Contrastive Pre-training of Spatial-Temporal Trajectory Embeddings

Yan Lin, Huaiyu Wan, Shengnan Guo, Youfang Lin

Abstract—Pre-training trajectory embeddings is a fundamental and critical procedure in spatial-temporal trajectory mining, and is beneficial for a wide range of downstream tasks. The key for generating effective trajectory embeddings is to extract high-level travel semantics from trajectories, including movement patterns and travel purposes, with consideration of the trajectories' long-term spatial-temporal correlations. Despite the existing efforts, there are still major challenges in pre-training trajectory embeddings. First, commonly used generative pretext tasks are not suitable for extracting high-level semantics from trajectories. Second, existing data augmentation methods fit badly on trajectory datasets. Third, current encoder designs fail to fully incorporate long-term spatial-temporal correlations hidden in trajectories. To tackle these challenges, we propose a novel Contrastive Spatial-Temporal Trajectory Embedding (CSTTE) model for learning comprehensive trajectory embeddings. CSTTE adopts the contrastive learning framework so that its pretext task is robust to noise. A specially designed data augmentation method for trajectories is coupled with the contrastive pretext task to preserve the high-level travel semantics. We also build an efficient spatial-temporal trajectory encoder to efficiently and comprehensively model the long-term spatial-temporal correlations in trajectories. Extensive experiments on two downstream tasks and three real-world datasets prove the superiority of our model compared with the existing trajectory embedding methods.

Index Terms—Representation learning, spatial-temporal data, trajectory embedding, pre-training, self-supervised learning.

1 Introduction

With the rapid growth of location-based services and the wide spreading of GPS-equipped devices, recent years witness increasing availability of trajectory data, including taxi trajectories, check-ins to point-of-interests, cellular signaling sequences, etc. As a result, mining spatial-temporal trajectory data has been extensively studied, such as predicting future trajectories [1], estimating the time of arrival [2], [3] and trajectory clustering [4], [5]. Among these researches, learning embeddings of trajectories is a fundamental and critical issue, since the accuracy and comprehensiveness of embeddings are of crucial importance for achieving good performances in downstream tasks. Existing trajectory mining methods mostly adopt the end-to-end embedding strategy, which usually trains a trajectory encoder with task-specific learning objectives [6]. Yet, we argue that the pre-training trajectory embedding methods deserve more attention. Firstly, they can make use of widely existing unlabeled trajectory datasets, and explore the universal spatial-temporal patterns underling trajectories. These patterns can promote the prediction accuracy and generalization performance of downstream tasks. Secondly, they can be fine-tuned to suit a wide variety of downstream tasks with just a small-scale labeled dataset, or directly applied to unsupervised tasks. This can make the downstream prediction models train faster, and improve the overall computational efficiency.

The core idea of pre-training spatial-temporal trajectory embeddings is to design a self-supervised pretext task to pre-train a trajectory encoder, so that the embedding method can extract high-level travel semantics from unlabeled trajectory datasets, including users or vehicles' spatial-temporal moving patterns and travel purposes. This information can then be utilized by downstream models to perform prediction or classification tasks.

In recent years, some methods have been proposed to pre-train general trajectory embeddings for various downstream tasks. t2vec [7] adopts a generative auto-encoding-based objective to pre-train a recurrent neural network (RNN) [8] trajectory encoder. The encoder can then be used for generating latent embedding vectors for trajectories. GM-VSAE [5] extends the above idea by designing a variational encoder to cast trajectory embeddings into a multidimensional Gaussian space, so that a high-performance online anomalous trajectory detection task can be performed. traj2vec [4] and TremBR [9] further incorporate some aspects of spatial-temporal information in trajectories into the embeddings by special trajectory encoder designs. These researches demonstrate that the pre-trained trajectory embeddings are indeed beneficial for downstream tasks.

Despite the existing efforts, there are still some challenges lie in pre-training spatial-temporal trajectory embeddings.

First, generative self-supervised pretext tasks aim to recover all the details of a trajectory, so that they fail to extract the high-level semantics of trajectories. Since trajectories are sequences, existing pre-training trajectory embedding methods mostly design their pretext tasks following the language models in natural language processing (NLP) [10], which uses generative auto-regressive or auto-

Corresponding author: Shengnan Guo

- *Yan Lin, Huaiyu Wan, Shengnan Guo and Youfang Lin are with the Beijing Key Laboratory of Traffic Data Analysis and Mining, School of Computer and Information Technoloty, Beijing Jiaotong University, Beijing 100044, China, and the Key Laboratory of Intelligent Passenger Service of Civil Aviation, CAAC, Beijing, 101318, China.*
  E-mail: hywan@bjtu.edu.cn; ylincs@bjtu.edu.cn; guoshn@bjtu.edu.cn; yflin@bjtu.edu.cn.

Manuscript received xx xx, xxxx; revised xx xx, xxxx.

encoding self-supervised objectives. Specifically, the input trajectory sequence is encoded into a latent representation through a RNN-based encoder, then is recovered to its original form relying on the latent representation. We argue that this type of pretext tasks is not suitable for pre-training trajectory embedding vectors. As shown in Figure 1(a), the process of recovering the original input sequence will force the result embedding focus on reflecting all the details contained in a trajectory [11], which violates trajectory embedding learning's goal, i.e., modeling high-level semantics. Generative pretext tasks can also lead the embeddings be sensitive to noise, which is common in real-world trajectory datasets. By comparison, the contrastive pretext task aims to model the relationship between trajectory samples by contrasting them in the high-level embedding space, as demonstrated in Figure 1(b). Theoretically, it can effectively extract trajectories' high-level travel semantics and is robust to the widely existing noise. Yet, the contrastive learning framework is not implemented in existing pre-training trajectory embedding methods.

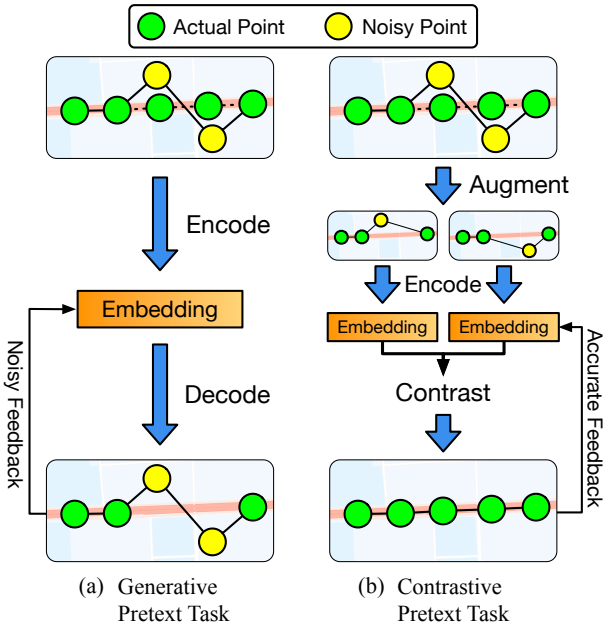


Fig. 1. Robustness of generative and contrastive pretext tasks.

Second, data augmentation methods that can preserve high-level semantics is crucial for constructing contrastive pretext task. Yet, existing data augmentation methods do not apply well to trajectories. As shown in Figure 1(b), data augmentation can decide the quality of feedback information in contrastive learning. Contrastively pre-training embedding methods in computer vision (CV) [12] and Natural Language Processing (NLP) [13] have developed several data augmentation methods to extract high-level information from unlabeled datasets, including adding random noise, cropping, reversing and so on [14]. Yet, as demonstrated in Figure 2, when applied to trajectories, these augmentation methods will fundamentally change the high-level information of trajectories, which indicates that they are not suitable for trajectory data. How to design a data augmentation method that can preserve and extract high-level semantics and the spatial-temporal patterns embedded in trajectories is challenging.

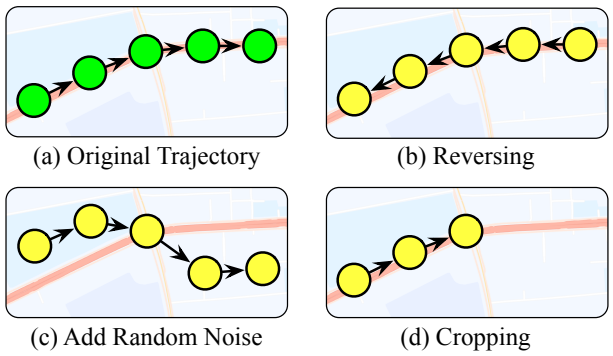


Fig. 2. Existing data augmentation methods applied to trajectories.

