PreCLN: Pretrained-based Contrastive Learning Network for Vehicle Trajectory Prediction

Bingqi Yan
Ocean University of China
Geng Zhao
Worcester Polytechnic Institute
Lexue Song
Duke Kunshan University
Yanwei Yu (yuyanwei@ouc.edu.cn)
Ocean University of China
Junyu Dong
Ocean University of China

Research Article

Keywords: Trajectory Prediction, Trajectory Representation, Contrastive Learning, Pre-training, GPS Trajectory

Posted Date: October 6th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2121114/v1

PreCLN: Pretrained-based Contrastive Learning Network for Vehicle Trajectory Prediction

Bingqi Yan[1], Geng Zhao[2], Lexue Song[3], Yanwei Yu[1*] and Junyu Dong[1]

[1*]College of Computer Science and Technology, Ocean University of China, 238 Songling RD, Qingdao, 266100, Shandong, China.
[2]Department of Computer Science, Worcester Polytechnic Institute, 100 Institute RD, Worcester, 01609, MA, USA.
[3]Department of Data Science, Duke Kunshan University, 8 Duke Avenue, Kunshan, 215316, Jiangsu, China.

*Corresponding author(s). E-mail(s): yuyanwei@ouc.edu.cn;
Contributing authors: ybq@stu.ouc.edu.cn; gzhao4@wpi.edu;
ls440@duke.edu; dongjunyu@ouc.edu.cn;

Abstract

Trajectory prediction of vehicles is of great importance to various smart city applications ranging from transportation scheduling, vehicle navigation, to location-based advertisements. Existing methods all focus on modeling spatiotemporal relations with explicit contextual semantics from labeled trajectory data, and rarely consider the effective use of large amounts of available unlabeled trajectory data with the assistance of contrastive learning and pre-training techniques. To this end, we develop a novel Pretrained-based Contrastive Learning Network (PreCLN) for vehicle trajectory prediction. Specifically, we propose a dual-view trajectory contrastive learning framework to achieve self-supervised pre-training. A Transformer-based trajectory encoder is designed to effectively capture the long-term spatiotemporal dependencies in trajectories to embed input trajectories into fixed-length representation vectors. Moreover, three auxiliary pre-training tasks, *i.e.*, trajectory imputation, trajectory destination prediction, and trajectory-user linking, are used to assist the training of PreCLN with the dual-view trajectory contrastive learning framework.

After pre-training, the result trajectory encoder is used to generate trajectory representations for future trajectory prediction. Extensive experiments on two real-world large-scale trajectory datasets demonstrate the significant superiority of PreCLN against state-of-the-art trajectory prediction baselines in terms of all evaluation metrics.

Keywords: Trajectory Prediction, Trajectory Representation, Contrastive Learning, Pre-training, GPS Trajectory

1 Introduction

With the rapid growth of location-based services and the wide spread of GPS-equipped devices, the availability of large-scale trajectory data has been increasing in recent years, including taxi trajectories, human check-ins, traffic camera surveillance data, and more. Therefore, mining spatiotemporal trajectory data has been extensively studied, such as predicting future trajectories [1, 2], estimating arrival times [3, 4], traffic prediction [5–7], and route planning [8–12]. Among these researches, trajectory prediction has attracted a great amount of attention, which is of crucial importance for varieties of smart city applications, such as transportation scheduling, location-based recommendations, and autonomous driving. For example, trajectory prediction can assist traffic managers in predicting possible impending congestion and travel delays, and executing corresponding traffic control and scheduling strategies. If the driving path of the vehicle can be accurately predicted, the intelligent transportation system can provide the driver with personalized assistance or recommendations, such as dynamic re-routing, personalized risk assessment and prompts, speed suggestions, etc. Additionally, based on trajectory prediction, customized location-based advertisements can also be developed for those most likely to pass through certain commercial stores. Nonetheless, due to the complexity of the road network and the dynamics of urban traffic, as well as the uncertainty of the driving conditions, effective vehicle trajectory prediction faces many challenges.

Early methods towards this goal, have made significant efforts on trajectory prediction and next location prediction based on Markov model [13–15] or Recurrent Neural Network (RNN) model [16–19]. Based on the great success of RNN model in sequence modeling, a handful of RNN model-based location prediction or recommendation studies have been developed. LSTPM [17] is proposed to use a geo-dilated RNN to exploit the geographical relations among non-consecutive locations to conquer the limitation of RNNs in short-term user preference modeling. Li *et al.* [20] design a multi-layer LSTM encoder-decoder model that leverages temporal attention mechanism to enhance model sequence learning ability. STKG [19] is developed to incorporate knowledge graph embedding and Graph Convolution Network (GCN) into RNN-based models for better capturing the sequential transition patterns. In recent years,

self-attention models have been extensively applied to temporal location modeling [21, 22]. The recent state-of-the-art works [22–24] feed time interval, user, and location into self-attention based models to explicitly learn spatiotemporal dependencies in trajectories, and stack extra modules [17, 24, 25] to integrate contextual information.

Despite the effectiveness of existing trajectory prediction methods [2, 14, 19, 22–24, 26, 27], these works have three key limitations. *First*, RNN-based models can hardly capture the long-term dependencies among the long trajectory sequences as RNNs have a certain degree of forgetfulness in processing long-sequence data [28]. *Second*, most works do not adequately exploit the potential of pre-training techniques for trajectory prediction on the available large-scale trajectory data. Recent spatiotemporal data mining works [24, 26] prove that pre-trained models effectively improve model performance. *Third*, most works do not consider strengthening the trajectory prediction ability with the assistance of a large number of auxiliary tasks, for example, enhancing the model's ability to capture long-term dependencies through destination prediction, and enhancing the model's ability to learn user preferences through trajectory-user linking.

To tackle the aforementioned challenges, we develop a novel Pretrained-based Contrastive Learning Network for vehicle trajectory prediction, named PreCLN. To be more specific, we design a dual-view trajectory contrastive learning framework, which is to achieve self-supervised pre-training. Two trajectory representations of an input trajectory learned from two different views (*i.e.*, hierarchical map gridding and road network mapping), respectively, are regarded as the query and the positive sample; Trajectory representations learned from other trajectories in the same batch are treated as negative samples with respect to the query. To generate more trajectory samples from the original trajectory data, we employ three different trajectory data augmentation strategies for pre-training. To effectively capture the long-term spatiotemporal correlations in trajectories, we develop a Transformer-based trajectory encoder to embed the input trajectory sequences into fixed-length representations. Moreover, we leverage three auxiliary pre-training tasks, *i.e.*, trajectory imputation, trajectory destination prediction, and trajectory-user linking, to assist the training of the trajectory encoder. After the pre-training, the result trajectory encoder is used to generate trajectory representations for vehicle trajectory prediction. We conduct extensive experiments on two real-world large-scale trajectory datasets, and experimental results demonstrate that our PreCLN model can obtain significant performance improvement compared with state-of-the-art trajectory prediction techniques.

We summarize the contributions of this paper as follows:

- We propose PreCLN model, a novel pre-trained spatiotemporal contrastive learning network, to make fully effective use of context-aware geographical information and road network for vehicle trajectory prediction.

- We develop a dual-view trajectory contrastive learning framework that effectively captures long-term spatiotemporal dependencies in trajectories through a Transformer-based trajectory encoder with three auxiliary pre-training tasks.
- Extensive experiments on two real-world large-scale GPS trajectory datasets verify the superiority of our proposed model in vehicle trajectory prediction when competing with state-of-the-art baselines.

2 Related Work

In this section, we mainly discuss related works, including next location prediction and trajectory prediction.

