Future Generation Computer Systems 148 (2023) 357–366

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Self-supervised contrastive representation learning for large-scale trajectories

Shuzhe Li [a,1], Wei Chen [a,1], Bingqi Yan [a,1], Zhen Li [a], Shunzhi Zhu [b,*], Yanwei Yu [a,*]

[a] College of Computer Science and Technology, Ocean University of China, 1299 Sansha RD, Qingdao, 266404, Shandong, China
[b] Xiamen University of Technology, 600 Ligong RD, Xiamen, 361024, Fujian, China

ARTICLE INFO

Article history:
Received 31 January 2023
Received in revised form 21 April 2023
Accepted 13 May 2023
Available online 19 June 2023

Keywords:
Representation learning
Large-scale trajectory
Contrastive learning
Trajectory augmentation

ABSTRACT

Trajectory representation learning aims to embed trajectory sequences into fixed-length vector representations while preserving their original spatio-temporal feature proximity. Existing works either learn trajectory representations for specific mining tasks or fail to utilize large amounts of unlabeled trajectory data for representation learning. In this work, we propose a self-supervised Trajectory representation learning based on Reconstruction Contrastive Learning called TrajRCL. To be specific, TrajRCL first obtains low-distortion and high-fidelity views of trajectories through trajectory augmentation. Then, TrajRCL leverages a Transformer based encoder–decoder network to reconstruct low-distortion view trajectories to approximate high-fidelity trajectories. Self-supervised contrastive learning is finally used to enhance the consistency of the two view's trajectory representations. Extensive experiments on two real-world demonstrate the superiority of our model over state-of-the-art baselines and significant efficiency on similarity trajectory search and k-NN query.

1. Introduction

With the rise of mobile devices and the maturity of positioning technology, location-based applications have been widely used in people's daily life. Various remote sensing satellites, electronic map navigation, and terminal equipment with GPS functions are collecting massive trajectory data [1], which includes not only mobility trajectories such as pedestrian travel and animal migration, but also the driving trajectories of vehicles [2]. The effective mining of such spatial–temporal data is the core foundation for various applications to provide intelligent services [3]. However, in reality, the spatial–temporal application scenarios are complex and changeable, and the collected spatial–temporal data become more complex due to factors such as multi-source, sampling frequency, accuracy, data missing, etc. The trajectory representation learning [4] aims to embed the original trajectory data from a variable-length coordinate-time stamp sequence into a fixed-length vector while maintaining the original spatial–temporal feature proximity, without manually designing various fixed trajectory measurement methods for various specific scenes. This is crucial for various downstream spatial–temporal data mining tasks, ranging from location recommendation (*e.g.*, predicting tourists' future visit preferences) [5,6], to traffic forecasting (*e.g.*, traffic flow prediction) [7,8], and to public security (*e.g.*, identifying abnormal trajectories) [9].

In recent years, representation learning technology has attracted widespread attention [10,11]. In spatial–temporal data mining, spatial–temporal representation learning has also been partially applied and studied. In literature, Li et al. [12] propose a deep trajectory representation method t2vec, which is similar to the traditional Seq2Seq model. The difference is that its decoder is to maximize the conditional probability of the input trajectory to its high sampling trajectory. Recently, Chen et al. [13] propose a trajectory-enhanced Transformer module, Toast, which uses trajectory data to extract driving semantics on the road network. In addition to obtaining effective road segment representation, this method can also obtain route representation. However, existing methods still have three key challenges to be solved: (1) Complexity: high complexity and large capacity of spatial–temporal data, often accompanied by sampling frequency uncertainty, data sparsity, and noise; (2) Learning paradigm: the trajectory data set with labels for specific tasks is very limited. How to effectively construct learning tasks in a self-supervised or unsupervised way to fully utilize the large-scale trajectory data set is challenging; (3) Applicability: how to design effective spatial–temporal representation learning model to obtain robust trajectory representation, and easily extend the model to various spatial–temporal data mining tasks, has not been extensively explored.

* Corresponding authors.
E-mail addresses: zhusz99@qq.com (S. Zhu), yuyanwei@ouc.edu.cn (Y. Yu).
[1] Authors contributed equally to this work.

https://doi.org/10.1016/j.future.2023.05.033

To tackle these challenges, we propose TrajRCL, a self-supervised trajectory representation learning model based on the reconstitution contrastive learning framework, to solve the problem of trajectory representation learning. In TrajRCL, we use a trajectory adaptive transformer-based neural network architecture as the backbone. Specifically, three different trajectory perturbation policies are designed to alter the original trajectory to generate different view trajectories. Then, the Transformer encoder–decoder is used to reconstruct low-distortion view trajectories to high-fidelity view trajectories. For the three tasks of masking, reconstructing, and contrastive learning, three loss functions are designed to jointly train the model. Additionally, we also design a multi-scale spatial-aware embedding layer that uses Hilbert coding to generate the embeddings of spatial–temporal trajectories, which are fed into the backbone for representation learning. We conduct extensive experiments to evaluate the model performance on two real-world trajectory datasets. The experimental results show the superiority of our model over several strong baselines on various trajectory mining tasks.

Our key contributions can be summarized as follows:

- We propose a self-supervised trajectory representation learning framework, TrajRCL, to obtain the effective trajectory representation for various downstream tasks. Our TrajRCL combines data augmentation, trajectory reconstruction, and self-supervised contrastive learning which can effectively capture the spatiotemporal dependences in trajectory data to obtain high-quality and robust trajectory representations.
- We design three loss functions for enhancing trajectory representation learning in TrajRCL to jointly train the model. Through the masking loss function, the encoder is forced to learn the semantic information of masked tokens. Through the reconstruction loss function, the decoder needs to reconstruct low-distortion trajectories to approximate high-fidelity trajectories. The trajectory representations learned by the encoder and decoder are forced to align via a contrastive loss function.
- Three downstream tasks are performed to evaluate the performance of the proposed model. The results on two real-world datasets demonstrate the effectiveness of the proposed model in learning trajectory representation in comparison to state-of-the-art baselines.

2. Related work

2.1. Representation learning

The goal of representation learning is to automatically embed the original data into the low-dimensional feature vectors, which can be effectively used as input feature information for various machine learning and deep learning models [11,14]. In the past few years, deep learning methods show their unique performance advantages in many fields, so deep representation learning algorithms receive extensive attention in various fields, including text processing [15], network analysis [16,17], recommendation [18,19], trajectory Planning [20–23], etc. Le et al. [24] propose an unsupervised text representation learning method, which can learn fixed-length feature representations from variable-length text fragments. Perozzi et al. [25] propose DeepWalk for multi-label network classification, and learn the latent representation of vertices in the network. Song et al. [26] propose a new metric learning scheme based on structured prediction, which aims to optimize clustering quality by grasping the global structure of the embedding space. Guo et al. [27] propose a method for Streaming Session-based Recommendation that leverages matrix factorization-based attention model and reservoir-based streaming model for efficiency. Zhang et al. [18] propose a self-supervised hypergraph learning framework for group recommendation to capture the intra- and inter-group interactions among users and alleviate the data sparsity issue with the raw data itself. Xia et al. [19] develop a self-supervised graph co-training framework for session-based recommendation to enhance data augmentation with genuine self-supervision signals. However, existing representation learning work is difficult to be directly applied to trajectory data with complex spatiotemporal features.