Third, trajectories contain long-term spatial-temporal correlations, which are not sufficiently modeled. Trajectories are often long sequences, with complex spatial-temporal correlations. Comprehensively modeling these correlations through the specific design of the trajectory encoder is essential for trajectory embedding methods to gain satisfactory results. As shown in Figure 3, we can consider the spatial-temporal correlations in trajectories from two aspects, i.e., absolute and relative, and both aspects can infer the trajectories' moving patterns and travel purposes. Yet, existing methods fail to explicitly incorporate both the two aspects of spatial-temporal correlations into consideration. traj2vec [4] and TremBR [9] only incorporate relative spatial-temporal properties by calculating the differences in a set of attributes between consecutive trajectory points. This limits the comprehensiveness of the learned trajectory embeddings. On the other hand, the most widely adopted RNN-based trajectory encoder often suffers from the vanishing gradient issue [15], while the vanilla Transformer encoder [16] has squared time and memory complexity. Thus, they are not suitable for modeling long-term correlations in trajectories. How to build a comprehensive trajectory encoder with the characteristics of spatial-temporal trajectories in mind is another challenge.

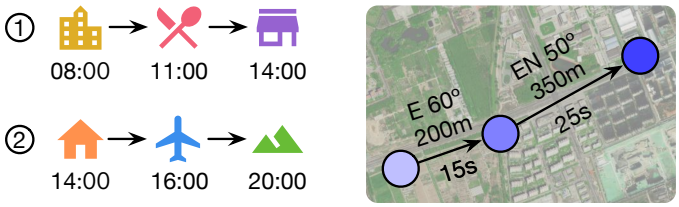


Fig. 3. Two aspects of spatial-temporal information embedded in trajectories.

In order to tackle the above-discussed challenges in pre-training trajectory embeddings, we aim to utilize the contrastive learning framework [14], [17] to construct a self-supervised model for efficiently pre-training comprehensive trajectory embeddings. In this paper, we propose a novel *Contrastive Spatial-Temporal Trajectory Embedding* (**CSTTE**)

model. To make the embedding model better at extracting high-level travel semantics of trajectories and more robust to noise, we construct our self-supervised pretext task under the contrastive learning framework, coupled with a specially designed data augmentation method for trajectories. We then propose an attention-based trajectory encoder with a spatial-temporal encoding layer and globally-shared attention anchors as the pre-training target. This makes our embedding method capable of modeling long-term spatial-temporal correlations in trajectories, while still being computationally efficient. The main contributions of this paper are summarized as follows:

- We propose a pre-training spatial-temporal trajectory embedding model CSTTE based on the contrastive learning framework. Its self-supervised pretext task is robust to noise and can extract high-level travel semantics from trajectories.
- A data augmentation method is designed to preserve the high-level travel semantics embedded in trajectories. It samples sub-trajectories from the trajectory dataset for the embedding method to be trained contrastively.
- An efficient spatial-temporal trajectory encoder is built to comprehensively and efficiently model the long-term spatial-temporal correlations in trajectories. It can generate latent embeddings for trajectories by incorporating comprehensive information to serve downstream tasks.
- We apply our pre-training embedding method to two downstream tasks, and conduct experiments on three real-world trajectory datasets. The experimental results prove that the embeddings generated from CSTTE can indeed help downstream models to improve the prediction performance.

2 Related Work

2.1 Learning Embeddings for Spatial-Temporal Data Mining

Embedding learning is a fundamental research topic in neural network data mining, for most neural network models are feature-based, and require the input targets to be represented by latent embedding tensors. In spatial-temporal data mining, embedding learning is also a crucial question. Spatial-temporal objects, like locations, road networks and trajectories, often contain complex spatial-temporal information, while geographically and dynamically correlated with other objects. How to capture and fuse the above-mentioned spatial-temporal information into embedding learning methods is challenging.

Locations are fundamental units in spatial-temporal datasets, and location embedding techniques are widely used in spatial-temporal data mining models. A straightforward idea is to use an index-fetching embedding layer [1], [6] since locations are often presented by discrete indices such as POIs and road segment IDs. Yet, this kind of embedding is trained through task-specific supervision, cannot properly capture the comprehensive spatial-temporal correlations between locations, and often require a large-scale labeled dataset for optimum performance. To tackle the above problems, DeepMove [18] implements word2vec [19] on unlabeled movement datasets to pre-train a set of embedding vectors for POIs. POI2Vec [20] and TALE [21] further incorporates spatial and temporal correlations between locations into the embedding vectors and gain better results. CTLE [22] migrates the idea of contextual embedding in language models into location embedding. The above pre-training methods can utilize large-scale unlabeled spatial-temporal datasets to obtain general location embedding vectors, and apply them to a wide variety of tasks, including location recommendation [23], [24] and location classification [25], [26].

Road segments or intersections in road networks can also be seen as locations, yet they contain more complex information due to the topology structure of road networks. Only using index-fetching embedding layers for road network representation will discard its intricate yet important topology information. IRN2Vec [27] proceeds random walking on road networks coupled with word2vec to incorporate topology correlations into road segment embedding. Toast [28] further utilizes real-world trajectories to fuse network topology with traffic patterns. HRNR [29] proposes a hierarchical graph neural network and an auto-encoding pre-training objective to learn comprehensive embedding vectors for road segments. Akin to pre-trained location embeddings, these pre-trained road network embeddings can be applied to various tasks, such as route planning [30], [31] and estimated time of arrival [2], [32].

In this paper, we focus on pre-training spatial-temporal trajectory embeddings. Compared to location embeddings and road network embeddings which aim to learn information for discrete units in spatial-temporal datasets, trajectory embeddings pay more attention to the high-level semantic information reflected in a series of movements as a whole, including moving patterns and travel purposes. The next section gives a summary of existing works in pre-training trajectory embeddings.

2.2 Pre-training Trajectory Embedding

The quality of trajectory embeddings is essential for achieving good performance in trajectory mining tasks. The most widely applied strategy for trajectory embedding is to construct an end-to-end sequence encoder for generating latent representations [1], [6]. Yet, trajectory embedding vectors generated this way often lack generalization, thus hard to migrate to other models or tasks. This strategy can also be unworkable when lacks labeled data, such as in trajectory similarity measurement.

To tackle the above problems, there is a rising interest in pre-training trajectory embeddings with self-supervised learning objectives. t2vec [7] and GM-VSAE [5] build an RNN-based trajectory encoder and a generative pre-training objective to infer the probability distribution of trajectories based on sequential correlations of visited locations. Trajectories also contain complex spatial-temporal properties which shouldn't be ignored. TremBR [9] incorporates temporal information by concatenating the travel time of each trajectory point with its latent embedding vector. traj2vec [4] considers relative spatial-temporal properties by taking the differences in a set of attributes between consecutive

trajectory points as the input features. Experimental results show the spatial-temporal information is indeed helpful for generating higher quality trajectory embedding vectors. Yet, considering that there are complex long-term spatial-temporal correlations embedded in trajectories, existing methods are not comprehensive enough.

One of the core components of trajectory embedding methods is the pretext task. Most existing methods design their pretext tasks using generative objectives, including auto-encoding [33] and auto-regressive [34]. Yet, this type of objective involves reconstructing input features, which will lead the embeddings to reflect all the details contained in trajectories [11]. Contrastive learning [14] in computer vision inspired us to propose a self-supervised learning objective that can better extract the high-level sequential information from trajectories.

3 PRELIMINARIES

Definition 1. **Visiting Record.** In location-based services and GPS record datasets, a person or a vehicle's visit to a certain location is represented by a visiting record $r = (l, t, c_x, c_y)$, which indicates that location l is visited at time t. l is a discrete location index indicator, such as a POI index or a grid index. c_x and c_y are the longitude and latitude of the visited location, and t is the visited timestamp.