2.1 Next Location Prediction

RNN models are widely used in modeling sequential forecasting in most existing next location prediction works [19, 29–31]. Liu *et al.* [29] propose STRNN model that inputs temporal and spatial intervals between every two consecutive visits as additional auxiliary information into RNN model to improve prediction performance. Yao *et al.* [30] use a recurrent model to jointly learn user temporal preference and semantic contexts for next location prediction. Feng *et al.* [32] develop DeepMove model, which learns both long-term periodicities and short-term sequential regularities from correlated trajectories by combining an attention layer and a recurrent layer. Zhu *et al.* [33] propose Time-LSTM model that adds time gates into the LSTM network to capture the time factor to improve the spatiotemporal effect. To capture the spatiotemporal dependencies, Li *et al.* [34] further add spatiotemporal gates to the LSTM structure to capture the spatio-temporal relationships between successive check-ins. Huang *et al.* [18] propose ATST-LSTM model that leverages an attention mechanism to learn weights for each check-in in an LSTM network. Sun *et al.* [17] develop LSTPM model consisting of a nonlocal network for long-term preference modeling and a geo-dilated RNN for short-term preference learning. Rao *et al.* [19] propose Graph-Flashback that incorporates the learned POI transition graph into RNN-based models for better capturing the sequential transition patterns. Recently, Lian *et al.* [23] propose a self-attention based location recommendation model GeoSAN that considers point-to-point interaction within the trajectory. To aggregate all relevant visits from user trajectory, Luo *et al.* [22] further propose a Spatio-Temporal Attention Network (STAN), which allows a point-to-point interaction between non-adjacent locations and non-consecutive check-ins with explicit spatiotemporal effect. *Nonetheless, previous location prediction approaches mainly consider capturing correlations between locations from observed mobility data, but neglect to use pre-training techniques to learn non-trivial spatiotemporal dependencies between locations from large-scale trajectory data to improve model performance.*

2.2 Trajectory Prediction

In the early days, people generally used traditional mathematical models or machine learning methods to address the trajectory prediction problem, including Markov models, hidden Markov models, and Bayesian models. Asahara *et al.* [13] propose a hybrid Markov chain model for pedestrian trajectory prediction, which considers the relationship between the individual and the whole. Gambs *et al.* [15] use an extended Markov chain model to personalize the prediction of the subsequent location. Zhao *et al.* [35] use a Bayesian n-gram model to learn the spatiotemporal dependencies of location sequences. Mo *et al.* [14] develop a hidden Markov-based model to solve the problem of individual mobility prediction. *However, these traditional methods cannot effectively simulate the complex dependencies of spatiotemporal trajectory data.*

Deep learning models have been proven to better model spatiotemporal characteristics of trajectory sequences [5, 6, 36]. Among them, RNN-based models are representative and have achieved satisfactory performance on a variety of prediction tasks [37]. Park *et al.* [27] propose an LSTM-based encoder-decoder model to predict the next location of the vehicle trajectory on the map grid. Li *et al.* [20] develop a multi-layer LSTM encoder-decoder model in which the temporal attention mechanism is used to enhance the sequence learning ability for human mobility prediction. Capobianco *et al.* [38] leverage the attention mechanism to enhance the recurrent network model, which is applied to vessel trajectory prediction. CNN-based models are also used for mobility sequence prediction and trajectory prediction [39, 40]. *Nonetheless, these RNN models (e.g., LSTM) can hardly capture long-term dependencies in trajectory data [28], and can not run in parallel.*

Challenged by the flaws of RNN-based models, self-attention models thrive to replace them due to the advantages of better performance, fewer parameters, and parallel operation. Self-attention models are first applied to sequential recommendations [41, 42] and then widely used for location sequence prediction [22, 23, 43]. Lin *et al.* [24] propose CTLE that is a pre-trained model and applies a Transformer encoder to calculate contextual embeddings for next location prediction. In the follow-up work [26], they further propose a TALE pre-training method based on the CBOW framework, which is able to incorporate temporal information into the learned embedding vectors of locations. Shao *et al.* [2] use attention fusion dynamic GCN to obtain trajectory representation for trajectory prediction, and use two auxiliary tasks (*i.e.*, arrival time estimation and ranking of similarity weights) to optimize the model. *However, these models do not adequately exploit the potential of pre-training techniques and contrastive learning for trajectory prediction on large-scale trajectory data.*

3 Problem Definition

In this section, we first introduce the basic concepts used in this work, and then formally define the studied problem.

Definition 1 (Road Network). *A road network is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the collection of vertices, each representing an intersection, and $\mathcal{E}$ is the collection of edges between vertices, i.e., the collection of road segments. Each edge $e_{i,j} \in \mathcal{E}$ denotes the road segment from vertex v_i to vertex v_j.*

Definition 2 (Vehicle Trajectory). *A trajectory of a vehicle u_i is defined as a sequence of location points in chronological order $Tr_i = \langle (t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n) \rangle$, where n is the number of location points in Tr_i, and p_i represents the geographic coordinates of the i-th location point, i.e., latitude and longitude.*

We use TR_i and TR to denote the set of all trajectories of vehicle u_i and the set of all trajectories of all vehicles, respectively.

Problem (Vehicle Trajectory Prediction). *Given all trajectories of all vehicles TR and the road network $\mathcal{G}$, our goal is to learn a model $\hat{f}$ to predict future vehicle trajectory of the next τ time steps for any given vehicle: $\langle (t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n) \rangle \xrightarrow{\hat{f}} \langle (t_{n+1}, p_{n+1}), (t_{n+2}, p_{n+2}), \ldots, (t_{n+\tau}, p_{n+\tau}) \rangle$.*

The main notations used in this work are summarized in Table 1.

Table 1: Main notations and their definitions.

Notation	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	the road network
$e_{i,j}$	the road segment $e_{i,j}$
Tr_i	a trajectory of vehicle u_i
n	the number of location points in a trajectory
TR_i	the set of all trajectory of vehicle u_i
TR	the set of all trajectory of all vehicles
u, t, p	the user, time, and location in trajectories
x_i^u	the embedding of user u_i
x_j^t, x_j^p	the embeddings of time and location for j-th location
$\mathbf{X}$	the embedding matrix of location sequences
$\mathbf{H^g}$	the trajectory representation matrix based on gridding
$\mathbf{H}^r$	the trajectory representation matrix based on road network
$\mathbf{M}$	the mask matrix
$\mathbf{H}_{imp}^p, \mathbf{H}_{imp}^r$	the imputed trajectory representation matrix
Z_{des}^p, Z_{des}^r	the grid code of predicted trajectory destination
$\mathbf{U}_{tul}^p, \mathbf{U}_{tul}^r$	the user matrix of trajectories
$\hat{\mathbf{Y}}_{(n+1):(n+\tau)}$	the predicted future trajectory matrix
Π/Ω	the number of trajectories in training/testing set

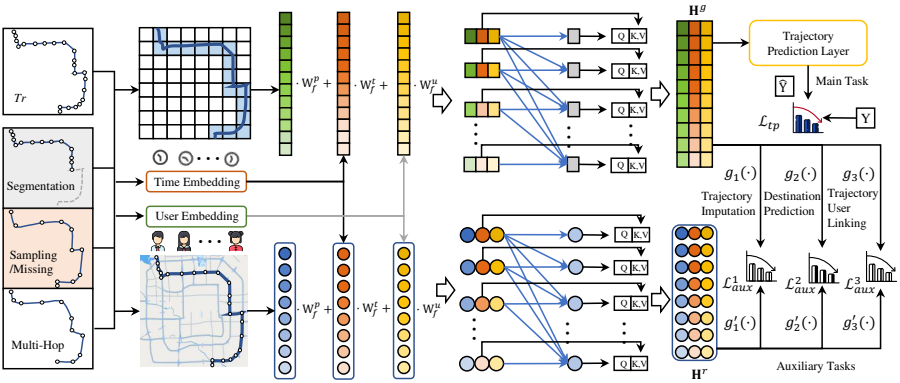


Fig. 1: The overall architecture of the proposed PreCLN.

