0.528	0.399	0.368
PeriodicMove	0.365	0.329	0.289	0.257
DHTR	0.405	0.384	0.344	0.316
TrajFormer	0.465	0.404	0.356	0.303
TERI	**0.552**	**0.544**	**0.412**	**0.388**

that Transformer-based framework is able to achieve satisfactory performance due to its superior power of modeling sequential data. However, applying GNN individually to each trajectory as opposed to the global transition graph introduces noise in capturing transition patterns, thereby leading to a significant performance drop for PeriodicMove. In addition, compared to Seq2seq framework employed in DHTR, it is more effective to regard trajectory recovery as an imputation process, as adopted in AttnMove, rather than as a sequence generation process. Given that TrajFormer is not designed for the recovery task, its modules exhibit suboptimal performance compared to the results in the original intended scenario. Most importantly, TERI outperforms all the baselines in the scenario where oracle recovery positions are provided. This demonstrates that our framework at effective at capturing not only complex spatial and temporal correlations, but also sequential transition patterns, thus enhancing Transformer encoder for both problem settings.

5.5 Model Efficiency

We further evaluate model efficiency in terms of training and inference time. Training time denotes the time required for training an epoch, while inference time refers to the time required to recover a batch of trajectories. We use batch size of 128 and compute the average inference time per batch on the test set for different methods. The performance for the compared methods on the SG and SYD datasets is presented in Figure 12.

For inference time, we can observe that Markov method is the most efficient as it can refer to the pre-stored transition probability matrix in search, while LI and TraImpute adopt dynamic search algorithm, leading to larger execution time. Moreover, Transformer-based models take more time compared to LSTM due to the relatively short sequence length. Remarkably, despite its two-stage pipeline, TERI shows comparable efficiency with the Transformer model. This can be attributed to the fact that the auto-regressive paradigm applied in imputation/generation process in TERI/Transformer requires multiple steps in decoding stage, thus dominating the execution time. In addition, SAGCopy is the least efficient method, as the copy mechanism applied in the model requires substantial time at every decoding step. For training time, the pattern is similar to that of inference time. Since our framework consists of two stages

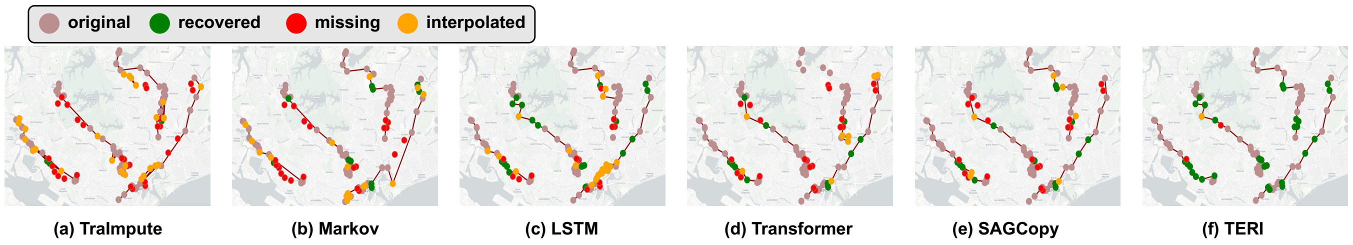
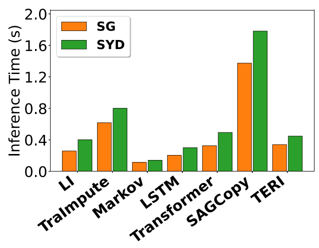
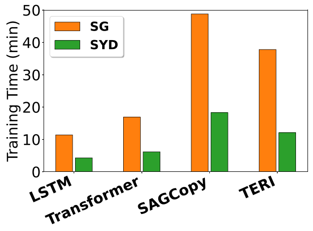


Figure 11: Case study for trajectory recovery on SG dataset. Dots are colored to denote different types of locations (*Brown*: the original locations present in input trajectories; *Green*: the correctly recovered locations; *Red*: the locations that are not successfully recovered; *Yellow*: the interpolated locations predicted by the model, but not in the ground truth trajectories).

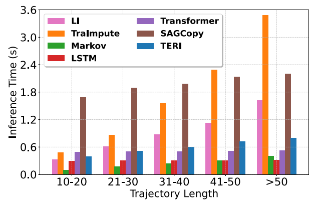


(a) Inference time

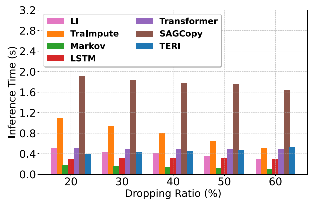


(b) Training time

Figure 12: Results for model efficiency.