Definition 2. **Spatial-Temporal Trajectory.** The movements of an object during a certain period can be represented by a list of sequential visiting records, which we called a spatial-temporal trajectory. We denote a trajectory as $s = \{r_1, r_2, \ldots, r_N\}$, where the visiting records are ordered by their visited time, and N is the length of the trajectory. We denote the set of all trajectories in a dataset as S.

Problem Statement. *Pre-training Trajectory Embeddings.* Given a set of spatial-temporal trajectories S, we aim to pre-train a trajectory encoder f to generate a embedding vector $\boldsymbol{e}_s$ given the trajectory s, i.e., $\boldsymbol{e}_s = f(s)$. This encoder should be pre-trained with a pretext task and a data augmentation method with no requirement for task-specific labels.

4 THE CSTTE MODEL

4.1 Overall Contrastive Framework

The overall framework of our proposed model is demonstrated in Figure 4. Following the contrastive learning framework [17], the overall model aims to achieve a self-supervised pretext task. That is, to maximize the similarity of embeddings between trajectory samples that represent similar high-level travel semantics, and to maximize the difference of embeddings between those that represent dissimilar semantics. To generate trajectory samples that represent similar and dissimilar high-level travel semantics, we design a trajectory data augmentation method to sample sub-trajectories from the original trajectory dataset. Two sub-trajectories sampled from the same trajectory are regarded as the query and the positive sample, respectively; sub-trajectories sampled from other trajectories are defined as the negative samples with regard to the query. Finally, to effectively and efficiently model the long-term spatial-temporal correlations embedded in trajectories, we build a trajectory encoder to cast the input trajectory record sequences into fixed-length embedding vectors. During self-supervised pre-training, the encoder is fed with the above sub-trajectory samples to calculate embeddings that correspond to these samples. The contrastive learning-based pretext task is then applied to guide the training of the trajectory encoder. After the pre-training, the result trajectory encoder can be used to generate spatial-temporal trajectories' embeddings for downstream tasks. The following content explains our proposed model in detail.

4.2 Contrastive Pretext Task

Self-supervised pretext tasks are the core of pre-training embedding methods. To learn high-quality trajectory embeddings, it is essential for the pretext task to extract high-level travel semantics from trajectories. Most existing pre-training trajectory embedding methods follow the language models in NLP, and design their pretext tasks based on generative objectives like auto-encoding and auto-regressive. Yet, generative pretext tasks are not suitable for extracting high-level semantics from trajectories, rather they will lead the learned embeddings to reflect all the details contained in the trajectories, as shown in Figure 1(a). To this end, we demonstrate our pretext task, which is based on the contrastive learning framework [17].

To extract high-level travel semantics from trajectories and embed them into the learned embeddings, we aim to design a pretext task that minimizes the distance between trajectory samples that represent similar high-level semantics in the embedding space, and maximizes the distance between those that represent dissimilar high-level semantics, as shown in Figure 4. Formally, we define three types of trajectory samples: the query s^q, the positive sample s^{k_+}, and the negative samples $s^{k_i}, i \in \{1, 2, \ldots, n_{\text{neg}}\}$, where n_{neg} is the number of negative samples. Ideally, the positive sample shares similar high-level travel semantics with the query, while the negative samples represent different semantics from the query. Since we aim to model their relationships in the embedding space, a trajectory encoder f is utilized to encode these samples into their corresponding embeddings:

$$\boldsymbol{q} = f(s^q), \boldsymbol{k}_+ = f(s^{k_+}), \boldsymbol{k}_i = f(s^{k_i}), \tag{1}$$

where $\boldsymbol{q}, \boldsymbol{k}_+, \boldsymbol{k}_i \in \mathbb{R}^d$, d is the dimension of embedding vectors. Next, to capture the relationships between these embeddings and to implement our contrastive pretext task, we adopt the InfoNCE loss [11]:

$$\mathcal{L} = -\log \frac{\exp(\boldsymbol{q}^\top \boldsymbol{k}_+/\tau)}{\sum_{\boldsymbol{k}} \exp(\boldsymbol{q}^\top \boldsymbol{k}/\tau)}, \boldsymbol{k} \in \{\boldsymbol{k}_+, \boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_{n_{\text{neg}}}\}, \tag{2}$$

where τ is the temperature hyper-parameter. The InfoNCE loss can be viewed as discriminating positive samples from the set of all samples with regard to the query. In this way, the learned embeddings can reflect the similarity or dissimilarity of trajectory samples' high-level semantics, thus leading the embedding method to extract the semantic information from trajectories. After the contrastive pre-training, we can utilize the learned trajectory encoder for generating trajectory embeddings as $\boldsymbol{e}_s = f(s)$.

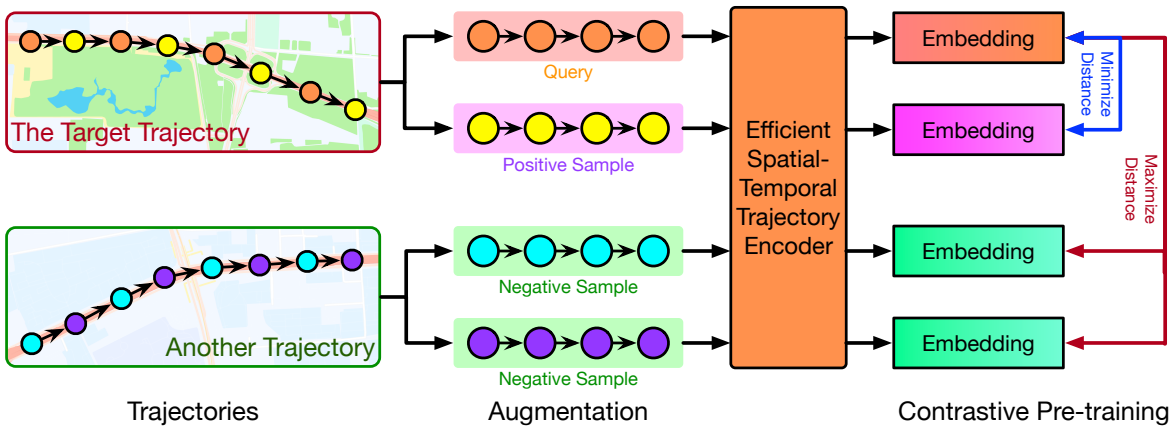


Fig. 4. The overall framework of our proposed model.

To complete the contrastive pretext task, two questions remain: 1. How to create trajectory samples s^q, s^{k_+} and s^{k_i}, i.e., what data augmentation method to use on the trajectory dataset; 2. How to implement the trajectory encoder f. In the following sections, we will answer these questions by introducing the data augmentation method we used for creating high-level semantic-preserved trajectory samples, and designing our trajectory encoder for modeling long-term spatial-temporal correlations.

4.3 Trajectory Data Augmentation

The data augmentation method is one of the basic components of the contrastive learning framework. Trajectory augmentation methods need to preserve the high-level travel semantics of trajectories so that they can be incorporated during pre-training. Yet, as we can see in Figure 2, popular data augmentation methods will fundamentally change the high-level semantics of trajectories. In this section, we introduce our proposed trajectory data augmentation method that can better suit the scenario of contrastive trajectory embedding learning.

In the real-world, the movement of an object always follows a continuous path, while visiting records in trajectory datasets can be seen as discrete samples of the path [7]. The underlying path of a trajectory reflects its high-level travel semantics, regardless of the sample rates. Thus, multiple sub-trajectories re-sampled from one trajectory correspond to the same underlying path and point to similar high-level semantics, as demonstrated in Figure 5. Based on the above analysis, we design our trajectory data augmentation method based on the 2-hop sampling. Formally, given a trajectory $s = \{r_1, r_2, \ldots, r_N\}$, we generate the query s^q and the corresponding positive sample s^{k_+} by:

$$\begin{aligned} s^q &= \{r_1, r_3, r_5, \ldots, r_{2\lfloor N/2 \rfloor + 1}\}, \\ s^{k_+} &= \{r_2, r_4, r_6, \ldots, r_{2\lfloor N/2 \rfloor}\}. \end{aligned} \tag{3}$$

While the negative samples s^{k_i} corresponds to the query s^q is randomly selected from the queries and the positive samples corresponds to the other trajectories s' in the batch, denoted as:

$$s^{k_i} \in \{s'^q, s'^{k_+} \text{ where } s' \in S, s' \neq s\}. \tag{4}$$

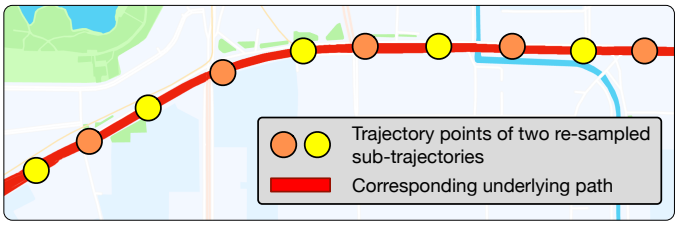


Fig. 5. Two re-sampled sub-trajectories correspond to the sample underlying path.