4 Methodology

In this section, we present the details of our proposed PreCLN, which consists of four major components: (1) a *multimodal embedding module* that learns the dense embedding vectors of users, locations, and time steps in vehicle trajectories; (2) a *dual-view contrastive learning network* that uses Transformer-based encoders to generate dual-view representation vectors of input trajectory for both auxiliary pre-training tasks and main task; (3) a *multi-auxiliary task pre-training module* that utilizes pre-training techniques with three auxiliary tasks to enhance the representation learning ability of the model to improve the performance of main task; (4) a *trajectory prediction layer* that uses an attention-based predictor to estimate future location sequences using learned trajectory representations. The overall architecture of the proposed PreCLN is shown in Figure 1.

4.1 Multi-modal Embedding Module

This module contains two sub-modules: *trajectory data augmentation* and *user and time aware location embedding layer*.

4.1.1 Trajectory Data Augmentation

In contrastive learning models, data augmentation is an essential step, and positive samples are obtained through data augmentation. Nevertheless, in our model, the purpose of data augmentation is to obtain a larger amount of more diverse pre-training trajectory samples. Therefore, trajectory data augmentation not only needs to preserve the travel semantics of trajectories, but also needs to consider the differences between different samples.

Based on the above analysis, we explore the following three different trajectory augmentation strategies:

- Multi-hop Augmentation. Multi-hop enhancement is to sample a multi-hop sub-trajectory from the original trajectory according to the given number of hops. For example, when the hop count is set to 2, a 2-hop sub-trajectory is sampled, *i.e.*, $\langle (t_1, p_1), (t_3, p_3), \ldots, (t_{2\lfloor n/2 \rfloor+1}, p_{2\lfloor n/2 \rfloor+1}) \rangle$ or $\langle (t_2, p_2), (t_4, p_4), \ldots, (t_{2\lfloor n/2 \rfloor}, p_{2\lfloor n/2 \rfloor}) \rangle$ are sampled for original trajectory $Tr_i = \langle (t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n) \rangle$.
- Downsampling/Missing Augmentation. Downsampling enhancement is to randomly sample a sub-trajectory from the original trajectory according to a given sampling ratio. For example, when the sampling ratio is set to 50%, the probability of each location point being sampled is 50%. Missing Augmentation is to use different mask matrices to hide some location points in trajectories to form the missing trajectories. Both methods can get sub-trajectories with the same effect.
- Segmentation Augmentation. Segmentation augmentation is to truncate the long trajectory according to a certain proportion to obtain multiple sub-trajectory segments. Give a trajectory $Tr_i = \langle (t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n) \rangle$, we can get sub-trajectory segments $Tr_i^1 = \langle (t_1, p_1), (t_2, p_2), \ldots, (t_j, p_j) \rangle$, $Tr_i^2 = \langle (t_{j+1}, p_{j+1}), (t_{j+2}, p_{j+2}), \ldots, (t_n, p_n) \rangle$ through segmentation augmentation.

4.1.2 User and Time Aware Location Embedding Layer

This layer is used to encode user, time, and location and fuse them into dense embedding vectors.

User and Time Embedding. Many previous studies have shown that user and temporal information of trajectories have an important impact on trajectory sequence prediction tasks [22, 23]. User information reflects users' spatiotemporal preferences and is beneficial for personalized modeling. For example, some users have to commute between home and the workplace every day. Meanwhile, the time dimension contains the specific day, week, hour, and minute information of each trajectory location, reflecting the periodicity of trajectory data. Therefore, we consider the user information in our model and learn the embedded user representation through:

$$x_i^u = \tanh\left(u_i \cdot \mathbf{W}_u + b_u\right) \tag{1}$$

where u_i is the one-hot encoding of the i-th user, and $\mathbf{W}_u$ and b_u are learnable parameters.

Similarly, we first encode time information into time vector $t_j = [t_j^w, t_j^d, t_j^h, t_j^m, t_j^s]$ where t_j^w, t_j^d, t_j^h, t_j^m and t_j^s are the week, day, hour, minute, and second respectively, and then obtain the embedded time representation through:

$$x_j^t = \tanh\left(t_j \cdot \mathbf{W}_t + b_t\right) \tag{2}$$

where $\mathbf{W}_t$ and b_t are learnable parameters.

Location Embedding. Location information is of great significance for trajectory representation learning, and effective location embedding is beneficial

to obtain a superior trajectory representation as it captures spatial correlations effectively. The exact locations of vehicle trajectories are usually described by GPS coordinates, *i.e.*, latitude and longitude. Though they are continuous, they are difficult to directly be applied to deep learning models since GPS coordinates have poor discreteness. On the other hand, vehicle trajectory data is constrained by the road network, and considering the road network information can better characterize the representation of the trajectory [44].

To this end, we present two location encoding methods that embed the exact coordinates of location into the unique id:

- Location encoding based on hierarchical map gridding: following [23], we first use a hierarchical map gridding method to map latitude and longitude into a grid. Specifically, each geographic area can be divided into four sub-areas according to a cross, and the four sub-areas are respectively encoded with 0, 1, 2, and 3 according to a fixed rule. Each sub-area can be recursively divided into four sub-areas and encoded again. Hence, any GPS coordinate can be mapped to a grid with a quadkey code. For example, if the gridding level is set to 3, the length of the quadkey code for each grid is 3, and the i-th digit indicates the grid code of the i-th lever where the GPS coordinate is located.
- Location encoding based on road network mapping: we use the Fast Map Matching (FMM)[1] algorithm [45] to map vehicle trajectories to the road network, and use road network node sequence to represent each trajectory. Given a trajectory $Tr_i = \langle p_1, p_2, \ldots, p_n \rangle$, we can get a node sequence $Tr_i^R = \langle v_1, v_2, \ldots, v_m \rangle$ using FMM. To consider spatial correlation, we further encode road network nodes through:

$$x_i^v = \sum_{j=1}^{K} \frac{1}{|e_{i,j}|} \vec{v}_j + \vec{v}_i, \tag{3}$$

where $|e_{i,j}|$ represents the length of road segment $e_{i,j}$, K is the number of nodes connected to node v_i, and $\vec{v}_i$ and $\vec{v}_j$ are the one-hot vectors of nodes v_i and v_j, respectively.

Embedding Fusion. To incorporate user and time information, we further use a fusion layer to obtain user and time aware location embeddings:

$$x_j^p = \tanh([x_i^u \cdot \mathbf{W}_f^u + b_f^u;\ x_j^t \cdot \mathbf{W}_f^t + b_f^t;\ p_j \cdot \mathbf{W}_f^p + b_f^p]), \tag{4}$$

where p_j can be the location code based on hierarchical map gridding or road network mapping, and $\mathbf{W}_f^u, \mathbf{W}_f^t, \mathbf{W}_f^p, b_f^u, b_f^t$, and b_f^p are learnable parameters.

[1] https://github.com/cyang-kth/fmm

4.2 Dual-view Contrastive Learning Network

In this section, we propose a dual-view trajectory contrastive learning framework that leverages trajectory representations learned from two different views for contrastive learning across auxiliary tasks.