2.2. Self-supervised representation learning

Contrastive learning, as a form of self-supervised learning, plays a critical role in representation learning. With the emphasis on deep learning, comparative learning makes great progress in the field of representation learning, and several important researches are produced [28,29]. Some recent studies show that the success of contrastive learning can be attributed to the maximization of mutual information [30]. More precisely, the widely used infoNCE loss is a lower bound of the mutual information.

Existing studies [31–34] conduct contrastive learning at the instance level, use data augmentation to generate different views, and learn relevant data representations by comparing positive and negative samples. Chen et al. [31] propose a contrastive learning framework for visual representations and simplify contrastive self-supervised learning algorithms. Lin et al. [32] combine representation learning and data recovery into a unified framework from the perspective of information theory. He et al. [33] built a dynamic dictionary with a queue and a moving average encoder so that the learned representations can be efficiently transferred to downstream tasks. Self-supervised learning has obvious effects in addressing the effects of data sparsity and data noise in recommendation and trajectory analysis tasks. Sun et al. [35] propose a self-supervised hypergraph representation learning approach for sociological analysis to explore richer patterns under various sociological criteria. Jiang et al. [36] propose a self-supervised trajectory representation learning framework with temporal regularities and travel semantics, namely START, to convert raw trajectories into low-dimensional representation vectors by exploiting spatial–temporal characteristics such as temporal regularities and travel semantics.

2.3. Representation learning of trajectories

Trajectory representation learning can be considered as a special kind of representation learning for processing sequence data, which receive a lot of attention recently [11,23,37]. Therefore, it is natural to consider using an RNN-based encoder–decoder to learn representations of sequential data [23,37]. The traditional RNN-based encoder–decoder is designed for text data in natural language processing [38], where text documents have little noise and no time gap between words. To solve this problem, several models based on attention mechanism are proposed [39–42], and the time information is also encoded reasonably [43]. Trajectory data is not only a simple time series, but also contains complex spatial dependencies and road network dependencies. To incorporate road network information, some works extract road network graphs from real roads [44], and others encode spatial gridding trajectory sequences [44,45].

In recent years, trajectory representation learning has been applied to a variety of different trajectory mining tasks, such as trajectory prediction [46], location recommendation [34], traffic forecasting [7,47], outlier detection [48,49]. Li et al. [50] 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 representation. Capobianco et al. [51] leverage attention mechanism to enhance the recurrent network model, which is applied to vessel trajectory representation learning. CNN-based models are also used for mobility sequence representation and trajectory representation learning [52,53]. Recently, the self-attention model is used to replace RNN in trajectory sequence modeling [54]. Lin et al. [55] propose CTLE which is a pre-trained model and applies a Transformer encoder to calculate contextual embeddings for trajectory representation learning. In the follow-up work [56], 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.

A line of trajectory representation learning studies is proposed for trajectory similarity computation [17,57]. Li et al. [12] propose the first deep learning approach to learning representations of trajectory. Yang et al. [10] propose T3S to embed each trajectory into a vector in a d-dimensional space. Yao et al. [58] develop NEUTRAJ to collaborate with all spatial-based trajectory indexing methods to reduce the search space. Yang et al. [57] design a learning-based model to consider interactions between the trajectory pairs. However, most existing methods learn trajectory representation by approximating some traditional distance metrics as ground truth, while ignoring the exploration of new trajectory distance metrics based on the self-supervised paradigm of spatiotemporal features. Only a few studies have been carried out from this aspect, Liu et al. [59] propose a novel contrastive model to learn trajectory representations by distinguishing the trajectory-level and point-level differences between trajectories. Deng et al. [60] also learn the consistent representations of trajectories by applying trajectory data augmentations under the framework of contrastive learning. Compared wit them, we have unified the three joint training objectives of contrast, denoising and reconstruction to achieve more generalized trajectory representation and apply it to downstream tasks.

3. Problem definition

In this section, we first introduce the common definitions and then present problem formulations.

Definition 1 (*Real Path*)**.** The real moving path $\mathcal{T}$ of a moving object is a continuous spatial curve in the longitude and latitude domain, representing the exact path of the object during its movement.

Definition 2 (*Trajectory*)**.** A trajectory is the sample point (a, b) sequence of the real path, where a is longitude, b is latitude, and (a, b) is the trajectory point in geographic coordinates. Each trajectory can be expressed as $Tr = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, n is the length of the trajectory.

Due to the limitation of recording equipment, we cannot obtain the real path of the moving object, but only the raw trajectory. When the sampling rate is high enough, it can be approximated as the real path.

Based on the above definitions, we define the studied problem of this work as follows:

Problem 1 (*Trajectory Representation Learning*)**.** Given the large-scale trajectory dataset, our goal is to learn the low-dimensional vector representation $y \in \mathbb{R}^m$ (m is the embedding dimension) for each trajectory Tr such that the learned representation can reflect the real path of the trajectory, and thus be applied to various downstream trajectory data mining tasks.

4. Methodology

The overall architecture of the proposed TrajRCL is presented in Fig. 1. In particular, TrajRCL consists of three key components: *trajectory augmentation module*, *Transformer-based sequence modeling module*, and *self-supervised contrastive learning*. *Trajectory augmentation module* uses different data augmentation policies to obtain trajectory sequences in both low-distortion trajectory view and high-fidelity trajectory view. As the backbone of TrajRCL, the goal of *Transformer-based sequence modeling module* is to reconstruct the potential underlying path through the trajectory sequences, which is mainly divided into two parts: multi-scale spatial-aware embedding and Transformer encoder–decoder. *Self-supervised contrastive learning module* uses a weight-shared projection layer to obtain deeper trajectory embeddings, and introduces contrastive learning to constrain the trajectory representation learning between two different views to maximize the representation between different views consistency.

4.1. Trajectory augmentation

The real path $\mathcal{T}$ of a moving object is a continuous spatial curve (*e.g.*, in the latitude–longitude domain) representing the exact path taken by the object. In real-world data, a real path can be represented by different trajectories Tr, which is a sequence of sampling points of the underlying path of a moving object. Taking a trajectory $Tr_i = \{\langle a_1, b_1\rangle, \langle a_2, b_2\rangle, \ldots, \langle a_n, b_n\rangle\}$ as an example, we create two trajectory views for it: low-distortion trajectory Tr^i_{low} and high-fidelity trajectory Tr^i_{high}, respectively.

4.1.1. Low-distortion trajectory view

For the low-distortion trajectory view, we randomly adopt two strategies of downsampling and dynamic distortion to obtain its corresponding trajectory sequence:

- **Downsampling**. For each trajectory sequence with a given length n, we dynamically select 20%–60% of trajectory points for random masking. The remaining trajectory sequence is the low-distortion trajectory view for the original trajectory.
- **Dynamic Distortion**. For each trajectory sequence with a given length n, we dynamically select 20%–60% of the trajectory points for distorting, and the distorted trajectory sequence is the low-distortion trajectory view. Specifically, point $\langle a_i, b_i\rangle$ is distorted by adding Gaussian noise with a radius of 50 m, as follows:

$$\begin{cases} a_i = a_i + 50 \cdot \epsilon, & \epsilon \sim \text{Gaussian}(0, 1) \\ b_i = b_i + 50 \cdot \epsilon, & \epsilon \sim \text{Gaussian}(0, 1) \end{cases}. \tag{1}$$

4.1.2. High-fidelity trajectory view

For the high-fidelity trajectory view, we randomly adopt two strategies of original preservation and linear interpolation to obtain the corresponding trajectory sequence:

- **Original Preservation**. Do nothing to the trajectory, and use its original trajectory sequence as a high-fidelity trajectory view.
- **Linear Interpolation**. For a trajectory sequence of given length n, we dynamically randomly select trajectory point pairs of 10%–20%, and use linear interpolation technology to add new points in the middle of point pairs. The new trajectory sequence serves as a high-fidelity trajectory view.