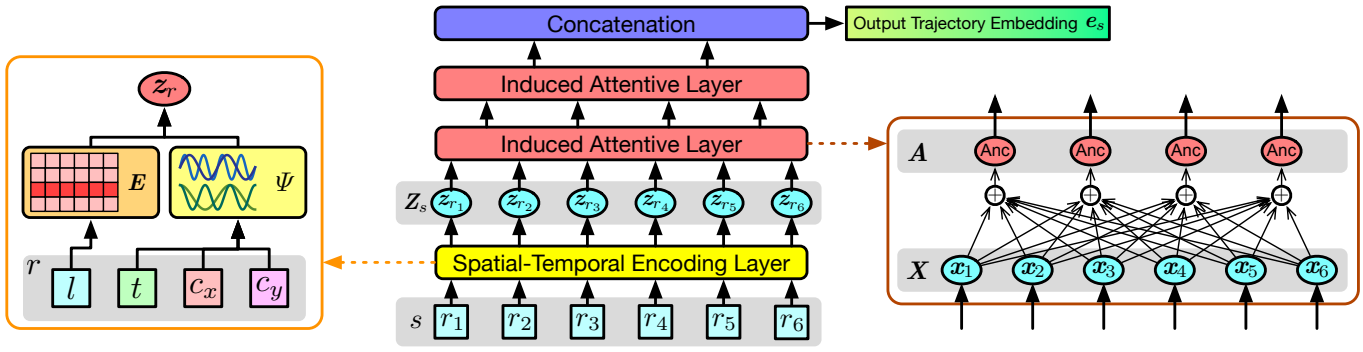
The idea behind the above augmentation method is that the sub-trajectories sampled from the same trajectory represent similar high-level travel semantics, while the ones sampled from different trajectories represent dissimilar semantics. Coupling this high-level semantics-preserved augmentation method with the contrastive pretext task introduced in Section 4.2, the learned embeddings are able to extract and incorporate the semantic information embedded in the trajectories.

4.4 Efficient Spatial-Temporal Trajectory Encoder

In pre-training trajectory embedding methods, trajectory encoders are built to model the embedded spatial-temporal correlations, and to cast the input trajectory sequences into their corresponding embeddings. Trajectories are often long sequences, and possess complex spatial-temporal correlations that can be viewed from absolute and relative aspects. These characteristics impose challenges to trajectory encoder design. To tackle these challenges, we build a trajectory encoder f that can efficiently and comprehensively model long-term spatial-temporal correlations in trajectories. As shown in Figure 6(b), our proposed trajectory encoder mainly consists of two types of layers: the spatial-temporal encoding layer, and the induced attentive layer. The following sections present its construction in detail.

4.4.1 *Spatial-Temporal Encoding Layer*

As demonstrated in Figure 3, spatial-temporal information embedded in trajectories can be viewed from two aspects: absolute and relative. Absolute spatial-temporal information, including the discrete indices and geographical posi-



(a) Spatial-Temporal Encoding Layer (b) Efficient Spatial-Temporal Trajectory Encoder (c) Induced Attentive Layer

Fig. 6. The architecture of our proposed efficient spatial-temporal trajectory encoder and its components.

tions of visited locations, plus the time periods of trajectory records, can reveal trajectories' corresponding travel purposes. Relative spatial-temporal information, including the time difference and relative position between trajectory records, indicates the movement direction and speed of a user or a vehicle, thus representing moving patterns. These two aspects of spatial-temporal information are both complicated and essential for trajectory embedding. Directly feeding raw features into the trajectory encoder is not the optimal choice, since spatial-temporal features don't scale linearly like some physical variables. We propose to more properly incorporate these raw features through a specially designed spatial-temporal encoding layer with learnable parameters, and encode trajectories into their corresponding input latent sequences. In this way, both absolute and relative aspects of spatial-temporal information in trajectories can be properly incorporated.

Encoding Absolute Spatial-temporal Information. The absolute spatial-temporal information of trajectories includes the timestamp of records, the indices and the geographical coordinates of visited locations. In other words, we aim to encode each record $r = (l, t, c_x, c_y)$ into a latent vector $\boldsymbol{z}_r \in \mathbb{R}^{d_L}$ to extract the absolute spatial-temporal information embedded in the record. d_L denotes the size of the input latent vector.

For the discrete location index l, we implement an index fetching embedding module. Suppose there is a total of N_l locations in the dataset. We initialize an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{N_l \times d_L}$, and fetch its l-th row vector $\boldsymbol{E}_l$ as the encoding vector of location index l:

$$\boldsymbol{z}_l = \boldsymbol{E}_l. \tag{5}$$

In this way, every location is represented by a learnable latent vector.

For the three continuous features t, c_x and c_y, directly regarding them as input features will lose information, since they don't scale linearly in the feature space. Akin to the location index, we aim to cast these features into learnable latent vectors, while also emphasizing their characteristics, such as periodicity. Inspired by the positional encoding introduced in the Transformer [16] and the temporal encoding proposed in some recent works [22], [35], we construct a continuous encoding module as:

$$\Psi(v) = [\cos(\omega_1 v), \sin(\omega_1 v), \ldots, \cos(\omega_{d_v/2} v), \sin(\omega_{d_v/2} v)], \tag{6}$$

where Ψ denotes the continuous encoding module, v denotes the input numerical feature, $\{\omega_1, \omega_2, \ldots, \omega_{d_v/2}\}$ is a set of learnable parameters, d_v is the dimension of the encoding vector. By using trigonometric functions, we can preserve the periodicity of space and time. By using learnable parameters, the information embedded in the continuous features can be learned by the model during training. The three continuous features' corresponding encoding vectors are calculated as:

$$\begin{aligned} \boldsymbol{z}_t &= \Psi(t) \in \mathbb{R}^{d_L}, \\ \boldsymbol{z}_{c_x} &= \Psi(c_x) \in \mathbb{R}^{d_L/2}, \\ \boldsymbol{z}_{c_x} &= \Psi(c_x) \in \mathbb{R}^{d_L/2}. \end{aligned} \tag{7}$$

Finally, we construct the spatial-temporal encoding layer by fusing all encoding vectors to gain the input latent vector of record r:

$$\boldsymbol{z}_r = \mathrm{ST}(r) = \boldsymbol{z}_l + \boldsymbol{z}_t + [\boldsymbol{z}_{c_x} || \boldsymbol{z}_{c_y}], \tag{8}$$

where ST denotes the spatial-temporal encoding layer, $||$ denotes concatenation along feature dimension. We illustrate the structure of ST in Figure 6(a).

Encoding Relative Spatial-temporal Information. To extract the relative spatial-temporal information, we want the model to formulate and parameterize the difference or the distance between the continuous features of two records. Due to one characteristic of the trigonometric functions in Ψ, there is no need to explicitly encode the difference value of two features into a latent vector. To explain, we give the dot-product of two encoding vectors $\Psi(v)$ and $\Psi(v+\delta)$ as:

$$\Psi(v) \cdot \Psi(v+\delta) = \cos(\omega_1 \delta) + \cos(\omega_2 \delta) + \cdots + \cos(\omega_{d_v/2} \delta), \tag{9}$$

which means the distance (measured by dot-product) between $\Psi(v)$ and $\Psi(v+\delta)$ is only dependent on the set of learnable parameters and the difference δ, not on the offset v. In other words, the relative information is only related to the difference between two features, and can be learned by the model during training. Since we use dot-product attention in our trajectory encoder, as denoted later in Equation 11, the dot-product between the encoding vectors of two records' timestamps or coordinates can be easily

incorporated by the encoder. Thus, information embedded in the relative differences between spatial-temporal features can be captured and learned.