4.2.1 Dual-view Trajectory Contrastive Learning Framework

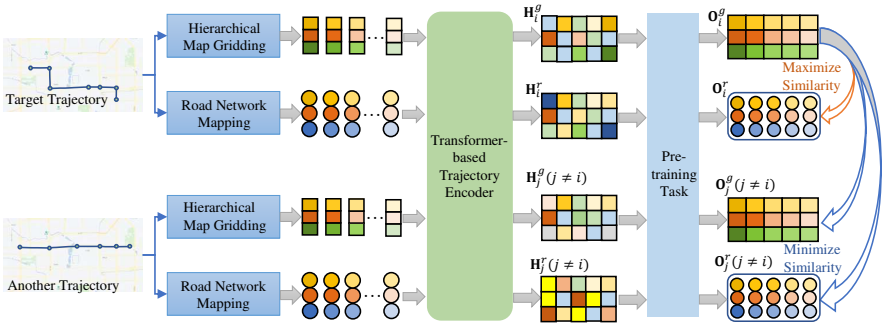


Fig. 2: Dual-view trajectory contrastive learning framework

Inspired by [46], we design a dual-view trajectory contrastive learning framework. As shown in Figure 2, the framework aims to achieve self-supervised pre-training. For an input trajectory, two trajectory representations are learned from two different views (*i.e.*, hierarchical map gridding and road network mapping), respectively, and then fed into the same task to expect the same result (*i.e.*, maximize the similarity of outputs $\mathbf{O}_i^g$ and $\mathbf{O}_i^g$). That is, one view representation is regarded as the query, and the other view is regarded as the positive sample. On the other hand, learned representations of other trajectories in the same batch are treated as negative samples with respect to the query, *i.e.*, minimize the similarity of outputs ($\mathbf{O}_i^g$ and $\mathbf{O}_j^g$, $\mathbf{O}_i^g$ and $\mathbf{O}_j^r$) between them. To effectively model the long-term spatiotemporal correlations in trajectories, we build a Transformer-based trajectory encoder to embed the input trajectory sequences into fixed-length representation vectors. The auxiliary pre-training tasks are then applied to guide the training of the trajectory encoder. After the pre-training, the result trajectory encoder can be used to generate vehicle trajectories' representations for trajectory prediction.

4.2.2 Transformer-based Trajectory Encoder

In this section, we develop a Transformer-based trajectory encoder to capture the long-term dependencies in trajectory sequences. The Transformer-based trajectory encoder stacks multiple self-attention blocks, each consisting of a self-attention layer and a point-wise Feed-Forward Network (FFN). In particular, the self-attention layer takes the embedding matrix $\mathbf{X}$ of the location

sequence as input and feeds them into an attention module after converting it through three weight matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$:

$$\mathbf{S} = self\_att(\mathbf{X} + \mathbf{P}) = att((\mathbf{X} + \mathbf{P})\mathbf{W}^Q, (\mathbf{X} + \mathbf{P})\mathbf{W}^K, (\mathbf{X} + \mathbf{P})\mathbf{W}^V) \quad (5)$$

where $\mathbf{P}$ is the positional embedding matrix.

The attention module, *i.e.*, $att(\cdot, \cdot, \cdot)$, is the scaled dot-product attention:

$$att(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}. \quad (6)$$

The FFN is used to non-linearize the output of the self-attention layer and transform dimensions, which is a two-layer network, and the formula is defined as follows:

$$\mathbf{H} = ffn(\mathbf{S}) = max(0, \mathbf{S} \cdot \mathbf{W}_1^{ffn} + b_1^{ffn}) \cdot \mathbf{W}_2^{ffn} + b_2^{ffn}. \quad (7)$$

Residual connection and layer normalization are applied in the FFN and self-attention layers to stabilize and speed up the training process when stacking multiple self-attention blocks.

Notice that we use $\mathbf{H}^g$ and $\mathbf{H}^r$ to denote the trajectory representation matrix obtained from hierarchical gridding location encoding and road network mapping location encoding, respectively.

4.3 Multi-Auxiliary Task Pre-training

In this section, we present three auxiliary pre-training tasks employed in our model, *i.e.*, trajectory imputation, trajectory destination prediction, and trajectory-user linking.

4.3.1 Trajectory Imputation Task

The trajectory imputation task mainly predicts random hidden location points through trajectory context semantics, which has a strong correlation with the main task of trajectory prediction, significantly increasing the training sample space. Specifically, we first obtain a random mask matrix $\mathbf{M}$ for input trajectories according to a certain proportion, and then predict the missing location points through the learned trajectory representations.

The hidden operation using the mask matrix is:

$$TR^{'} = TR \odot \mathbf{M} \quad (8)$$

where $\odot$ is Hadamard product. $TR^{'}$ is the input of the task and TR is the prediction target.

In this task, we employ a two-layer fully connected layer to map the learned trajectory representations to the target trajectory representations:

$$\mathbf{H}_{imp}^{g} = g_1(\mathbf{H}^{g}) = max(0, \mathbf{H}^{g} \cdot \mathbf{W}_1^{imp} + b_1^{imp}) \cdot \mathbf{W}_2^{imp} + b_2^{imp}, \tag{9}$$

$$\mathbf{H}_{imp}^{r} = g_1'(\mathbf{H}^{r}) = max(0, \mathbf{H}^{r} \cdot \mathbf{W}_3^{imp} + b_3^{imp}) \cdot \mathbf{W}_4^{imp} + b_4^{imp}, \tag{10}$$

where $\mathbf{H}^{g}$ and $\mathbf{H}^{r}$ are learned trajectory representations from two different views, and $\mathbf{H}_{imp}^{g}$ and $\mathbf{H}_{imp}^{r}$ are the two corresponding imputed trajectory representations.

4.3.2 Trajectory Destination Prediction Task

To enhance the ability of long-term trajectory sequence prediction, we introduce a trajectory destination prediction pre-training task. The historical trajectory data is used as the model input, and the last location point of each trajectory is used as the destination. The grid code of the destination location can be obtained through a two-layer FFN:

$$Z_{des}^{g} = g_2(\mathbf{H}^{g}) = max(0, \mathbf{H}^{g} \cdot \mathbf{W}_1^{des} + b_1^{des}) \cdot \mathbf{W}_2^{des} + b_2^{des}, \tag{11}$$

$$Z_{des}^{r} = g_2'(\mathbf{H}^{r}) = max(0, \mathbf{H}^{r} \cdot \mathbf{W}_3^{des} + b_3^{des}) \cdot \mathbf{W}_4^{des} + b_4^{des}, \tag{12}$$

where Z_{des}^{g} and Z_{des}^{r} are the two destination grid codes learned by the model.

4.3.3 Trajectory-User Linking Task

The trajectory-user linking task enables the model to identify the correlation between trajectories and their users, which facilitates our model to capture user trajectory preferences in a personalized manner. All trajectory data is used as model input, and the users of the trajectory are used as training targets. We obtain the probability that each trajectory belongs to the user through a classifier:

$$\mathbf{U}_{tul}^{g} = g_3(\mathbf{H}^{g}) = softmax(max(0, \mathbf{H}^{g} \cdot \mathbf{W}_1^{tul} + b_1^{tul}) \cdot \mathbf{W}_2^{tul} + b_2^{tul}), \tag{13}$$

$$\mathbf{U}_{tul}^{r} = g_3'(\mathbf{H}^{r}) = softmax(max(0, \mathbf{H}^{r} \cdot \mathbf{W}_3^{tul} + b_3^{tul}) \cdot \mathbf{W}_4^{tul} + b_4^{tul}), \tag{14}$$

where $\mathbf{U}_{tul}^{g}$ and $\mathbf{U}_{tul}^{r}$ are the probability matrices of users.