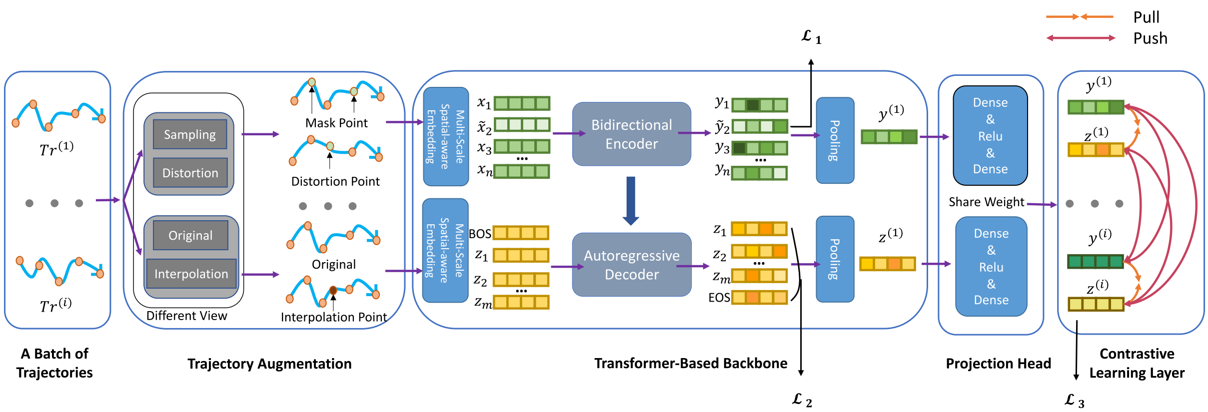


Fig. 1. The overall architecture of our proposed TrajRCL.

4.2. Transformer-based sequence modeling

Due to the natural sequential nature of trajectories, we choose Transformer encoder–decoder as our backbone network, with the goal of reconstructing the underlying path R from the trajectory sequence Tr_i, maximizing the conditional probability $\mathbb{P}(R \mid Tr_i)$. However, due to the unavailability of the underlying path, we replace maximizing objective $\mathbb{P}(R \mid Tr_i)$ with maximizing objective $\mathbb{P}\left(Tr_{high} \mid Tr_{low}\right)$, and use an encoder–decoder module with a self-attention network as the backbone. Specifically, the module is mainly composed of two parts: multi-scale spatial-aware embedding module and Transformer encoder–decoder module.

4.2.1. Multi-scale spatial-aware embedding module

We first design a novel multi-scale spatial encoding method, using the Hilbert Curve to encode the latitude and longitude points in the two-dimensional space into binary forms in the corresponding real number domain. This encoding method well preserves the spatial structure of the original data, while increasing the density of multi-scale spatial information. Specifically, different view trajectories Tr_* are embedded into the low-dimensional space vector H_* through a linear layer:

$$\begin{cases} X_{\text{low}} = \sigma\left(W_{\text{low}} H_{\text{low}} + b_{\text{low}}\right), & H_{\text{low}} = \text{Hilbert}\left(Tr_{\text{low}}\right) \\ Z_{\text{high}} = \sigma\left(W_{\text{high}} H_{\text{high}} + b_{\text{high}}\right), & H_{\text{high}} = \text{Hilbert}\left(Tr_{\text{high}}\right) \end{cases}, \tag{2}$$

where H_* is the representation of trajectory Tr_* obtained through Hilbert curve transformation, σ is the Leaky ReLU activation function with a leaky rate of 0.2, W_* and b_* are the trainable parameters, and X_{low} and Z_{high} are the low-dimensional embedding of low-distortion view and high-fidelity view trajectories obtained by multi-scale spatial-aware embedding module, respectively.

When encoding trajectory data using the Hilbert curve, each longitude–latitude point (a_i, b_i) in a trajectory Tr is first converted to a two-dimensional point (x_i, y_i) in a Cartesian coordinate system. Next, We use the Hilbert curve to map these points to a one-dimensional index space [61], where each point corresponds to a unique index value on the curve. The index values are represented using binary numbers, and concatenated to form a single vector as an alternative to coordinate points. After transformation, a vector sets H can be obtained. This process can be represented as:

$$H = \text{Hilbert}(Tr) = \{h_1, h_2, \ldots, h_n\}, \tag{3}$$

where each h_i represents the index of the ith longitude-latitude point on the Hilbert curve, and n represents the length of the trajectory. The Hilbert curve transformation enables the preservation of the spatial structure of trajectory data while increasing the density of multi-scale spatial information, thus realizing the coding and compression of trajectory data.

4.2.2. Transformer encoder–decoder module

As our work is to solve the problem of trajectory sequence representation, we employ Transformer network architecture as the backbone network, and its encoder embeds sequence representation H_{low} of the low-distortion view trajectory T_{low} by the bidirectional encoding, so that the embedding of each trajectory point through the encoder can effectively fuse the spatial information on the entire trajectory. Its decoder utilizes an autoregressive language model to reconstruct high-fidelity view trajectories from low-distortion trajectories.

For the Transformer encoder, we stack several transformer layers, each layer consisting of a causal masked multi-head self-attention module and a position-wise feed-forward network (FFN) module. Position-wise FFN will output a bag of embeddings, where the embedding at each position predicts the corresponding next point of the sequence. Residual connections and normalization have been applied to both modules.

For the lth layer, the input $\mathbf{X}^{(l)} \in \mathbb{R}^{m \times d}$ is firstly transformed by the multi-head self-attention module. The output of the first attention head is:

$$\text{head}^{\#1} = \text{softmax}\left(\frac{\mathbf{X}^{(l)}\mathbf{W}^{Q}\left(\mathbf{X}^{(l)}\mathbf{W}^{K}\right)^{\top}}{\sqrt{d}}\right)\mathbf{X}^{(l)}\mathbf{W}^{V}, \tag{4}$$

where $\mathbf{W}^{Q}, \mathbf{W}^{K}, \mathbf{W}^{V} \in \mathbb{R}^{d \times d/h}$ are the weight matrices corresponding to "Query", "Key" and "Value", respectively. $\mathbf{X}^{(0)}$ is the learned embedding matrix in the input low-distortion trajectory. Then, we stack multiple attention heads and merge the outputs from different attention heads by performing a linear transformation operation:

$$\text{MultiHead}\left(\mathbf{X}^{(l)}\right) = \left[\text{head}^{\#1}; \text{head}^{\#2}; \cdots; \text{head}^{\#h}\right] \times \mathbf{W}_o, \tag{5}$$

where $\mathbf{W}_o \in \mathbb{R}^{d \times d}$ is the learnable parameter matrix.