Forming Input Latent Sequence. Finally, the input latent sequence $\boldsymbol{Z}_s$ of a trajectory s is formed by the input latent vectors of every record in the trajectory. Formally,

$$\begin{aligned} \text{Given } s &= \{r_1, r_2, \ldots, r_N\}, \\ \boldsymbol{Z}_s &= \mathrm{ST}(s) \\ &= \mathrm{ST}(\{r_1, r_2, \ldots, r_N\}) \\ &= \{\boldsymbol{z}_{r_1}, \boldsymbol{z}_{r_2}, \ldots, \boldsymbol{z}_{r_N}\}. \end{aligned} \tag{10}$$

4.4.2 Induced Attentive Layer

Trajectories contain long-term correlations, which means that all visiting records in a trajectory can be useful for revealing the trajectory's high-level semantic information, regardless of their position. For example, starting a trajectory from hotels or ending one at scenic spots can both indicate that the corresponding mobile user is traveling. The Recurrent Neural Network (RNN) [8] used in existing trajectory embedding methods suffers from the vanishing gradient issue [15], thus not powerful enough for modeling long-term correlations. On the other hand, different visiting records have different importance with regard to the trajectory. For instance, in a taxi trajectory, the staying points contain more semantic information than the passing-by points. To solve the vanishing gradient issue of RNNs and to dynamically model the importance of different visiting records, a straightforward idea is to construct the trajectory encoder based on self-attention [16] and pooling layers. Yet, self-attention will lead to square time and memory complexity, which hinders the model's scalability on long trajectories. Plus, max and mean pooling layers are unable to dynamically calculate an importance score for each item in a sequence.

Inspired by recent advances in sequential models in neural language processing [36], we utilize the attention mechanism for better incorporation of long-term correlations, and use globally-shared anchor sequences for attention weight calculation to reduce computational complexity. We call this layer the induced attentive layer for its calculation process can be viewed as inducing the input sequence into the anchor sequence through the attention mechanism. Formally, we implement each induced attentive layer based on attention and feed-forward networks:

$$\begin{aligned} &\mathrm{IA}(\boldsymbol{X}, \boldsymbol{A}) = \mathrm{Norm}(\boldsymbol{H} + \mathrm{FFN}(\boldsymbol{H})), \\ &\text{where } \boldsymbol{H} = \mathrm{Norm}(\boldsymbol{A} + \mathrm{Att}(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{X})), \end{aligned} \tag{11}$$

where IA denotes one induced attentive layer. Norm, FFN, Att represents the layer normalization [37], the feed-forward network and the multi-head dot-product attention, respectively. $\boldsymbol{X} \in \mathbb{R}^{N_X \times d_L}$ is the input sequence for this layer, $\boldsymbol{A} \in \mathbb{R}^{N_A \times d_L}$ represents the anchor sequence with length N_A, and can be viewed as a set of learnable parameters of this layer. During attention calculation, $\boldsymbol{A}$ is regarded as the query, $\boldsymbol{X}$ is regarded as the key and value. In conclusion, Equation 11 can be viewed as inducing the input sequence $\boldsymbol{X}$ into the anchor sequence $\boldsymbol{A}$. The structure of IA can be seen in Figure 6(c).

Because the attention scores of all records in one input sequence are synchronously calculated, the induced attentive layer can avoid the vanishing gradient issue [15] found in RNNs, thus better at modeling the long-term correlations of trajectories. On the other hand, sharing the anchor sequences across the whole dataset not only can significantly reduce the parameter numbers, but also is useful to learn the most prominent patterns among all the trajectories.

4.4.3 Stacking Multiple Layers

Finally, our trajectory encoder is constructed by stacking multiple induced attentive layers on top of one spatial-temporal encoding layer, as shown in Figure 6(b). We give an example of a trajectory encoder with two induced attentive layers as:

$$\begin{aligned} f(s) &= \mathrm{cat}(\mathrm{IA}_2(\mathrm{IA}_1(\mathrm{ST}(s), \boldsymbol{A}_1), \boldsymbol{A}_2)), \\ &= \mathrm{cat}(\mathrm{IA}_2(\mathrm{IA}_1(\boldsymbol{Z}_s, \boldsymbol{A}_1), \boldsymbol{A}_2)), \end{aligned} \tag{12}$$

where IA_1 and IA_2 denotes the first and the second induced attentive layers, $\boldsymbol{A}_1 \in \mathbb{R}^{N_{A_1} \times d_L}$, $\boldsymbol{A}_2 \in \mathbb{R}^{N_{A_2} \times d_L}$ are the anchor sequences of two induced attentive layers, f denotes our proposed efficient spatial-temporal trajectory encoder introduced in Equation 1, $\boldsymbol{Z}_s$ is the input latent sequence of trajectory s. cat denotes concatenation along the feature dimension. In other words, we concatenate the output from IA_2 to get the result trajectory embedding vector. It is clear that the size of the result embedding vector is related to the length of the anchor sequence in IA_2. We denote the size as d_O, and $d_O = N_{A_2} * d_L$.

By compressing the input trajectories into corresponding embeddings utilizing the attention mechanism, the importance of each visiting record in a trajectory is efficiently and adaptively estimated. Meanwhile, the spatial-temporal encoding layer with learnable parameters enables our encoder to comprehensively model the spatial-temporal information embedded in trajectories.

5 Experiments

In order to evaluate the quality of trajectory embedding vectors generated by our model, we incorporate these vectors into two downstream trajectory mining tasks, and compare the results with other methods on three real-world datasets.

5.1 Datasets

We conduct our experiments on three real-world datasets. Two of which contain taxi trajectories collected from the center areas of Chengdu and Xian by Didi, denoted as Taxi-CD and Taxi-XA. These trajectories are reported by GPS equipments on taxes during journeys. The other one contains mobile signaling data collected from Shenyang, denoted as Mobile-SY. It records mobile phone users' switching events between telecommunication towers.

For taxi trajectories, we re-sample the original data to modify the sample rate to roughly 1 minute, and filter out trajectories that contain less than 20 records. We then choose the trajectories that started from November 1st to 7th, 2018. Finally, we use a squared grid with 250 meter-long edges to assign each coordinate point in the trajectories with a discrete location index. For mobile signaling data, we

regard telecommunication towers as locations, and a mobile user's switch records within one day as one trajectory. We then remove records with staying time below 1 minute, and trajectories with less than 20 records. The statistics of datasets after preprocessing are shown in Table 1.

TABLE 1
Statistics of datasets.

Dataset	#Trajectories	#Locations	#Records	Time span
Taxi-CD	44,551	1,443	1,555,042	7 days
Taxi-XA	70,222	1,469	2,597,066	7 days
Mobile-SY	64,328	7,456	1,338,838	11 days

5.2 Comparison Methods

To prove the superiority of our proposed model, we include two end-to-end embedding methods that are widely adopted by trajectory mining models, and some state-of-the-art pre-training trajectory embedding methods as baselines.

- **Mean**: a permutation equivariant embedding method, which applies a mean pooling operation on a trajectory's sequence of input features, followed by a linear transform layer to generate its latent embedding vector.
- **RNN** [8]: aggregates a trajectory's sequence of input features using a recurrent neural network, and regards the output hidden state as the latent embedding vector of the trajectory.
- **traj2vec** [4]: constructs feature sequences by calculating the difference in spatial-temporal attributes between consecutive points, and applies an RNN encoder to aggregate them.
- **t2vec** [7]: pre-trains an RNN trajectory encoder based on de-noise auto-encoding, aims to recover the underlying route of trajectories.
- **GM-VSAE** [5]: casts trajectories into a multi-dimension Gaussian space by coupling an RNN trajectory encoder with variational fully-connections.
- **TremBR** [9]: incorporates temporal information by concatenating the travel time with location embedding vectors to form the input features.

5.3 Downstream Trajectory Mining Tasks

We evaluate the performance of trajectory embedding methods through two downstream mining tasks: similar trajectory search and destination prediction.