4.4 Trajectory Prediction Layer

To consider the user and temporal information of the target trajectory when predicting future trajectory, we design an attention-based predictor:

$$\mathbf{Z} = predictor(\mathbf{H}^{g} | x_i^{u}, x_i^{t}) = att(f(x_i^{u}, x_i^{t}), \mathbf{H}^{g} \cdot \mathbf{W}^{P}, \mathbf{H}^{g}) \tag{15}$$

where $f(x_i^{u}, x_i^{t}) = \tanh([x_i^{u} \cdot \mathbf{W}^{u} + b^{u};\ x_i^{t} \cdot \mathbf{W}^{t} + b^{t}])$, $att(\cdot, \cdot, \cdot)$ is defined as Eq. (6), and $\mathbf{W}^{P}$ is the learnable weight matrix.

Finally we use a linear layer to map $\mathbf{Z}$ to the future location sequence:

$$\hat{\mathbf{Y}}_{(n+1):(n+\tau)} = Linear(\mathbf{Z}) = \mathbf{Z} \cdot \mathbf{W}^l + b^l. \tag{16}$$

where τ is the number of future time steps, and $\mathbf{W}^l$ and b^l are the learnable parameters.

Notice that we can realize multiple transformations from the trajectory representation to future location sequence by stacking multiple $att(\cdot,\cdot,\cdot)$ layers to obtain a robust trajectory predictor.

4.5 Loss Function

Trajectory prediction loss. The final output from the prediction decoder is the location-level trajectory prediction matrix $\hat{\mathbf{Y}}_{(n+1):(n+\tau)} = [\hat{y}_{n+1}, \hat{y}_{n+2}, \ldots, \hat{y}_{n+\tau}]$. We introduce the mean absolute error (MAE) to calculate the error between the predicted trajectory sequence and the real trajectory sequence as the loss function:

$$\mathcal{L}_{tp} = \mathbf{MAE}(\hat{\mathbf{Y}}_{(n+1):(n+\tau)}, \mathbf{Y}_{(n+1):(n+\tau)}) = \sum_{i=1}^{\Pi} \sum_{j=n+1}^{n+\tau} |\hat{y}_i^j - y_i^j|. \tag{17}$$

where Π denotes the number of trajectories in training set.

Contrastive learning loss. As described in Section 4.2.1, we treat the two different view representations of each trajectory as the query and the positive sample, and the representations of other trajectories in the same batch as negative samples. Then these trajectory representations are fed into the same task. To maximize the similarity of the task outputs of the query and the positive sample and the difference of the task outputs of the query and the negative samples, we employ a normalized temperature-scaled cross-entropy loss:

$$l_i^g = -\log \frac{exp(sim(\mathbf{O}_i^g, \mathbf{O}_i^r)/T)}{\sum_{k \in B, k \neq i}(exp(sim(\mathbf{O}_i^g, \mathbf{O}_k^g)/T) + exp(sim(\mathbf{O}_i^g, \mathbf{O}_k^r)/T))}, \tag{18}$$

$$l_i^r = -\log \frac{exp(sim(\mathbf{O}_i^r, \mathbf{O}_i^g)/T)}{\sum_{k \in B, k \neq i}(exp(sim(\mathbf{O}_i^r, \mathbf{O}_k^g)/T) + exp(sim(\mathbf{O}_i^r, \mathbf{O}_k^r)/T))}, \tag{19}$$

where $\mathbf{O}$ represents the task output, and $sim(\cdot,\cdot)$ is the similarity of two output results, B denotes the set of all samples in the same batch, and T denotes the temperature parameter. The contrastive learning loss function can be calculated by accumulating all losses across all batches:

$$\mathcal{L}_{cl} = \frac{1}{2\Pi} \sum_{i=1}^{\Pi} (l_i^g + l_i^r). \tag{20}$$

Auxiliary task loss. Furthermore, we also need auxiliary task loss function to complete the auxiliary pre-training to guide the learning of the model. Let $\hat{\mathbf{A}}$ denotes the auxiliary task prediction result, and $\mathbf{A}$ denotes the ground-truth result. The auxiliary task loss function is formulated as:

$$\mathcal{L}_{aux} = \mathbf{MAE}(\hat{\mathbf{A}}, \mathbf{A}). \tag{21}$$

Eventually, we integrate $\mathcal{L}_{tp}$, $\mathcal{L}_{cl}$ and $\mathcal{L}_{aux}$ into a joint learning framework through hyperparameters α and β:

$$\mathcal{L} = \mathcal{L}_{tp} + \alpha\mathcal{L}_{cl} + \beta\mathcal{L}_{aux} + \frac{\lambda}{2}\|\Theta\|^2, \tag{22}$$

where λ represents the hyperparameter for regularization, and Θ denotes the model parameters.

5 Experiment

5.1 Datasets

We use two publicly available real-world large-scale vehicle trajectory datasets to evaluate our model: Porto Taxi Trajectory Data and T-Drive Trajectory Data.

- Porto Taxi Trajectory Data. This dataset is from the ECML-PKDD competition [47], containing over 1.7 million complete trajectories collected from 442 taxis running in the city of Porto from 1st July 2013 to 30th June 2014. The sampling frequency is once every 15 seconds.
- T-Drive Trajectory Data. This dataset is a sample of T-Drive trajectory dataset [48] that contains one-week trajectories of 10,357 taxis from February 2 to February 8, 2008. The total number of points in this dataset is about 17 million and the total distance of the trajectories reaches 9 million kilometers.

The basic statistics of the two datasets are summarized in Table 2.

Table 2: Basic statistics of two datasets.

Dataset	#vehicles (users)	#trajectories	#locations	Time spans
Porto	442	1.7M	8.3M	2013/07/01-2014/06/30
T-Drive	10,357	2.1M	17.6M	2008/02/02-2008/02/08

5.2 Baseline Methods

We compare our method with the following state-of-the-art baselines:

- ST-LSTM [1]: This is a spatial-temporal LSTM model for next location prediction.
- STAN [22]: This is an attention-based model, which explicitly uses relative spatiotemporal information between POIs within the user trajectory.
- CTLE [24]: This is a contextual location embedding method that is built upon a bi-directional Transformer framework.
- TALE [26]: This is a time-aware location embedding method that incorporates temporal information through designing a novel temporal tree structure for hierarchical softmax calculation.
- Graph-Flashback [19]: This method uses GCN on the learned POI transition graph to learn the representation of POIs, and then feeds them into RNN-based models for location prediction.

5.3 Evaluation Metrics and Experiment Setting

5.3.1 Evaluation Metrics

In experiments, we use Mean Absolute Error (MAE) and Root of Mean Square Error (RMSE) to evaluate the effectiveness of models, which are defined as:

$$MAE = \frac{1}{\Omega\tau} \sum_{i=1}^{\Omega} \sum_{j=n+1}^{n+\tau} dist(y_i^j, \hat{y}_i^j) \tag{23}$$

$$RMSE = \sqrt{\frac{1}{\Omega\tau} \sum_{i=1}^{\Omega} \sum_{j=n+1}^{n+\tau} dist(y_i^j, \hat{y}_i^j)^2}, \tag{24}$$

where Ω denotes the number of trajectories in the test set, and $dist(\cdot, \cdot)$ represents the distance calculation function between two latitude and longitude location points.