(a) Varying trajectory lengths



(b) Varying dropping ratios

Figure 13: Inference time for different cases.

for training, which inevitably leads to a more extended training time compared to single Transformer or LSTM model. Moreover, SAGCopy requires the highest training time due to the inefficiency of its copy mechanism. Nevertheless, considering that the training process is executed only once, and the inference time satisfies the business requirement, the amount of time for training and inference for TERI is acceptable in practice.

To investigate the inference time on diverse trajectory characteristics, we further extend the evaluation to include varying trajectory lengths and dropping ratios per batch on the SYD dataset. The results are shown in Figure 13. We can observe that with the increase of trajectory length, all the methods need increased processing time. Furthermore, search algorithms demonstrate a heightened sensitivity to trajectory length compared to deep learning models. On the other hand, as the dropping ratio increases, all the methods, with the exception of TERI framework, require less inference time. This can be attributed to their decreased capability to identify recovery positions, resulting in few recovery attempts in the decoding or search phrase. In contrast, TERI tends to perform more imputation operations, which lead to increased inference time. In summary, compared to all the baselines, while TERI does not demonstrate optimal efficiency, it achieves a favourable trade-off between effectiveness and efficiency to satisfy the needs in downstream services.

5.6 Visualization and Case Study

We perform qualitative analysis by visualizing the recovery results for four trajectories in various regions within Singapore, as illustrated in Figure 11. Then we can make several observations.

First, despite the ability of TraImpute and Markov to preserve all original locations from the input trajectories in the prediction results, they are not effective in recovering missing locations due to their limited model capability on capturing complex patterns. Second, deep learning methods can recover larger number of correct locations compared to heuristic and statistical learning methods. However, as mentioned previously, these Seq2seq models often struggle with reconstructing the original locations, as illustrated by the upper trajectory in Figure 11(c) and (d). Third, since SAGCopy adopts the copy mechanism, it is able to avoid the bad examples encountered by LSTM and Transformer. Lastly, it is obvious that TERI overcomes the issue in baselines and yields superior performance on all these trajectories.

6 CONCLUSION

Trajectory recovery is a critical task to provide high-quality trajectory data sources for downstream applications. While extensive studies have been proposed to tackle this problem, they make unrealistic assumptions that the recovery positions are known in advance. In this paper, we study a more realistic setting where no prior information is provided. Specifically, we propose our TERI framework consisting of two stages: the detection stage to identify the recovery positions, and the recovery stage to impute the missing locations. Both stages are built upon Transformer encoder and equipped with novel designs that enhance the modeling of spatial and temporal correlation, and the capture of sequential transition patterns. Experimental results show that TERI outperforms baseline methods in the new problem setting.

ACKNOWLEDGMENTS

This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from Singapore Telecommunications Limited (Singtel), through Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU). This research is supported in part by MOE Tier-2 grant MOE-T2EP20221-0015.

REFERENCES

[1] Prithu Banerjee, Sayan Ranu, and Sriram Raghavan. 2014. Inferring Uncertain Trajectories from Partial Observations. In *ICDM*. 30–39.

[2] Yanchuan Chang, Jianzhong Qi, Yuxuan Liang, and Egemen Tanin. 2023. Contrastive Trajectory Similarity Learning with Dual-Feature Attention. In *ICDE*. 2933–2945.

[3] Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016. Learning Points and Routes to Recommend Trajectories. In *CIKM*. 2227–2232.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*. 1597–1607.

[5] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. 2021. Robust Road Network Representation Learning: When Traffic Patterns Meet Traveling Semantics. In *CIKM*. 211–220.

[6] Yile Chen, Xiucheng Li, Gao Cong, Cheng Long, Zhifeng Bao, Shang Liu, Wanli Gu, and Fuzheng Zhang. 2021. Points-of-Interest Relationship Inference with Spatial-enriched Graph Neural Networks. *Proc. VLDB Endow.* 15, 3 (2021), 504–512.

[7] Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. 2023. RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer. In *ICDE*.

[8] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular routes from trajectories. In *ICDE*. 900–911.

[9] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. 1724–1734.

[10] Mohamed M. Elshrif, Keivin Isufaj, and Mohamed F. Mokbel. 2022. Network-less trajectory imputation. In *SIGSPATIAL*. 8:1–8:10.