5.3.1 Similar Trajectory Search

We aim to rank similar trajectories by utilizing their distances in the latent embedding space. Due to the lack of ground truth, we implement the strategy used in TremBR [9]. Specifically, for each trajectory s in the dataset S, we take odd-numbered and even-numbered points to create two sub-trajectories $s^{(a)}$ and $s^{(b)}$, thus forming two trajectory sets $S^{(a)}$ and $S^{(b)}$. While performing the task, given one odd-sampled trajectory $s_i^{(a)} \in S^{(a)}$, we rank even-sampled trajectories $s_j^{(b)} \in S^{(b)}$ in terms of their embedding vectors' dot-product value with $s_i^{(a)}$. Technically, the corresponding trajectory $s_i^{(b)}$ should have the highest dot-product value, and be ranked at the top. For this task, we include a classic sequence similarity measurement method **DTW** (Dynamic Time Warping) [38] as an additional baseline.

5.3.2 Destination Prediction

We intend to predict the destination for each trajectory given its embedding vector. The embedding methods are fed with the target trajectory excluding its last record to generate an embedding vector, which is then brought into a fully connected network to predict the location index of the last record. For this task, we include a classic sequential correlation modeling method **MC** (Markov Chain) [39] as an additional baseline.

5.4 Settings

For all datasets, we sort trajectories by their starting time, and split the whole trajectory set into the training, evaluation and testing sets by 8:1:1. Both embedding models and downstream models are trained with the training set, early-stopped on the evaluation set, and calculated final metrics on the testing set. We choose Top-N accuracy (i.e., Acc@N, $N \in [1, 5, 10, 20]$) and macro-F1 as the evaluation metrics for both tasks.

We implement all models using PyTorch [40]. [1] The size of input latent vector d_L is set to 64, the target dimension of the result trajectory embedding vectors d_O and the hidden size of feed-forward networks are set to 128. For our proposed model, we construct the encoder using 1 encoder layer, and set the length of anchor sequence N_{A_1} to 2, the number of attention heads N_H to 8. The number of negative samples n_{neg} is set to 2, and the temperature τ is set to 0.07. During optimization, we choose the Adam optimizer and an initial learning rate of 0.001 across the board. All experiments were run on a GPU cluster with 4 workers, where each worker equipped with one Intel Xeon Silver 4210 CPU, and 4 NVIDIA RTX2080Ti GPUs.

5.5 Experimental Results

5.5.1 Overall Performance

Table 2 and Table 3 demonstrates the performance comparison of different approaches for similar trajectory search and destination prediction, respectively. Our proposed method consistently shows its performance superiority over other trajectory embedding methods except on the similar trajectory search task, Mobile-SY dataset.

The two classic machine learning methods, DTW and MC specifically solve problems through feature engineering and kernel designs. Compared to embedding methods that are widely used in deep learning, they are struggling to model complex spatial-temporal features in trajectories. Yet, we can see that they can still beat the permutation equivalent embedding method Mean, for they take sequential correlation into consideration.

The two end-to-end embedding methods, Mean and RNN only trained on specific prediction objectives, which

1. The code and datasets are attached in the supplementary material, and will be published on GitHub after the review.

TABLE 2
Downstream prediction performance comparison of different approaches on similar trajectory search.

Dataset \ Metric	Embedding	Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)
Taxi-CD	DTW	11.947	30.089	43.805	56.637	8.563
	Mean	10.009±0.19	33.281±1.27	43.559±1.73	57.361±1.76	8.700±0.37
	RNN	19.962±0.24	46.589±0.13	58.977±0.51	70.994±1.06	15.416±0.23
	traj2vec	26.638±1.52	56.003±0.59	69.199±0.02	82.866±0.43	17.855±1.08
	t2vec	30.969±0.63	60.806±0.46	74.158±1.67	85.435±2.03	24.000±0.68
	GM-VSAE	28.097±1.98	60.368±3.41	74.237±2.60	86.378±1.16	21.494±2.13
	TremBR	34.706±0.43	65.934±1.49	78.714±1.16	88.936±0.92	25.818±0.32
	CSTTE	**40.911±1.87**	**72.969±0.43**	**83.802±0.66**	**92.157±0.36**	**33.723±1.93**
Taxi-XA	DTW	24.684	50.633	63.291	77.215	19.421
	Mean	13.570±0.66	33.106±1.61	41.250±2.84	52.442±3.16	10.451±0.44
	RNN	35.256±0.52	64.161±2.64	76.164±2.38	86.644±2.42	26.595±0.76
	traj2vec	42.589±1.91	73.323±2.74	82.629±2.36	89.278±1.96	35.893±1.11
	t2vec	51.004±1.73	81.966±0.69	90.196±0.63	94.511±1.20	43.699±2.05
	GM-VSAE	48.384±1.95	76.022±2.47	84.451±3.78	90.545±2.37	41.665±1.40
	TremBR	54.948±0.52	85.348±2.64	92.852±2.38	96.839±2.42	47.660±0.26
	CSTTE	**63.391±2.17**	**90.645±0.90**	**95.093±0.56**	**97.314±0.86**	**56.195±2.66**
Mobile-SY	DTW	5.185	18.221	23.644	31.887	3.799
	Mean	4.055±0.17	14.625±2.39	23.755±1.84	33.102±2.36	2.291±0.15
	RNN	5.106±0.33	15.068±1.32	23.988±1.77	33.546±2.29	2.848±0.66
	traj2vec	7.145±0.40	20.416±1.60	29.631±2.94	40.068±4.47	4.437±0.32
	t2vec	9.916±1.45	24.175±2.67	32.628±2.45	42.762±1.46	6.497±1.22
	GM-VSAE	11.800±0.36	27.335±0.31	36.379±0.77	46.404±0.43	8.258±0.95
	TremBR	12.842±0.48	30.542±2.22	**41.477±1.05**	**52.249±1.82**	9.027±0.18
	CSTTE	**13.434±1.32**	**30.612±1.75**	40.528±2.50	50.615±2.63	**9.396±1.23**

TABLE 3
Downstream prediction performance comparison of different approaches on destination prediction.

Dataset \ Metric	Embedding	Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)
Taxi-CD	MC	11.491	26.509	37.815	43.663	0.876
	Mean	5.364±0.25	17.527±0.96	26.728±0.83	37.377±0.81	0.581±0.08
	RNN	15.215±0.57	32.148±0.86	41.427±0.59	51.346±0.78	0.888±0.06
	traj2vec	10.660±0.40	23.676±0.75	31.486±0.56	40.126±0.54	0.649±0.04
	t2vec	17.549±0.92	34.089±1.30	42.908±1.41	52.962±2.17	1.273±0.23
	GM-VSAE	16.786±0.71	33.707±1.03	41.809±1.05	51.167±1.22	0.973±0.06
	TremBR	20.007±0.40	37.264±1.13	45.433±1.16	54.275±1.51	1.407±0.07
	CSTTE	**22.913±1.24**	**41.394±0.90**	**50.707±0.36**	**60.536±0.75**	**1.714±0.15**
Taxi-XA	MC	13.031	31.287	36.178	49.715	0.879
	Mean	6.386±0.39	19.806±0.60	29.075±0.57	38.157±1.25	0.651±0.07
	RNN	16.203±0.50	33.348±0.94	40.977±0.88	49.964±0.80	0.889±0.12
	traj2vec	11.754±0.78	26.242±1.22	34.700±1.32	43.678±1.60	0.820±0.08
	t2vec	18.467±0.50	38.744±1.34	43.728±1.94	53.040±2.17	1.117±0.13
	GM-VSAE	18.425±0.92	35.241±1.27	41.172±1.83	49.722±1.94	0.957±0.06
	TremBR	20.953±0.21	39.307±0.53	46.853±1.62	55.204±1.87	1.326±0.20
	CSTTE	**24.050±0.38**	**43.037±0.31**	**51.260±0.16**	**59.853±0.11**	**1.430±0.10**
Mobile-SY	MC	3.0044	14.489	22.351	35.181	1.246
	Mean	1.276±0.09	5.666±0.15	10.048±0.17	16.462±0.36	1.127±0.06
	RNN	4.125±0.22	15.894±0.08	24.782±0.94	35.788±0.32	1.498±0.10
	traj2vec	3.923±0.13	14.306±0.15	22.408±0.65	33.562±0.57	1.571±0.08
	t2vec	4.732±0.28	15.707±0.96	24.408±1.10	34.760±1.49	1.792±0.29
	GM-VSAE	5.168±0.20	17.341±1.02	26.728±1.54	39.072±1.86	2.103±0.24
	TremBR	5.164±0.41	19.100±1.29	28.502±1.02	41.687±0.75	1.972±0.36
	CSTTE	**6.157±0.21**	**21.069±0.19**	**31.522±0.31**	**43.501±0.39**	**2.344±0.14**

makes it hard to share embedding vectors between different tasks, and can lead to over-fitting on small datasets. By comparison, the two RNN-based pre-training methods t2vec and GM-VSAE show performance superiority, which proves the effectiveness of the self-supervised pre-training strategy.