5.3.2 Experiment Settings

In experiments, we randomly select 70%, 20%, and 10% of the data in each dataset as training, testing, and validation sets, respectively.

For baselines, we use the source code released by their authors, adopt the parameter settings recommended in their papers, and fine-tune them to be optimal. For our model, we adopt Adam with default parameter setting to optimize our objective functions. The depth of self-attention is set to 3 on both datasets. We set the grid size to 80 meters(*i.e.*, the level of gridding is 20), the learning rate to 0.001, and the batch size to 64. The embedding dimensions of user and time both are set to 12, and the representation dimension of trajectories (*i.e.*, $\mathbf{H}$) is set to 512. The blocking ratio for mask matrix is set to 30%. The multi-hop enhancement is set to two hops, the sampling ratio of down-sampling enhancement is set to 30%, and the segmentation enhancement adopts random truncation. The training epoch is set to 100 and the dropout

rate is set to 0.2, however, an early stopping mechanism is used to avoid overfitting problems. All models are trained on a Linux server with a 2.60GHz Intel(R) Xeon(R) Gold 6126 CPU and 2 × 16GB NVIDIA Tesla V100 GPU.

5.4 Experimental Results

5.4.1 Overall Performance

Table 3: Overall performance comparisons for trajectory prediction

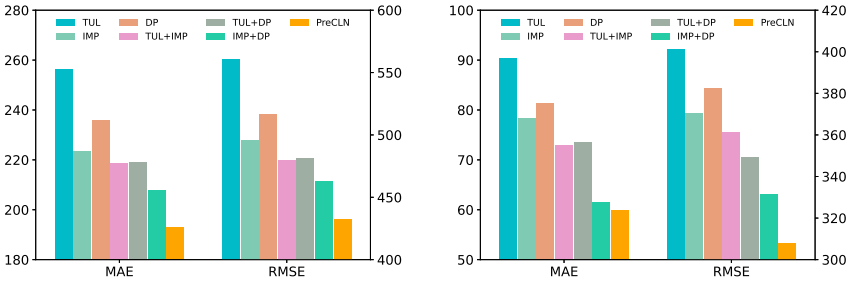
Dataset	Method	MAE	Std	RMSE	Std
Porto	ST-LSTM	876.59	11.39	3395.26	238.19
	STAN	613.24	8.36	2578.16	121.83
	CTLE	588.95	9.31	2385.60	89.61
	TALE	211.44	5.79	1636.35	32.43
	Graph-Flashback	209.37	7.11	464.29	22.19
	PreCLN	**193.15**	**4.99**	**432.18**	**21.86**
T-Drive	ST-LSTM	99.31	3.89	1029.74	29.03
	STAN	84.93	3.42	714.37	27.14
	CTLE	71.21	2.91	603.38	20.18
	TALE	68.38	2.73	693.16	23.86
	Graph-Flashback	59.33	2.69	337.91	17.83
	PreCLN	**55.90**	**2.35**	**307.95**	**12.49**

The comparison of PreCLN with all baselines on two real-world datasets is shown in Table 3, where the best is shown in bold.

From the evaluation results, we can observe that our PreCLN framework achieves the best prediction results as compared to state-of-the-art baselines. In particular, the relative performance improvement of our PreCLN over the best-performed baseline Graph-Flashback is 6.23% and 8.58% in terms of MAE and RMSE on T-Drive dataset. Although Graph-Flashback incorporates knowledge graph embedding and GCN to learn location representations and transition patterns between locations, it only relies on RNN model to learn temporal dependencies. Our PreCLN not only uses Transformer-based trajectory encoder to efficiently learn the mid- and long-term spatiotemporal dependencies in trajectories, but also utilizes contrastive learning to enhance the representation learning ability of the model. In addition, PreCLN significantly outperforms the pre-trained baseline TALE by average 14.91% and 52.36% improvements in terms of MAE and RMSE on two datasets, respectively. The main reason behind this is that our model combines dual-view contrastive learning and three auxiliary pre-training tasks to effectively model and capture spatiotemporal dependencies in trajectories that contribute to trajectory prediction, while TALE only uses pre-training techniques to enhance the ability to learn location representations that incorporate contextual semantics such as user and time. Additionally, the prediction performance superiority

can be observed in our PreCLN compared with all competitive methods, which validates the effectiveness of our pre-trained model with the integration of dual-view contrastive learn framework and auxiliary pre-training tasks.

5.4.2 Ablation Study



(a) Porto (b) T-Drive

Fig. 3: Ablation study on auxiliary tasks

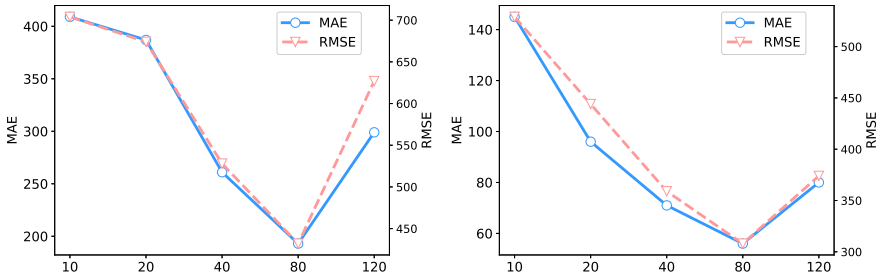
To verify the effect of each auxiliary task used in our PreCLN, we further conduct the ablation study. We compare our model with eight carefully designed variations. Despite the changed part(s), all variations have the same framework structure and parameter settings. The performance of all variations on two trajectory datasets are shown in Figure. 3

- **IMP** - This variation only considers trajectory imputation as auxiliary task during pre-training.
- **DP** - This variation only considers trajectory destination prediction as auxiliary task during pre-training.
- **TUL** - This variation only considers trajectory-user linking as auxiliary task during pre-training.
- **IMP+DP** - This variation considers both trajectory imputation and trajectory destination prediction as auxiliary tasks during pre-training.
- **TUL+IMP** - This variation considers both trajectory imputation and trajectory-user linking as auxiliary tasks during pre-training.
- **TUL+DP** - This variation considers both trajectory-user linking and trajectory destination prediction as auxiliary tasks during pre-training.

As we can see in Figure 3, all variants significantly perform worse than PreCLN, which fully validates the effectiveness of all auxiliary pre-training tasks used in our model in assisting trajectory prediction.

The variations with two tasks are generally better than the variations with one task (*i.e.*, (TUL+IMP)/(TUL+DP)/(IMP+DP) vs. TUL/IMP/DP),

which verifies the effectiveness of the pairwise combination of the three pre-training tasks. More specifically, from the comparisons among the three single-task variations, it can be seen that both IMP and DP are significantly better than TUL task on both metrics on two datasets, which reflects that trajectory imputation task and destination prediction task can better assist trajectory prediction compared to trajectory-user linking task. That is, for trajectory prediction, the enhancement of short- and long-term spatiotemporal dependence learning is more important than the enhancement of user preference learning. This may be because we have effectively fused user information through the multimodal embedding layer in trajectory representation learning. This is also verified by IMP+DP being significantly better than TUL+IMP and TUL+DP variations. PreCLN model using three auxiliary tasks further improves trajectory prediction performance, indicating that the three pre-training tasks can promote each other and strengthen the model's ability to learn effective trajectory representations from different aspects.