[11] Aristides Gionis, Theodoros Lappas, Konstantinos Pelechrinis, and Evimaria Terzi. 2014. Customized tour recommendations in urban areas. In *WSDM*. 313–322.

[12] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*.

[13] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian S. Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *ICDE*. 1073–1084.

[14] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2022. Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting. *IEEE Trans. Knowl. Data Eng.* 34, 11 (2022), 5415–5428.

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

[16] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. 2023. Self-supervised Trajectory Representation Learning with Temporal Regularities and Travel Semantics. In *ICDE*. 843–855.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[18] Xiucheng Li, Gao Cong, and Yun Cheng. 2020. Spatial Transition Learning on Road Networks with Deep Probabilistic Models. In *ICDE*. 349–360.

[19] Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. 2019. Learning Travel Time Distributions with Deep Generative Model. In *WWW*. 1017–1027.

[20] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. 2021. Learnable Fourier Features for Multi-dimensional Spatial Positional Encoding. In *NeurIPS*. 15816–15829.

[21] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. 2022. TrajFormer: Efficient Trajectory Classification with Transformers. In *CIKM*. 1229–1237.

[22] Zongyu Lin, Guozhen Zhang, Zhiqun He, Jie Feng, Wei Wu, and Yong Li. 2021. Vehicle Trajectory Recovery on Road Network Based on Traffic Camera Video Data. In *SIGSPATIAL*. 389–398.

[23] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M. Ni. 2013. Finding time period-based most frequent path in big trajectory data. In *SIGMOD*. 713–724.

[24] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *Ubicomp*. 911–918.

[25] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL*. 336–343.

[26] Ali Rahimi and Benjamin Recht. 2007. Random Features for Large-Scale Kernel Machines. In *NIPS*. 1177–1184.

[27] Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. 2021. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. In *KDD*. 1410–1419.

[28] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. 2013. Calibrating trajectory data for similarity-based analysis. In *SIGMOD*. 833–844.

[29] Hao Sun, Changjie Yang, Liwei Deng, Fan Zhou, Feiteng Huang, and Kai Zheng. 2021. PeriodicMove: Shift-aware Human Mobility Recovery with Graph Neural Network. In *CIKM*. 1734–1743.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[31] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. 2021. Deep Trajectory Recovery with Fine-Grained Calibration using Kalman Filter. *IEEE Trans. Knowl. Data Eng.* 33, 3 (2021), 921–934.

[32] Jingyuan Wang, Ning Wu, and Wayne Xin Zhao. 2022. Personalized Route Recommendation With Neural Network Enhanced Search Algorithm. *IEEE Trans. Knowl. Data Eng.* 34, 12 (2022), 5910–5924.

[33] Hao Wu, Jiangyun Mao, Weiwei Sun, Baihua Zheng, Hanyuan Zhang, Ziyang Chen, and Wei Wang. 2016. Probabilistic Robust Route Recovery with Spatio-Temporal Dynamics. In *KDD*. 1915–1924.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24.

[35] Dongbo Xi, Fuzhen Zhuang, Yanchi Liu, Jingjing Gu, Hui Xiong, and Qing He. 2019. Modelling of Bi-Directional Spatio-Temporal Dependence and Users' Dynamic Preferences for Missing POI Check-In Identification. In *AAAI*. 5458–5465.

[36] Tong Xia, Yunhan Qi, Jie Feng, Fengli Xu, Funing Sun, Diansheng Guo, and Yong Li. 2021. AttnMove: History Enhanced Trajectory Recovery via Attentional Network. In *AAAI*. 4494–4502.

[37] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *ICLR*.

[38] Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-Attention Guided Copy Mechanism for Abstractive Summarization. In *ACL*. 1355–1362.

[39] Can Yang and Gyözö Gidófalvi. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *Int. J. Geogr. Inf. Sci.* 32, 3 (2018), 547–570.

[40] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudré-Mauroux. 2020. Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States!. In *IJCAI*. 2184–2190.

[41] Fudan Yu, Wenxuan Ao, Huan Yan, Guozhen Zhang, Wei Wu, and Yong Li. 2022. Spatio-Temporal Vehicle Trajectory Recovery on Road Network Based on Traffic Camera Video Data. In *KDD*. 4413–4421.

[42] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2021. An Effective Joint Prediction Model for Travel Demands and Traffic Flows. In *ICDE*. 348–359.

[43] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *KDD*. 316–324.

[44] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing Uncertainty of Low-Sampling-Rate Trajectories. In *ICDE*. 1144–1155.