Incorporating spatial-temporal information is essential for generating high-quality trajectory embeddings. t2vec and GM-VSAE only implement index-fetching embedding layers to represent location indices, thus failing to incorporate temporal information. TremBR takes relative temporal information into consideration on top of location indices. traj2vec constructs relative features by calculating the difference in spatial-temporal attributes between consecutive visiting records in trajectories. Yet, none of the aforementioned trajectory embedding methods explicitly incorporate both relative and absolute aspects of spatial-temporal informa-

tion, which will limit their comprehensiveness. By contrast, the spatial-temporal encoding layer in our proposed CSTTE method can extract relative and absolute aspects of spatial-temporal information from trajectories, and fuses them into the trajectory encoder.

In addition, we propose an efficient spatial-temporal trajectory encoder to dynamically summarize trajectories into their embedding vectors, and train the encoder under the contrastive learning framework. Compared to the RNN encoder and generative pretext task used in the baseline trajectory embedding methods, our method is more capable of digging high-level travel semantics and long-term correlations from trajectories. As we can see from Table 2 and Table 3, these designs result in higher quality trajectory embedding vectors, thus helping downstream prediction models to achieve better performance.

5.5.2 Influence of Contrastive Pretext Task and Spatial-Temporal Encoding

To investigate the effectiveness of the contrastive pretext task, and the modules of our proposed spatial-temporal encoding layer in the trajectory encoder, we design five variants of the model:

1) Generative: uses a mirrored trajectory decoder to train the encoder in a generative auto-encoding manner.
2) -Discrete: removes the index fetching embedding module $\boldsymbol{E}$. In other words, removes the encoding of discrete location indices, only uses the encoding of continuous features $\boldsymbol{z}_t$, $\boldsymbol{z}_{c_x}$ and $\boldsymbol{z}_{c_y}$.
3) -Continuous: removes the continuous encoding module Ψ, only uses the encoding of discrete location index $\boldsymbol{z}_l$, and the original positional encoding from the vanilla Transformer [16].
4) -Time: removes the encoding vectors for the visited time $\Psi(t)$ in the continuous encoding module.
5) -Coordinate: removes the encoding vectors for the coordinate $\Psi(c_x), \Psi(c_y)$ in the continuous encoding module.

We compare these variants with the full model on all three datasets, destination prediction task. As Figure 7 illustrates, the CSTTE model will receive a performance penalty if the encoder is trained with the generative pretext task rather than the contrastive one. This proves the effectiveness of the contrastive learning framework. As demonstrated in Figure 1, generative pretext task can lead the learned embeddings to be sensitive to noise, while contrastive one focuses on extracting high-level semantics and is more robust.

The spatial-temporal encoding layer in the trajectory encoder incorporates both absolute and relative aspects of spatial and temporal information into the model, by encoding the discrete location indices and three continuous features of trajectory records into their latent vectors. We can see from Figure 7 that both discrete location indices and continuous features are essential for comprehensively describing the spatial-temporal properties of trajectories. The spatial-temporal encoding layer in our final model combines these features to gain the optimum results.

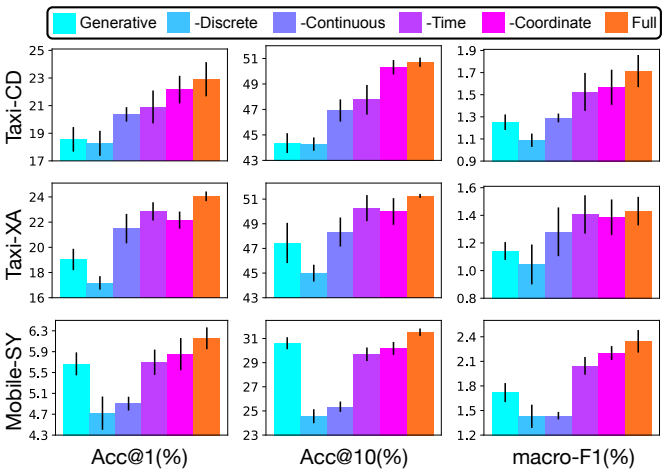


Fig. 7. Comparison of the generative pretext task variant and four spatial-temporal encoding variants with the full model on three datasets, destination prediction task.

5.5.3 Influence of Trajectory Data Augmentation Methods

We further investigate the effectiveness of the trajectory data augmentation methods used in generating samples for the contrastive pretext task. We implement four sequence sampling methods that are frequently used [13]:

1) Random: for each sample, select a portion of the trajectory by randomly deciding whether to choose each trajectory record.
2) Adjacent: select two sub-trajectories that are adjacent, yet not overlapping with each other.
3) Overlap: select two sub-trajectories that are overlapping with each other, yet neither is totally subsumed by the other.
4) Subsume: select two sub-trajectories where the long one subsumes the short one.

We compare these methods with the augmentation method we used in CSTTE (denoted as 2-Hop) on all three datasets, destination prediction task. As we can see in Figure 8, the 2-Hop method constantly gains the best result compared with other augmentation methods. This shows that choosing an optimum data augmentation method is important under the contrastive learning framework, and our proposed augmentation method is effective on trajectories.

5.5.4 Choosing the Optimum Input and Output Embedding Size

The input embedding size or the size of input latent vector d_L, and the output embedding size or the size of the result trajectory embedding vector d_O, will influence the model capacity of the spatial-temporal encoding layers and the downstream prediction models. In our CSTTE model, they are also related to the length of the anchor sequence in the last induced attentive layer N_A, since $N_A = d_O/d_L$. We proceed a set of hyper-parameter experiments on all three datasets, destination prediction task to find the optimum choice for d_L and d_O. While testing on d_L, we fix d_O to 128; while testing on d_O, we fix d_L to 64. Figure 9 and Figure 10 illustrate the results.

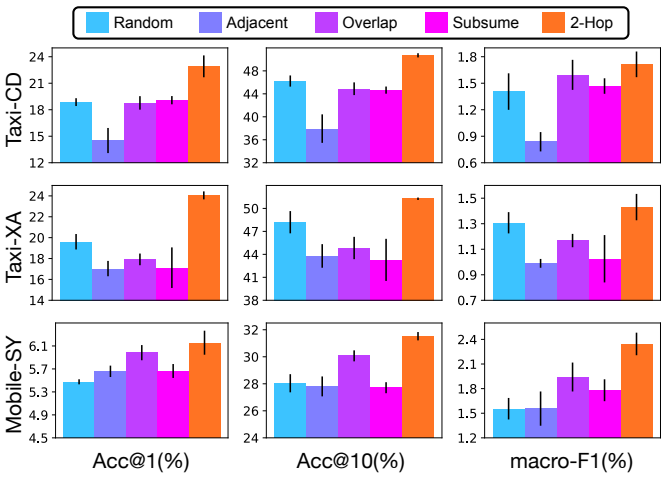


Fig. 8. Comparison of the trajectory data augmentation methods on three datasets, destination prediction task.

As we can see in the results, in most scenarios, the optimum input embedding size d_L will be 64. As for the output size d_O, setting it to larger than 128 can gain some improvements, yet the improvements on the prediction metrics are limited, and it comes with higher computational complexity. Thus, we set d_O to 128 for our final model.

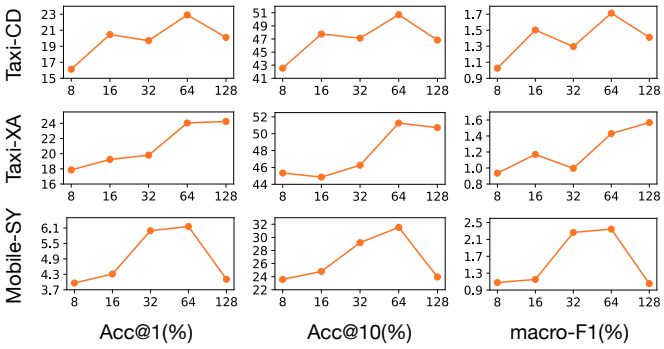


Fig. 9. Prediction performance w.r.t. the input embedding size on three datasets, destination prediction task.