Next, layer normalization and residual connection on attention module are performed to obtain the final output of the attention module:

$$\mathbf{X}_{\text{ATT}}^{(l)} = \text{LayerNorm}\left(\mathbf{X}^{(l)} + \text{MultiHead}\left(\mathbf{X}^{(l)}\right)\right), \tag{6}$$

After the attention module, it will pass through a position-wise FFN:

$$\mathbf{X}_{\text{FFN}}^{(l)} = \max\left(0, \mathbf{W}_1\mathbf{X}_{\text{ATT}}^{(l)} + b_1\right)\mathbf{W}_2 + b_2, \tag{7}$$

where $\mathbf{W}_1, \mathbf{W}_2$ are trainable weight matrices, and b_1, b_2 are biases. Lastly, the output of the l-th encoder layer can be obtained:

$$\mathbf{X}^{(l+1)} = \text{LayerNorm}\left(\mathbf{X}_{\text{ATT}}^{(l)} + \mathbf{X}_{\text{FFN}}^{(l)}\right), \tag{8}$$

The decoder uses an autoregressive language model to reconstruct high-fidelity view trajectories through low-distortion trajectories. At the reconstruction step of each trajectory, the new decoder query $\mathbf{W}^{\mathbf{Q}}_{\mathbf{Dec}}$ is compared with the encoded keys $\mathbf{W}^K$ and values $\mathbf{W}^V$ according to Eq. (3) to complete the reconstruction of high-fidelity view trajectories.

In this part, we design two loss functions, which are the masking distortion trajectory reconstruction loss of the bidirectional encoder $\mathcal{L}_1$ and the overall decoding loss of the autoregressive decoder $\mathcal{L}_2$. $\mathcal{L}_1$ is used to capture the dynamic spatial proximity information of trajectory points, and $\mathcal{L}_2$ is used to reconstruct the high-fidelity view trajectory. The two loss functions are formalized as follows:

$$\mathcal{L}_1 = -\sum_{\tilde{x} \in M(x)} \log P\left(\tilde{x} \mid \bar{M}(x)\right), \tag{9}$$

$$\mathcal{L}_2 = -\log \prod_t P\left(z_t \mid z_{1:t-1}, \bar{M}(x)\right), \tag{10}$$

where $\bar{M}(x)$ denotes the embedding of unmasked or undistorted trajectory points in the low-distortion view trajectory, $\tilde{x} \in M(x)$ denotes the embedding of masked or distorted trajectory points in the low-distortion view trajectory, and z_t is the embedding of reconstructed tth trajectory point in the corresponding high-fidelity view trajectory.

Next, we send the embeddings of the low-distortion view trajectory through the encoder and the embeddings of the high-fidelity view trajectory decoded by the decoder to the maximum pooling layer to obtain the corresponding overall trajectory representations $y^{(i)}$ and $z^{(i)}$ as follows:

$$y^{(i)} = \text{MaxPooling}\left(y_1, y_2, \ldots, y_n\right), \tag{11}$$

$$z^{(i)} = \text{MaxPooling}\left(z_1, z_2, \ldots, z_n\right). \tag{12}$$

4.3. Self-supervised contrastive learning

After obtaining trajectory representations of different views, we further feed them into a weight-shared projection layer to obtain their deep representations. Contrastive learning has demonstrated its superiority in various representation learning applications. Inspired by that, we introduce a self-supervised contrastive learning framework to enhance the model's representation learning ability, maximizing the consistency of representations learned from different trajectory views.

We implement the operation of trajectory projection using two dense layers to obtain a deep representation:

$$\begin{aligned} y^{(i)} &= \mathbf{W}^{(2)} \sigma\left(\mathbf{W}^{(1)} y^{(i)} + b^{(1)}\right) + b^{(2)}, \\ z^{(i)} &= \mathbf{W}^{(2)} \sigma\left(\mathbf{W}^{(1)} z^{(i)} + b^{(1)}\right) + b^{(2)}, \end{aligned} \tag{13}$$

where $\mathbf{W}^{(*)}$ is the learnable weight vector and $b^{(*)}$ is the bias. σ is a leaky ReLU activation function with a leaky rate of 0.2.

In self-supervised contrastive learning, the deep representation pairs under different views of the same trajectory are regarded as positive sample pairs, while other samples under the same batch are regarded as negative samples. The goal of optimization is to make the different views (*i.e.*, low-distortion trajectory view and high-fidelity trajectory view) of the same trajectory sample as consistent as possible in the representation space and at the same time be as far away as possible from other negative samples in the same batch. The loss function $\mathcal{L}_3$ is formalized as follows:

$$\mathcal{L}_3 = -\log \frac{e^{\operatorname{sim}\left(y^{(i)}, z^{(i)}\right)/\tau}}{\sum_{j=1}^{N} e^{\operatorname{sim}\left(y^{(i)}, z^{(j)}\right)/\tau}}, \tag{14}$$

where τ is the temperature parameter, $\operatorname{sim}(\cdot,\cdot)$ is the cosine similarity function, and $(y^{(i)}, z^{(j)})$ represents a pair of negative samples.

4.4. Model learning

The overall loss $\mathcal{L}_{model}$ of the model is obtained by combining the trajectory reconstruction loss $\mathcal{L}_1$, the overall decoding loss $\mathcal{L}_2$, with the contrastive learning loss $\mathcal{L}_3$. More specifically, we optimize our model by maximizing the following objective function:

$$\mathcal{L}_{model} = \beta(\mathcal{L}_1 + \alpha \mathcal{L}_2) + \mathcal{L}_3. \tag{15}$$

where α and β are used to balance the importance of reconstruction loss, decoding loss, and contrastive learning loss.

5. Experiment

To verify the effectiveness of our proposed model, we conduct extensive experiments on two public real-world trajectory datasets.

5.1. Datasets

We evaluate TrajRCL on two public real-world trajectory datasets in two cities, *i.e.*, Porto taxi trajectory dataset and T-Drive trajectory dataset. The Porto dataset comes from the ECML-PKDD competition and contains more than 1.7 million complete trajectories collected from 442 taxis operating in the city of Porto from July 1, 2013 to June 30, 2014. Its sampling frequency is once every 15 s. The T-Drive dataset is a sample of the T-Drive trajectory dataset, which contains the weekly trajectories of 10,357 taxis from February 2 to February 8, 2008, and the total number of trajectory points is about 17 million.

5.2. Baseline methods

To evaluate the performance of our TrajRCL, we study the most similarity search problem and k-NN query problem on the Porto dataset, and compare TrajRCL with six classical trajectory distance measures: the Hausdorff distance [62], the Fréchet distance [63], Dynamic Time Warping (DTW) [64], Longest Common SubSequence (LCSS) [65], Edit distance with Real Plenty (ERP) [66], Edit Distance on Real sequence (EDR) [67] and the latest Contrastive Learning based Trajectory Similarity Computation (CL-TSim) [60] method.

We also evaluate the effectiveness of our TrajRCL for the trajectory prediction task on both Porto and T-Drive datasets, and compare our model with the following baselines:

- ST-LSTM [68]: This method is a long- and short-term trajectory prediction model considering spatial trajectory relationship.
- STAN [54]: This method explicitly exploits relative spatiotemporal information of all the points with self-attention layers along the trajectory.
- CTLE [55]: This is a Context and Time aware Location Embedding (CTLE) model, which calculates a location's representation vector with consideration of its specific contextual neighbors in trajectories.
- TALE [56]: This is a Time-Aware Location Embedding (TALE) pre-training method based on the CBOW framework, which is able to incorporate temporal information into the learned embedding vectors of locations.
- Graph-Flashback (G-Fback) [34]: This is a state-of-the-art graph-based model with strong location representation ability.
- PreCLN [46]: This is a pretrained-based contrastive learning network for vehicle trajectory prediction.

Table 1
Overall performance comparisons for most similar trajectory search on Porto dataset.