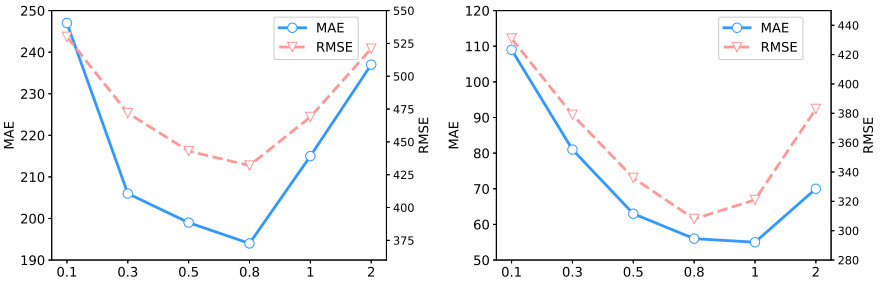
(a) Porto (b) T-Drive

Fig. 4: Model performance *w.r.t.* grid size

5.4.3 Parameter Study

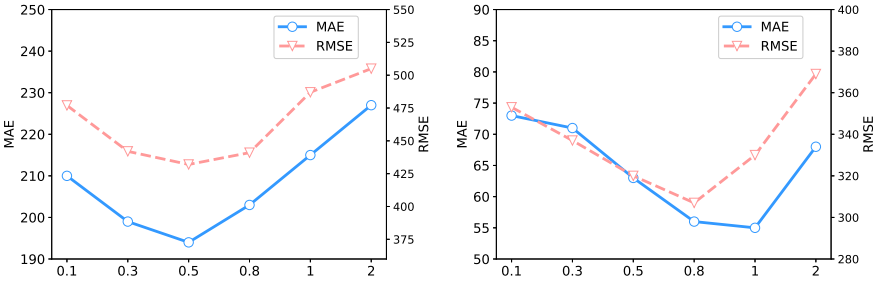
We finally investigate the sensitivity of our PreCLN with respect to the important parameters, including grid size and coefficients α and β. We report MAE and RMSE on two datasets with different parameter settings in Figure 4, Figure 5 and Figure 6.

As shown in Figure 4, at first, the prediction error of PreCLN decreases as the grid size increases, and then begins to increase when the grid size is larger than 80 meters. This is mainly because the grid size setting too large may lose the spatial information of trajectory locations, since many GPS points share a grid code. If the grid size is set too small, a large number of grids will be generated, which increases the grid code length and reduces the spatial correlation between adjacent GPS points, which in turn leads to poor prediction performance.



(a) Porto (b) T-Drive

Fig. 5: Model performance *w.r.t.* α



(a) Porto (b) T-Drive

Fig. 6: Model performance *w.r.t.* β

We further study the coefficients α and β which balance weights between main task and auxiliary pre-training tasks. As we can observe, the prediction error first drops and then rises as α and β increase on both datasets. Moreover, our model achieves relatively low prediction errors when α and β fall into a certain range. However, the performance of PreCLN is worse when α and β are closer to 0 or 2. This illustrates that trajectory prediction performance can be significantly improved by considering contrastive learning loss and/or auxiliary task loss, but too much focus on contrastive pre-training will also hurt trajectory prediction performance of the model. Additionally, as can be seen from the results, compared with auxiliary task loss (*i.e.*, β), considering contrastive learning loss (*i.e.*, α) can reduce the model prediction error faster in both MAE and RMSE, reflecting that the contrastive learning can integrate the high-level travel semantics of trajectories with the fine-grained gridding, which effectively the learning ability of trajectory representation.

Table 4: Effect of trajectory augmentation strategies

	with multi-hop		with downsmapling		with segmentation		w/o all	
Dataset	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Porto	223.56	453.85	209.54	391.56	217.60	446.26	275.43	563.29
T-Drive	73.55	389.56	60.57	336.81	68.54	372.29	90.51	427.95

5.4.4 Strategy Study

Effect of Augmentation Strategy.

We now evaluate the effect of the employed three trajectory data augmentation strategies. We report the experimental results in Table 4.

Based on the results on two datasets in Table 4, we have two observations: (1) All trajectory data augmentation strategies help improve model prediction performance. (2) The downsampling/missing augmentation performs better than the other two augmentation methods. The reason may be that the downsampling/missing augmentation method hides some location points by random sampling, so that more diverse sub-trajectory samples can be obtained, resulting in better results.

Table 5: Effect of trajectory encoder selection

	Hierarchical map gridding		Road network mapping	
Dataset	MAE	RMSE	MAE	RMSE
Porto	193.45	432.18	351.07	534.30
T-Drive	55.90	307.95	81.54	381.27

Effect of Trajectory Encoder Selection. We also evaluate the selection of different views of trajectory encoder. The experimental results on two datasets are shown in Table 5. We can conclude that the trajectory prediction performance of the trajectory encoder based on hierarchical map gridding is significantly better than that of the trajectory encoder based on road network mapping. This may be because the location encoding based on hierarchical map gridding uses more fine-grained spatial information, while the location encoding based on road network mapping only preserves the high-level transition patterns of trajectories. In addition, the goal of the trajectory prediction task is to predict fine-grained location points, and it is more reasonable to choose the trajectory encoder based on fine-grained location encoding view, *i.e.*, hierarchical map gridding.

6 Conclusion

In this work, we propose a novel pretrained-based contrastive learning Network PreCLN for vehicle trajectory prediction. It effectively captures the complex spatio-temporal dependencies in vehicle trajectories for trajectory prediction through the designed dual-view trajectory contrastive learning framework with assistance of three auxiliary pre-training tasks. The experimental results on two real-life large-scale vehicle trajectory datasets demonstrate the effectiveness and superiority of our proposed model for trajectory prediction.

Acknowledgments. We also acknowledge the editorial committee's support and all anonymous reviewers for their insightful comments and suggestions, which improved the content and presentation of this manuscript.

Declarations

Human and animal ethics. Not applicable.

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

Authors' contributions. Bingqi Yan gave the main idea of this paper, programmed the codes and made experiments. The first draft of the manuscript was written by Bingqi Yan and Geng Zhao. The material preparation includes figures, data analysis were performed by Bingqi Yan, Geng Zhao and Lexue Song. Junyu Dong gave suggestions and help to improve the manuscript. Yanwei Yu contributed significantly to preparation and modification of manuscript. All the authors read and approved the final manuscript of this paper.

Funding. This work was supported by the National Natural Science Foundation of China under grant Nos. 62176243, 61773331, and 41927805, and the National Key Research and Development Program of China under grant Nos. 2018AAA0100602 and 2019YFC1509100.

Data availability. Two datasets used in this work are publicly available. The Porto dataset can be downloaded at https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i and the T-Drive dataset can be found at https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/.

Ethics approval and consent to participate. Not applicable.

Consent for publication. All authors have read and agreed to the published version of the manuscript.