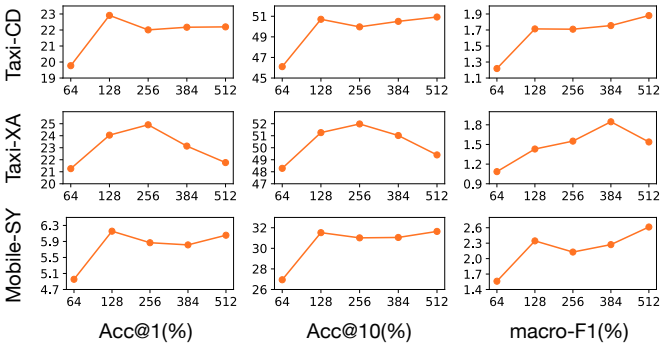


Fig. 10. Prediction performance w.r.t. the output embedding size on three datasets, destination prediction task.

5.5.5 *Choosing the Optimum Number of Encoder Layers*

We also test the optimum number of encoder layers for the trajectory encoder on all three datasets, destination prediction task. For the encoder with one layer, the length of anchor sequence N_{A_1} is set to 2; for the encoder with two layers, $N_{A_1}, N_{A_2} = 8, 2$; for the encoder with three layers, $N_{A_1}, N_{A_2}, N_{A_3} = 16, 8, 2$. Figure 11 demonstrates the results, and it is clear that on these datasets, using only one encoder layer is the best.

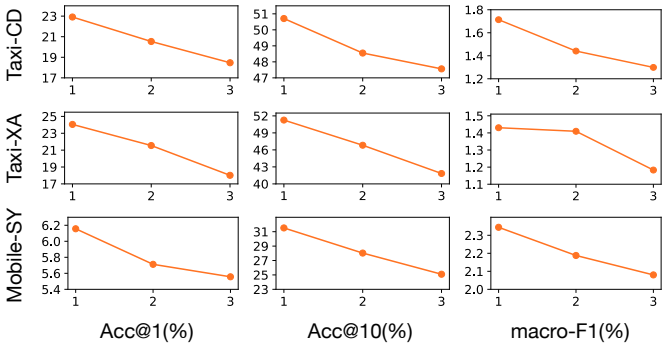


Fig. 11. Prediction performance w.r.t. the number of trajectory encoder layers on three datasets, destination prediction task.

6 Conclusion

In this paper, we propose a novel spatial-temporal aware trajectory embedding method CSTTE. It can effectively extract high-level travel semantics from trajectories, while also incorporating long-term spatial-temporal information into the embeddings both comprehensively and efficiently. The experimental results prove the effectiveness and superiority of our method for pre-training high-quality trajectory embeddings.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities (Grant No. 2021YJS030).

References

[1] D. Kong and F. Wu, "Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction." in *27th International Joint Conference on Artificial Intelligence*, vol. 18, 2018, pp. 2341–2347.

[2] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *28th World Wide Web Conference*, 2019, pp. 1017–1027.

[3] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye, "Heteta: heterogeneous information network embedding for estimating time of arrival," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2444–2454.

[4] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *16th International Joint Conference on Neural Networks*. IEEE, 2017, pp. 3880–3887.

[5] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," in *36th International Conference on Data Engineering*, 2020, pp. 949–960.

[6] P. Zhao, H. Zhu, Y. Liu, J. Xu, Z. Li, F. Zhuang, V. S. Sheng, and X. Zhou, "Where to go next: A spatio-temporal gated network for next poi recommendation," in *33rd AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5877–5884.

[7] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *34th International Conference on Data Engineering*, 2018, pp. 617–628.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[9] T.-Y. Fu and W.-C. Lee, "Trembr: Exploring road networks for trajectory representation learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 1, pp. 1–25, 2020.
[10] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," *Advances in neural information processing systems*, vol. 28, 2015.
[11] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
[12] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," *Advances in neural information processing systems*, vol. 32, 2019.
[13] J. Giorgi, O. Nitski, B. Wang, and G. Bader, "Declutr: Deep contrastive learning for unsupervised textual representations," in *59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, pp. 879–895.
[14] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *16th European Conference on Computer Vision*, 2020, pp. 776–794.
[15] T. Linzen, E. Dupoux, and Y. Goldberg, "Assessing the ability of lstms to learn syntax-sensitive dependencies," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 521–535, 2016.
[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *30th Advances in Neural Information Processing systems*, 2017, pp. 5998–6008.
[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
[18] Y. Zhou and Y. Huang, "Deepmove: Learning place representations through large scale movement data," in *6th IEEE International Conference on Big Data*, 2018, pp. 2403–2412.
[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representation*, 2013.
[20] S. Feng, G. Cong, B. An, and Y. M. Chee, "POI2Vec: Geographical latent representation for predicting future visitors." in *31th AAAI Conference on Artificial Intelligence*, 2017, pp. 102–108.
[21] H. Wan, Y. Lin, S. Guo, and Y. Lin, "Pre-training time-aware location embeddings from spatial-temporal trajectories," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
[22] Y. Lin, H. Wan, S. Guo, and Y. Lin, "Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction," in *35th AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 4241–4248.
[23] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new POI recommendation," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 2069–2075.
[24] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "ATRank: An attention-based user behavior modeling framework for recommendation," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 4564–4571.
[25] Z. Yao, Y. Fu, B. Liu, W. Hu, and H. Xiong, "Representing urban functions through zone embedding with human mobility patterns." in *27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3919–3925.
[26] T. Shimizu, T. Yabe, and K. Tsubouchi, "Learning fine grained place embeddings with spatial hierarchy from human mobility trajectories," *arXiv preprint arXiv:2002.02058*, 2020.
[27] M.-x. Wang, W.-C. Lee, T.-y. Fu, and G. Yu, "Learning embeddings of intersections on road networks," in *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 309–318.
[28] Y. Chen, X. Li, G. Cong, Z. Bao, C. Long, Y. Liu, A. K. Chandran, and R. Ellison, "Robust road network representation learning: When traffic patterns meet traveling semantics," in *30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 211–220.
[29] N. Wu, X. W. Zhao, J. Wang, and D. Pan, "Learning effective road network representation with hierarchical graph neural networks," in *26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 6–14.
[30] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *31st international conference on data engineering*. IEEE, 2015, pp. 543–554.
[31] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering a* search algorithms with neural networks for personalized route recommendation," in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 539–547.
[32] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1695–1704.
[33] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.
[34] A. Pauls and D. Klein, "Faster and smaller n-gram language models," in *49th annual meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 258–267.
[35] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *8th International Conference on Learning Representations*, 2020.
[36] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *36th International Conference on Machine Learning*, 2019, pp. 3744–3753.
[37] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[38] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.
[39] S. Brooks, "Markov chain monte carlo method and its application," *Journal of the royal statistical society: series D (the Statistician)*, vol. 47, no. 1, pp. 69–100, 1998.
[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *32nd Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

Yan Lin received the B.S. degree in computer science from Beijing Jiaotong University, Beijing, China, in 2019.

He is currently working toward the Ph.D. degree in the School of Computer and Information Technology, Beijing Jiaotong University. His research interests include spatial-temporal data mining and graph neural networks.

Huaiyu Wan received the Ph.D. degree in computer science and technology from Beijing Jiaotong University, Beijing, China, in 2012.

He is an Associate Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His current research interests focus on spatial-temporal data mining, social network mining, information extraction, and knowledge graph.

Shengnan Guo received the B.S. degree in computer science from Beijing Jiaotong University, Beijing, China, in 2015.

She is currently working toward the Ph.D. degree in the School of Computer and Information Technology, Beijing Jiaotong University. Her research interests focus on the area of deep learning and spatial-temporal data mining.

Youfang Lin received the Ph.D. degree in signal and information processing from Beijing Jiaotong University, Beijing, China, in 2003.

He is a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His main fields of expertise and current research interests include big data technology, intelligent systems, complex networks, and traffic data mining.