Method	5k	10k	20k	30k	40k	50k	60k	Time
Hausdorff	2.14	3.1	5.26	7.47	9.74	11.94	14.18	17886 s
Fréchet	2.11	2.99	4.91	6.96	9.06	10.88	12.79	34280 s
DTW	1.32	1.78	**2.12**	2.63	3.21	3.78	**4.32**	18420 s
LCSS	25.31	28.45	40.77	39.43	42.09	50.85	48.81	21060 s
ERP	41.41	78.59	152.62	229.07	307.07	383.06	460.09	22800 s
EDR	16.00	29.92	58.29	83.31	105.14	139.23	169.53	20340 s
CL-TSim	**1.21**	1.97	2.74	2.86	**3.05**	**3.47**	4.89	1764 s
TrajRCL	1.42	**1.61**	2.47	**2.58**	4.20	6.85	7.90	**68 s**

5.3. Experimental settings

One of the most important tasks in trajectory analysis is similar trajectory search. Following [12], we design two experiments (*i.e.*, most similar trajectory search and k-nearest-neighbors (k-NN) query) to evaluate the accuracy of methods for computing trajectory similarity using self-similarity and cross-similarity comparisons. Specifically, We randomly select 1,000 trajectories from the test dataset, denoted by Q, and then we select another n trajectories, denoted as P. For each trajectory $Tr_i \in Q$, we generate two sub-trajectories from it by the odd-even sampling of trajectory points, denoted as Tr_a and $Tr_{a'}$, and use them to construct two datasets $D_Q = \{T_a\}$ and $D_{Q'} = \{T_{a'}\}$. We perform the same operation for the trajectories in P to get D_P and $D_{P'}$. For each query $T_a \in D_Q$, we compute its top-k most similar trajectories from database $D_{Q'} \cup D_{p'}$ and calculate the rank of $Tr_{a'}$. Ideally, $Tr_{a'}$ is ranked at the top since it is produced from the same original trajectory as Tr_a.

Moreover, we further validate TrajRCL using the trajectory prediction task. Following [46], for each trajectory $Tr_i = \langle (t_1, lat_1, lon_1), (t_2, lat_2, lon_2), \ldots, (t_n, lat_n, lon_n)\rangle$, we encode the latitude lat_j and longitude lon_j, and then generate the trajectory location coding sequence to predict the future trajectory location sequence. Given all trajectories of all vehicles TR and the road network $\mathcal{G}$, our goal is to predict the future vehicle trajectory of the next Δ time steps for any given vehicle.

5.4. Performance evaluation for most similar trajectory search

We first evaluate the performance of our TrajRCL for the most similar trajectory search task compared to six trajectory distance measurement methods by increasing the number of trajectories from 5k to 60k. We report the mean rank of 1k queries in D_Q and search time on the 60k dataset on Porto dataset in Table 1.

It can be seen that the mean rank performance of all methods decreases as the number of trajectories in queried dataset increases. Among all methods, DTW achieves the best performance, and ERP achieves the worst performance. This may be because DTW considers the sequence information of the trajectory, but does not strictly obey the trajectory point order, while ERP introduces an interval as the threshold of edit distance, and it is not easy to determine a reasonable threshold for large-scale trajectory distance calculation. Although our TrajRCL does not achieve the best performance in most cases, it consistently achieves at least the second-best performance, which demonstrates that our proposed model can effectively capture the spatiotemporal dependencies in trajectories.

More importantly, one research motivation of trajectory representation learning is to speed up the efficiency of trajectory similarity computation for large-scale trajectories. As can be seen from the time results in Table 1, our TrajRCL achieves significant efficiency improvements. In particular, compared with DTW measurement, our model improves the efficiency by 271 times, even compared with the fastest baseline CL-TSim, it also improves the efficiency by 26 times.

5.5. Performance evaluation for k-NN query

We next evaluate the performance of different methods on Porto dataset for the kNN query task. In particular, we randomly select 10,000 trajectories from the test set as the target database, and 1,000 trajectories as the query database. We query the k-nearest neighbors for each trajectory from the target database as its ground truth. Then we transform the trajectories in both the query and target databases by randomly dropping or distorting points at a certain rate. Next, for each transformed query, we use each method to find its k-NNs from the target database, and then compare the result with the corresponding ground truth. Tables 2, 3, and 4 show the *Precision* results of all methods *w.r.t.* different dropping rates and distorting rates. The corresponding average query time for k-NN queries is also shown in the last column in each table.

From the results in the three tables, we have the following three findings: (1) As the dropping/distorting rate increases, the performance of all methods decreases. More specifically, dropping has a greater impact on model performance than distorting on the same scale. It is obvious that discarding trajectory points is more likely to change the spatial characteristics of the trajectory. (2) As k increases, the performance of our TrajRCL gets better and better. For example, when $k = 20$, our model achieves the best performance in most cases, but when $k = 40$ our model achieves the best performance in all cases. This also demonstrates that our model can learn the representations of similar trajectories more closely. (3) The query efficiency of our TrajRCL is far superior to other measurement methods. Specifically, our model is up to 78 times more efficient than the Fréchet measurement. Compared with the state-of-the-art CL-TSim for 40-NN query, the efficiency is improved by 8.2 times.

5.6. Performance evaluation for trajectory prediction

To further evaluate the trajectory representation ability of the model, we introduce a trajectory prediction task. Specifically, we use TrajRCL to learn trajectory representations, *i.e.*, y and z, and then concatenate them to obtain the final trajectory representation. Then we import the trajectory representation into an MLP trajectory decoder to predict future trajectory sequences. We adopt the MSE, RMSE, and their standard deviation as evaluation metrics to verify the performance of the proposed model.

Table 5 shows the comparison results of TrajRCL with the baseline models, where the best values are shown in bold. From all the results, we can see that TrajRCL achieves the best trajectory prediction performance, which demonstrates that our model has a strong learning ability to represent trajectories. In particular, the relative performance improvement of our TrajRCL over the best-performed baseline PreCLN is 6.1% and 8.3% in terms of MAE and RMSE on T-Drive dataset. TrajRCL also significantly outperforms the graph-based baseline Graph-Flashback by an average of 14.91% and 52.36% improvements in terms of MAE and RMSE on two datasets, respectively. Although PreCLN model considers the contrastive learning framework and three pre-training tasks,

Table 2
Overall performance comparisons for 20-NN queries on Porto dataset.

Method	$k = 20$, dropping rate					$k = 20$, distorting rate					Time
	0.2	0.3	0.4	0.5	0.6	0.2	0.3	0.4	0.5	0.6	
Hausdorff	0.5043	0.4021	0.2230	0.1542	0.1593	0.3712	0.2215	0.1342	0.1215	0.0852	12.18 s
Fréchet	0.5523	0.3673	0.2815	0.1725	0.1730	0.4125	0.2921	0.1354	0.1230	0.0850	68.19 s
DTW	0.0379	0.0300	0.0250	0.0279	0.0270	0.0269	0.0216	0.0178	0.0147	0.0143	19.76 s
LCSS	0.3720	0.3521	0.2315	0.1763	0.1763	0.5140	0.4521	0.4012	0.3536	0.3573	20.96 s
ERP	0.0325	0.0230	0.0190	0.0145	0.0145	**0.8170**	0.7700	**0.7590**	0.7035	0.6985	40.64 s
EDR	0.0130	0.0060	0.0040	0.0075	0.0075	0.0995	0.0790	0.0640	0.0565	0.0570	20.32 s
CL-TSim	0.6924	0.6243	0.5874	0.5079	0.5441	0.7164	0.6948	0.6793	0.6185	0.6523	7.23 s
TrajRCL	**0.7820**	**0.7048**	**0.6518**	**0.6045**	**0.6027**	0.8002	**0.7763**	0.7428	**0.7137**	**0.7125**	**0.87 s**

Table 3
Overall performance comparisons for 30-NN queries on Porto dataset.