References

[1] Kong, D., Wu, F.: Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction. In: IJCAI, vol. 18, pp. 2341–2347 (2018)

[2] Shao, K., Wang, Y., Zhou, Z., Xie, X., Wang, G.: Trajforesee: How limited detailed trajectories enhance large-scale sparse information to predict vehicle trajectories? In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 2189–2194 (2021). IEEE

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

[4] Chen, Z., Xiao, X., Gong, Y.-J., Fang, J., Ma, N., Chai, H., Cao, Z.: Interpreting trajectories from multiple views: A hierarchical self-attention network for estimating the time of arrival. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2771–2779 (2022)

[5] Rao, X., Wang, H., Zhang, L., Li, J., Shang, S., Han, P.: Fogs: First-order gradient supervision with learning-based graph for traffic flow forecasting. In: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI (2022). ijcai. org

[6] Liu, D., Wang, J., Shang, S., Han, P.: Msdr: Multi-step dependency relation networks for spatial temporal forecasting. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1042–1050 (2022)

[7] Yu, Y., Tang, X., Yao, H., Yi, X., Li, Z.: Citywide traffic volume inference with surveillance camera records. IEEE Transactions on Big Data **7**(6), 900–912 (2019)

[8] Li, K., Shang, S.S., *et al.*: Towards alleviating traffic congestion:, optimal route planning for massive-scale trips. traffic **7**(v8), 9 (2020)

[9] Chen, L., Shang, S., Jensen, C.S., Yao, B., Zhang, Z., Shao, L.: Effective and efficient reuse of past travel behavior for route recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 488–498 (2019)

[10] Li, K., Shang, S., Kalnis, P., Yao, B., *et al.*: Traffic congestion alleviation over dynamic road networks: Continuous optimal route combination for trip query streams. (2021). International Joint Conferences on Artificial Intelligence Organization

[11] Shang, S., Ding, R., Yuan, B., Xie, K., Zheng, K., Kalnis, P.: User oriented trajectory search for trip recommendation. In: Proceedings of the 15th International Conference on Extending Database Technology, pp. 156–167

(2012)

[12] Shang, S., Chen, L., Wei, Z., Jensen, C.S., Wen, J.-R., Kalnis, P.: Collective travel planning in spatial networks. IEEE Transactions on Knowledge and Data Engineering **28**(5), 1132–1146 (2015)

[13] Asahara, A., Maruyama, K., Sato, A., Seto, K.: Pedestrian-movement prediction based on mixed markov-chain model. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 25–33 (2011)

[14] Mo, B., Zhao, Z., Koutsopoulos, H.N., Zhao, J.: Individual mobility prediction in mass transit systems using smart card data: an interpretable activity-based hidden markov approach. IEEE Transactions on Intelligent Transportation Systems (2021)

[15] Gambs, S., Killijian, M.-O., del Prado Cortez, M.N.: Next place prediction using mobility markov chains. In: Proceedings of the First Workshop on Measurement, Privacy, and Mobility, pp. 1–6 (2012)

[16] Dai, S., Yu, Y., Fan, H., Dong, J.: Personalized poi recommendation: spatio-temporal representation learning with social tie. In: International Conference on Database Systems for Advanced Applications, pp. 558–574 (2021). Springer

[17] Sun, K., Qian, T., Chen, T., Liang, Y., Nguyen, Q.V.H., Yin, H.: Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 214–221 (2020)

[18] Huang, L., Ma, Y., Wang, S., Liu, Y.: An attention-based spatiotemporal lstm network for next poi recommendation. IEEE Transactions on Services Computing **14**(6), 1585–1597 (2019)

[19] Rao, X., Chen, L., Liu, Y., Shang, S., Yao, B., Han, P.: Graph-flashback network for next location recommendation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1463–1471 (2022)

[20] Li, F., Gui, Z., Zhang, Z., Peng, D., Tian, S., Yuan, K., Sun, Y., Wu, H., Gong, J., Lei, Y.: A hierarchical temporal attention-based lstm encoder-decoder model for individual mobility prediction. Neurocomputing **403**, 153–166 (2020)

[21] Dai, S., Wang, J., Huang, C., Yu, Y., Dong, J.: Temporal multi-view graph convolutional networks for citywide traffic volume inference. In:

2021 IEEE International Conference on Data Mining (ICDM), pp. 1042–1047 (2021). IEEE

[22] Luo, Y., Liu, Q., Liu, Z.: Stan: Spatio-temporal attention network for next location recommendation. In: Proceedings of the Web Conference 2021, pp. 2177–2185 (2021)

[23] Lian, D., Wu, Y., Ge, Y., Xie, X., Chen, E.: Geography-aware sequential location recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2009–2019 (2020)

[24] Lin, Y., Wan, H., Guo, S., Lin, Y.: Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4241–4248 (2021)

[25] Guo, Q., Sun, Z., Zhang, J., Theng, Y.-L.: An attentional recurrent neural network for personalized next location recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 83–90 (2020)

[26] Wan, H., Lin, Y., Guo, S., Lin, Y.: Pre-training time-aware location embeddings from spatial-temporal trajectories. IEEE Transactions on Knowledge and Data Engineering (2021)

[27] Park, S.H., Kim, B., Kang, C.M., Chung, C.C., Choi, J.W.: Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1672–1678 (2018). IEEE

[28] Khandelwal, U., He, H., Qi, P., Jurafsky, D.: Sharp nearby, fuzzy far away: How neural language models use context. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 284–294 (2018)

[29] Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: A recurrent model with spatial and temporal contexts. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)

[30] Yao, D., Zhang, C., Huang, J., Bi, J.: Serm: A recurrent model for next location prediction in semantic trajectories. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2411–2414 (2017)

[31] Han, P., Li, Z., Liu, Y., Zhao, P., Li, J., Wang, H., Shang, S.: Contextualized point-of-interest recommendation. (2020). International Joint

Conferences on Artificial Intelligence

[32] Feng, J., Li, Y., Zhang, C., Sun, F., Meng, F., Guo, A., Jin, D.: Deepmove: Predicting human mobility with attentional recurrent networks. In: Proceedings of the 2018 World Wide Web Conference, pp. 1459–1468 (2018)

[33] Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., Cai, D.: What to do next: Modeling user behaviors by time-lstm. In: IJCAI, vol. 17, pp. 3602–3608 (2017)

[34] Li, X., Cong, G., Sun, A., Cheng, Y.: Learning travel time distributions with deep generative model. In: The World Wide Web Conference, pp. 1017–1027 (2019)

[35] Zhao, Z., Koutsopoulos, H.N., Zhao, J.: Individual mobility prediction using transit smart card data. Transportation research part C: emerging technologies **89**, 19–34 (2018)

[36] Wang, S., Cao, J., Yu, P.: Deep learning for spatio-temporal data mining: A survey. IEEE transactions on knowledge and data engineering (2020)

[37] Bahdanau, D., Cho, K., et al.: Neural machine translation by jointly learning to align and translate. arxiv preprint arxiv: 1409.0473 (2014)

[38] Capobianco, S., Millefiori, L.M., Forti, N., Braca, P., Willett, P.: Deep learning methods for vessel trajectory prediction based on recurrent neural networks. IEEE Transactions on Aerospace and Electronic Systems **57**(6), 4329–4346 (2021)

[39] Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 565–573 (2018)

[40] Chen, M., Zuo, Y., Jia, X., Liu, Y., Yu, X., Zheng, K.: Cem: A convolutional embedding model for predicting next locations. IEEE Transactions on Intelligent Transportation Systems **22**(6), 3349–3358 (2020)

[41] Kang, W.-C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 197–206 (2018). IEEE

[42] Li, Y., Chen, T., Zhang, P.-F., Yin, H.: Lightweight self-attentive sequential recommendation. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 967–977 (2021)

[43] Xie, J., Chen, Z.: Hierarchical transformer with spatio-temporal context aggregation for next point-of-interest recommendation. arXiv preprint arXiv:2209.01559 (2022)

[44] Ren, H., Ruan, S., Li, Y., Bao, J., Meng, C., Li, R., Zheng, Y.: Mtrajrec: Map-constrained trajectory recovery via seq2seq multi-task learning. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1410–1419 (2021)

[45] Yang, C., Gidofalvi, G.: Fast map matching, an algorithm integrating hidden markov model with precomputation. International Journal of Geographical Information Science **32**(3), 547–570 (2018)

[46] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, pp. 1597–1607 (2020). PMLR

[47] Kaggle: Kaggle Competition. Website. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i (2015)

[48] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: driving directions based on taxi trajectories. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 99–108 (2010)