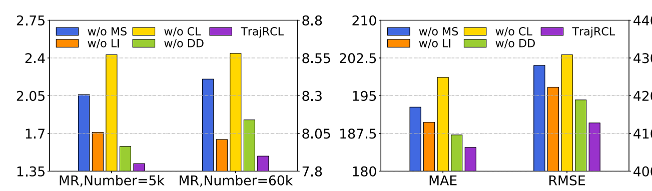
Method	$k = 30$, dropping rate					$k = 30$, distorting rate					Time
	0.2	0.3	0.4	0.5	0.6	0.2	0.3	0.4	0.5	0.6	
Hausdorff	0.4815	0.3819	0.2155	0.1342	0.1334	0.3568	0.2158	0.1275	0.1024	0.0880	12.46 s
Fréchet	0.5368	0.3564	0.2743	0.1656	0.1638	0.4052	0.2825	0.1348	0.1145	0.0879	69.75 s
DTW	0.0287	0.0233	0.0183	0.0197	0.0197	0.2467	0.1930	0.1533	0.1247	0.1217	20.21 s
LCSS	0.3964	0.3820	0.2640	0.2015	0.1995	0.5246	0.4785	0.4314	0.3840	0.3821	21.44 s
ERP	0.0233	0.0167	0.0147	0.0127	0.0127	**0.7983**	0.7577	0.7373	0.6830	0.6907	41.57 s
EDR	0.0127	0.0067	0.0047	0.0057	0.0057	0.0873	0.0677	0.0523	0.0453	0.0450	20.76 s
CL-TSim	0.6920	0.6345	0.5846	0.5077	0.5312	0.7125	0.6912	0.6702	0.6117	0.6433	7.64 s
TrajRCL	**0.7762**	**0.7021**	**0.6558**	**0.6017**	**0.6005**	0.7968	**0.7740**	**0.7385**	**0.7129**	**0.7014**	**0.89 s**

Table 4
Overall performance comparisons for 40-NN queries on Porto dataset.

Method	$k = 40$, dropping rate					$k = 40$, distorting rate					Time
	0.2	0.3	0.4	0.5	0.6	0.2	0.3	0.4	0.5	0.6	
Hausdorff	0.4835	0.3840	0.2214	0.1387	0.1380	0.3611	0.2248	0.1243	0.1079	0.1045	9.87 s
Fréchet	0.5413	0.3613	0.2790	0.1842	0.1830	0.4005	0.3029	0.1394	0.1144	0.1131	55.30 s
DTW	0.0237	0.0207	0.0167	0.0175	0.0175	0.2430	0.1883	0.1535	0.1233	0.1188	16.02 s
LCSS	0.4453	0.4230	0.3045	0.2482	0.2475	0.5573	0.5149	0.4720	0.4280	0.4275	17.00 s
ERP	0.0212	0.0148	0.0143	0.0115	0.0115	0.8022	0.7593	0.7388	0.6805	0.6882	32.96 s
EDR	0.0133	0.0070	0.0060	0.0063	0.0063	0.0825	0.0623	0.0513	0.0423	0.0417	16.46 s
CL-TSim	0.7142	0.6578	0.6050	0.5480	0.5744	0.7245	0.7102	0.6980	0.6443	0.6632	6.89 s
TrajRCL	**0.7963**	**0.7228**	**0.6745**	**0.6233**	**0.6220**	**0.8034**	**0.7903**	**0.7542**	**0.7200**	**0.7122**	**0.84 s**

Table 5
Overall performance comparisons for trajectory prediction.

Method	Porto				T-Drive			
	MAE	Std	RMSE	Std	MAE	Std	RMSE	Std
ST-LSTM	876.59	11.39	3395.26	238.19	99.31	3.89	1029.74	29.03
STAN	613.24	8.36	2578.16	121.83	84.39	3.42	714.37	27.14
CTLE	588.95	9.31	2385.60	89.61	71.21	2.91	603.38	20.18
TALE	211.44	5.79	1636.35	32.43	68.38	2.73	693.16	23.86
G-Fback	209.37	7.11	464.29	22.19	59.33	2.69	337.91	17.83
PreCLN	193.15	4.99	432.18	21.86	55.90	2.35	307.95	12.49
TrajRCL	**184.73**	**3.17**	**412.79**	**21.04**	**52.67**	**1.82**	**284.41**	**8.69**



(a) Most similar trajectory search (b) Trajectory prediction

Fig. 2. Experimental results of ablation study on Porto dataset.

it ignores the contrastive learning between different views of the trajectory and the original trajectory data. Our TrajRCL model can effectively learn representations from trajectory data through reasonable trajectory enhancement and appropriate use of instance pairs for comparative learning. In addition, our model is significantly higher than the pre-training methods such as TALE and CTLE in terms of MAE and RMSE. Through the reconstruction contrastive learning, on the one hand, our model effectively learns the key spatiotemporal dependencies of the trajectory through the reconstruction of low-distortion trajectories. On the other hand, our model also uses contrastive learning to effectively capture the common spatiotemporal dependencies in both high-fidelity view and low-distortion view trajectories.

5.7. Ablation study

To validate the effectiveness of each component in TrajRCL, we further conduct the ablation study. We compare our TrajRCL with four carefully designed variants.

- **w/o Dynamic Distortion (DD)**: This variant removes the dynamic distortion strategy in low-distortion trajectory view.
- **w/o Linear Interpolation (LI)**: This variant removes the linear interpolation strategy in high-Fidelity trajectory view.
- **w/o Multi-scale Spatial-aware embedding module (MS)**: This variant removes the multi-scale spatial-aware embedding module and uses a common compact layer to replace it.
- **w/o Contrastive Learning layer (CL)**: This variant removes the contrastive learning layer.

We conduct ablation experiments on the two tasks of the most similar trajectory search and trajectory prediction. In the most similar trajectory search experiment, we selected the number of trajectories 5k and 60k for comparison. As shown in Fig. 2, all variants exhibit similar performance trends on both tasks. All variants are significantly worse than the full model, demonstrating the effectiveness of the proposed sub-modules. In particular,

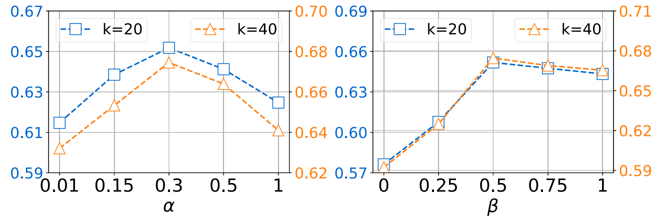


Fig. 3. Parameter sensitivity *w.r.t.* α and β on Porto dataset.

w/o CL variant shows the worst performance, which also shows that the contrastive learning module designed in this paper plays a major role in improving the model performance. Followed by the w/o MS variant, which also verifies the rationality of our proposed multi-scale spatial-aware embedded module. In addition, this experiment also shows that the joint use of multiple different trajectory enhancement strategies further improves the model performance.

5.8. Sensitivity study

We finally investigate the sensitivity of our TrajRCL with respect to the important parameters, including hyper-parameters α and β to balance the importance of reconstruction loss, decoding loss, and contrastive learning loss. We conduct sensitivity experiments on the *k*NN query. We select $k = 20$ and $k = 40$ and the dropping rate is 0.4 to evaluate the performance of our model. Results on Porto dataset are shown in Fig. 3. We can observe that: (1) When $\beta = 0$ (*i.e.*, ignoring $\mathcal{L}_1$ and $\mathcal{L}_2$), the performance is the worst, this emphasizes the importance of the reconstruction loss and decoding loss. (2) When α increases from 0.01, the performance of the model is significantly improved, which verifies the effectiveness of the decoding loss. When $\alpha = 0.3$, the performance of the model is the best, indicating that the reconstruction loss and decoding loss have reached a better balance.

6. Conclusion

In this paper, we present a self-supervised framework, TrajRCL, for learning effective trajectory representation. It combines data augmentation, reconstruction, and contrastive learning to capture dependencies in trajectories to obtain high-quality and robust trajectory representations. TrajRCL is then jointly trained with three designed loss functions to enhance trajectory representation learning. The performance of our proposed model is evaluated through three trajectory analytical tasks on two real-world datasets and results show the superiority of our model compared to state-of-the-art baselines. In future work, we plan to investigate efficient pre-trained representation learning techniques for large-scale multi-source multi-spatial-scale trajectory data.

CRediT authorship contribution statement

Shuzhe Li: Methodology, Software, Data curation, Validation, Visualization, Writing – original draft. **Wei Chen:** Methodology, Software, Validation, Formal analysis, Writing – original draft. **Bingqi Yan:** Methodology, Software, Visualization, Writing – review & editing. **Zhen Li:** Validation, Writing – original draft. **Shunzhi Zhu:** Writing – review & editing. **Yanwei Yu:** Conceptualization, Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The used data is publicly available.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under grant Nos. 62176243 and 61773331.

References

[1] L. Chen, S. Shang, K. Zheng, P. Kalnis, Cluster-based subscription matching for geo-textual data streams, in: 2019 IEEE 35th International Conference on Data Engineering, ICDE, IEEE, 2019, pp. 890–901.
[2] H. Zhang, X. Zhang, Q. Jiang, B. Zheng, Z. Sun, W. Sun, C. Wang, Trajectory similarity learning with auxiliary supervision and optimal matching, 2020.
[3] D. Liu, J. Wang, S. Shang, P. Han, MSDR: Multi-step dependency relation networks for spatial temporal forecasting, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1042–1050.
[4] S. Shang, R. Ding, K. Zheng, C.S. Jensen, P. Kalnis, X. Zhou, Personalized trajectory matching in spatial networks, VLDB J. 23 (3) (2014) 449–468.
[5] S. Feng, G. Cong, B. An, Y.M. Chee, Poi2vec: Geographical latent representation for predicting future visitors, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[6] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, D. Cai, What to do next: Modeling user behaviors by time-lstm., in: IJCAI, Vol. 17, 2017, pp. 3602–3608.
[7] K. Li, S. Shang, P. Kalnis, B. Yao, et al., Traffic congestion alleviation over dynamic road networks: Continuous optimal route combination for trip query streams, in: IJCAI, 2021, pp. 3656–3662.
[8] H. Ren, S. Ruan, Y. Li, J. Bao, C. Meng, R. Li, Y. Zheng, MTrajRec: Map-constrained trajectory recovery via Seq2Seq multi-task learning, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1410–1419.
[9] W. Chen, S. Li, C. Huang, Y. Yu, Y. Jiang, J. Dong, Mutual distillation learning network for trajectory-user linking, in: IJCAI, 2022.
[10] P. Yang, H. Wang, Y. Zhang, L. Qin, W. Zhang, X. Lin, T3s: Effective representation learning for trajectory similarity computation, in: 2021 IEEE 37th International Conference on Data Engineering, ICDE, IEEE, 2021, pp. 2183–2188.
[11] S. Shang, L. Chen, K. Zheng, C.S. Jensen, Z. Wei, P. Kalnis, Parallel trajectory-to-location join, IEEE Trans. Knowl. Data Eng. 31 (6) (2019) 1194–1207.
[12] X. Li, K. Zhao, G. Cong, C.S. Jensen, W. Wei, Deep representation learning for trajectory similarity computation, in: 2018 IEEE 34th International Conference on Data Engineering, ICDE, IEEE, 2018, pp. 617–628.
[13] Y. Chen, X. Li, G. Cong, Z. Bao, C. Long, Y. Liu, A.K. Chandran, R. Ellison, Robust road network representation learning: When traffic patterns meet traveling semantics, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 211–220.
[14] S. Shang, L. Chen, Z. Wei, C.S. Jensen, K. Zheng, P. Kalnis, Parallel trajectory similarity joins in spatial networks, VLDB J. 27 (3) (2018) 395–420.
[15] L. Chen, S. Shang, C.S. Jensen, J. Xu, P. Kalnis, B. Yao, L. Shao, Top-k term publish/subscribe for geo-textual data streams, VLDB J. 29 (5) (2020) 1101–1128.
[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[17] P. Han, J. Wang, D. Yao, S. Shang, X. Zhang, A graph-based approach for trajectory similarity computation in spatial networks, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 556–564.
[18] J. Zhang, M. Gao, J. Yu, L. Guo, J. Li, H. Yin, Double-scale self-supervised hypergraph learning for group recommendation, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 2557–2567.

[19] X. Xia, H. Yin, J. Yu, Y. Shao, L. Cui, Self-supervised graph co-training for session-based recommendation, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 2180–2190.
[20] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, P. Kalnis, User oriented trajectory search for trip recommendation, in: Proceedings of the 15th International Conference on Extending Database Technology, 2012, pp. 156–167.
[21] P. Han, Z. Li, Y. Liu, P. Zhao, J. Li, H. Wang, S. Shang, Contextualized point-of-interest recommendation, in: International Joint Conferences on Artificial Intelligence, 2020.
[22] K. Li, S.S. Shang, et al., Towards alleviating traffic congestion:, optimal route planning for massive-scale trips, Traffic 7 (v8) (2020) v9.
[23] S. Shang, L. Chen, C.S. Jensen, J.-R. Wen, P. Kalnis, Searching trajectories by regions of interest, IEEE Trans. Knowl. Data Eng. 29 (7) (2017) 1549–1562.
[24] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International Conference on Machine Learning, PMLR, 2014, pp. 1188–1196.
[25] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[26] H. Oh Song, S. Jegelka, V. Rathod, K. Murphy, Deep metric learning via facility location, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5382–5390.
[27] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, N. Quoc Viet Hung, Streaming session-based recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1569–1577.
[28] Y. Li, P. Hu, Z. Liu, D. Peng, J.T. Zhou, X. Peng, Contrastive clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 10, 2021, pp. 8547–8555.
[29] L. Chen, S. Shang, S. Feng, P. Kalnis, Parallel subtrajectory alignment over massive-scale trajectory data, in: IJCAI, 2021, pp. 3613–3619.
[30] Y.-H.H. Tsai, Y. Wu, R. Salakhutdinov, L.-P. Morency, Self-supervised learning from a multi-view perspective, 2020, arXiv preprint arXiv:2006.05576.
[31] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, PMLR, 2020, pp. 1597–1607.
[32] Y. Lin, Y. Gou, Z. Liu, B. Li, J. Lv, X. Peng, Completer: Incomplete multi-view clustering via contrastive prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11174–11183.
[33] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[34] X. Rao, L. Chen, Y. Liu, S. Shang, B. Yao, P. Han, Graph-flashback network for next location recommendation, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1463–1471.
[35] X. Sun, H. Cheng, B. Liu, J. Li, H. Chen, G. Xu, H. Yin, Self-supervised hypergraph representation learning for sociological analysis, IEEE Trans. Knowl. Data Eng. (2023).
[36] J. Jiang, D. Pan, H. Ren, X. Jiang, C. Li, J. Wang, Self-supervised trajectory representation learning with temporal regularities and travel semantics, 2022, arXiv preprint arXiv:2211.09510.
[37] S. Shang, L. Chen, Z. Wei, C.S. Jensen, J.-R. Wen, P. Kalnis, Collective travel planning in spatial networks, IEEE Trans. Knowl. Data Eng. 28 (5) (2015) 1132–1146.
[38] A. Madsen, S. Reddy, S. Chandar, Post-hoc interpretability for neural nlp: A survey, ACM Comput. Surv. 55 (8) (2022) 1–42.
[39] Q. Guo, Z. Sun, J. Zhang, Y.-L. Theng, An attentional recurrent neural network for personalized next location recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 01, 2020, pp. 83–90.
[40] L. Huang, Y. Ma, S. Wang, Y. Liu, An attention-based spatiotemporal lstm network for next poi recommendation, IEEE Trans. Serv. Comput. 14 (6) (2019) 1585–1597.
[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[42] L. Chen, S. Shang, T. Guo, Real-time route search by locations, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 01, 2020, pp. 574–581.
[43] W.X. Zhao, N. Zhou, A. Sun, J.-R. Wen, J. Han, E.Y. Chang, A time-aware trajectory embedding model for next-location recommendation, Knowl. Inf. Syst. 56 (2018) 559–579.
[44] D. Lian, Y. Wu, Y. Ge, X. Xie, E. Chen, Geography-aware sequential location recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2009–2019.
[45] C. Yang, L. Chen, S. Shang, F. Zhu, L. Liu, L. Shao, Toward efficient navigation of massive-scale geo-textual streams, in: IJCAI, 2019, pp. 4838–4845.
[46] B. Yan, G. Zhao, L. Song, Y. Yu, J. Dong, PreCLN: Pretrained-based contrastive learning network for vehicle trajectory prediction, World Wide Web (2022) 1–23.
[47] X. Rao, H. Wang, L. Zhang, J. Li, S. Shang, P. Han, FOGS: First-order gradient supervision with learning-based graph for traffic flow forecasting, in: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, 2022.
[48] X. Han, R. Cheng, C. Ma, T. Grubenmann, DeepTEA: effective and efficient online time-dependent trajectory outlier detection, Proc. VLDB Endow. 15 (7) (2022) 1493–1505.
[49] Y. Liu, K. Zhao, G. Cong, Z. Bao, Online anomalous trajectory detection with deep generative sequence modeling, in: 2020 IEEE 36th International Conference on Data Engineering, ICDE, IEEE, 2020, pp. 949–960.
[50] F. Li, Z. Gui, Z. Zhang, D. Peng, S. Tian, K. Yuan, Y. Sun, H. Wu, J. Gong, Y. Lei, A hierarchical temporal attention-based LSTM encoder-decoder model for individual mobility prediction, Neurocomputing 403 (2020) 153–166.
[51] S. Capobianco, L.M. Millefiori, N. Forti, P. Braca, P. Willett, Deep learning methods for vessel trajectory prediction based on recurrent neural networks, IEEE Trans. Aerosp. Electron. Syst. 57 (6) (2021) 4329–4346.
[52] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 565–573.
[53] M. Chen, Y. Zuo, X. Jia, Y. Liu, X. Yu, K. Zheng, CEM: A convolutional embedding model for predicting next locations, IEEE Trans. Intell. Transp. Syst. 22 (6) (2020) 3349–3358.
[54] Y. Luo, Q. Liu, Z. Liu, Stan: Spatio-temporal attention network for next location recommendation, in: Proceedings of the Web Conference 2021, 2021, pp. 2177–2185.
[55] Y. Lin, H. Wan, S. Guo, Y. Lin, 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, No. 5, 2021, pp. 4241–4248.
[56] H. Wan, Y. Lin, S. Guo, Y. Lin, Pre-training time-aware location embeddings from spatial-temporal trajectories, IEEE Trans. Knowl. Data Eng. (2021).
[57] P. Yang, H. Wang, D. Lian, Y. Zhang, L. Qin, W. Zhang, TMN: Trajectory matching networks for predicting similarity, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 1700–1713.
[58] D. Yao, G. Cong, C. Zhang, J. Bi, Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach, in: 2019 IEEE 35th International Conference on Data Engineering, ICDE, IEEE, 2019, pp. 1358–1369.
[59] X. Liu, X. Tan, Y. Guo, Y. Chen, Z. Zhang, Cstrm: Contrastive self-supervised trajectory representation model for trajectory similarity computation, Comput. Commun. 185 (2022) 159–167.
[60] L. Deng, Y. Zhao, Z. Fu, H. Sun, S. Liu, K. Zheng, Efficient trajectory similarity computation with contrastive learning, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 365–374.
[61] A.R. Butz, Alternative algorithm for Hilbert's space-filling curve, IEEE Trans. Comput. 100 (4) (1971) 424–426.
[62] F. Hausdorff, Grundzüge der mengenlehre, Vol. 7, von Veit, 1914.
[63] H. Alt, M. Godau, Computing the Fréchet distance between two polygonal curves, Internat. J. Comput. Geom. Appl. 5 (01n02) (1995) 75–91.
[64] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD Workshop, Vol. 10, No. 16, Seattle, WA, USA, 1994, pp. 359–370.
[65] M. Vlachos, G. Kollios, D. Gunopulos, Discovering similar multidimensional trajectories, in: Proceedings 18th International Conference on Data Engineering, IEEE, 2002, pp. 673–684.
[66] L. Chen, R. Ng, On the marriage of lp-norms and edit distance, in: Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30, 2004, pp. 792–803.
[67] L. Chen, M.T. Özsu, V. Oria, Robust and fast similarity search for moving object trajectories, in: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, 2005, pp. 491–502.
[68] D. Kong, F. Wu, HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction, in: IJCAI, Vol. 18, No. 7, 2018, pp. 2341–2347.

Shuzhe Li graduated from Shandong University of Science and Technology in 2021 with a bachelor degree in network engineering. He is currently a master student at the College of Computer Science and Technology, Ocean University of China. His research interests include large-scale spatio-temporal data mining and deep learning.

Wei Chen received the bachelor degree in computer science from the Jinggangshan University, China, in 2020. He is currently a master student with the College of Computer Science and Technology, Ocean University of China. His research interests include data mining, information retrieval and causal learning, especially in spatio-temporal field.

Bingqi Yan received his bachelor degrees from theYantai University, China, in 2020. He is currently pursuing the master degree with the College of Computer Science and Technology, Ocean University of China. His research interests include multivariate sequence data mining and spatio-temporal trajectory prediction.

Zhen Li received the bachelor degree in computer science from the Ocean University of China, China, in 2020. He is currently a master student with the College of Computer Science and Technology, Ocean University of China. His research interests include spatio-temporal data mining and machine learning.

Shunzhi Zhu received the Ph.D. degree from Xiamen University, in 2007. He was a visiting professor with Florida International University, USA. He is currently a full professor with the School of Computer and Information Engineering, Xiamen University of Technology. His research interests include big data management and analytics, data mining, and information retrieval.

Yanwei Yu received the Ph.D. degree in computer science from the University of Science and Technology Beijing, China, in 2014. From 2016 to 2018, he was a postdoctoral researcher with the College of Information Sciences and Technology, Pennsylvania State University. He is currently a professor with the College of Computer Science and Technology, Ocean University of China. His research interests include data mining, machine learning, and database systems.