International Journal of Geo-Information

MDPI

Article

Learning Universal Trajectory Representation via a Siamese Geography-Aware Transformer

Chenhao Wu [1,2], Longgang Xiang [3,*], Libiao Chen [1], Qingcen Zhong [3] and Xiongwei Wu [1,2]

[1] Fujian Expressway Science & Technology Innovation Research Institute Co., Ltd., Fuzhou 350000, China; ngch@whu.edu.cn (C.W.)
[2] College of Civil Engineering, Fuzhou University, Fuzhou 350000, China
[3] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, Wuhan 430079, China
* Correspondence: geoxlg@whu.edu.cn

Abstract: With the development of location-based services and data collection equipment, the volume of trajectory data has been growing at a phenomenal rate. Raw trajectory data come in the form of sequences of "coordinate-time-attribute" triplets, which require complicated manual processing before they can be used in data mining algorithms. Current works have started to explore the emerging deep representation learning method, which maps trajectory sequences to vector space and applies them to various downstream applications for boosting accuracy and efficiency. In this work, we propose a universal trajectory representation learning method based on a Siamese geography-aware transformer (TRT for short). Specifically, we first propose a geography-aware encoder to model geographical information of trajectory points. Then, we apply a transformer encoder to embed trajectory sequences and use a Siamese network to facilitate representation learning. Furthermore, a joint training strategy is designed for TRT. One of the training objectives is to predict the masked trajectory point, which makes the trajectory representation robust to low sampling rates and noises. The other is to distinguish the difference between trajectories by means of contrastive learning, which makes the trajectory representation more uniformly distributed over the hypersphere. Last, we design a benchmark containing four typical traffic-related tasks to evaluate the performance of TRT. Comprehensive experiments demonstrate that TRT consistently outperforms the state-of-the-art baselines across all tasks.

Keywords: trajectory representation; geography encoding; transformer; siamese network

Citation: Wu, C.; Xiang, L.; Chen, L.; Zhong, Q.; Wu, X. Learning Universal Trajectory Representation via a Siamese Geography-Aware Transformer. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 64. https://doi.org/10.3390/ijgi13030064

Academic Editors: Hartwig H. Hochmair and Wolfgang Kainz

Received: 21 November 2023
Revised: 2 February 2024
Accepted: 12 February 2024
Published: 20 February 2024

1. Introduction

With the innovative development of communication technology and location acquisition equipment, satellites, surveillance systems, and mobile devices are constantly collecting trajectory data from various moving objects. The massive trajectory data, containing rich information and knowledge, can be applied to such fields as location recommendation [1,2], intelligent transportation systems [3–5], and public safety [6,7]. Today, trajectory data mining has become one of the research hotspots in Computer Science and Geographic Information Science. The information mined from trajectory data can not only support the solution of a range of practical problems occurring in industrial production but also provide further insight into human activities.

However, due to the huge volume, the complex structure, and the uneven quality of trajectory data, labor-intensive feature engineering is required before using it for trajectory mining tasks, which results in inefficient algorithms and limited accuracy. In recent years, generic representation learning models inspired by natural language processing (NLP) techniques (e.g., NNLM [8], word2vec [9]) have been applied with great success to tasks related to language translation, semantic analysis, and even image processing. Scholars have started to extend this idea to trajectory mining tasks. The purpose of trajectory

representation learning is to map high-dimensional trajectory data to low-dimensional vector space through deep neural networks. This technique achieves data dimensionality reduction while preserving the feature of trajectory as much as possible and is scalable enough to be applied to multiple downstream tasks.

Current trajectory representation learning methods mainly use Recurrent Neural Network (RNN)-based encoder–decoder architectures to learn vectorized representations of trajectory data. For example, an enlightening work is [10], in which the authors transform trajectories into sequences of grid IDs and train a sequence-to-sequence (seq2seq) model to generate trajectory representations by reconstructing original trajectories from noisy trajectories. Inspired by that work, many later studies have adopted this encoder–decoder approach. For example, Yao et al. [11] trained a seq2seq model for generating the moving features of trajectory. Tedjopurnomo et al. [12] mapped trajectory points to 3D spatial–temporal grids but still used an RNN-based encoder and decoder to learn the representations. Although these existing methods achieve dimensionality reduction of trajectory data, three important problems are still not well addressed. First, the RNN-based encoder–decoder suffers from local-level (or point-level) feature-ignoring problems. The model trained with a reconstructive objective captures only the global feature of the trajectory and fails to model the local dependencies between trajectory points, which may harm the model's performance. Second, using separated and weakly correlated grid IDs to represent GPS coordinates fails to describe the spatial proximity between trajectory points. Since the representations of trajectories with similar GPS sampling points should be close in vector space, it is important to learn similar representations of grids that are spatially close. Finally, these methods are only applied to specific tasks and have not been validated in additional downstream tasks, so their scalability is still unclear.

To address these issues, we propose a universal trajectory representation learning method based on a Siamese geography-aware transformer (TRT). Figure 1 shows the overview of our approach. First, we adopt the Siamese architecture to train a transformer with two learning objectives. The transformer is effective owing to the self-attention mechanism, which allows the model to capture dependencies between trajectory points. Furthermore, we train the model by predicting randomly masked points in the input trajectory. In this way, the trajectory representation can be robust to low sampling and noise. Another training objective is to learn trajectory representations by distinguishing trajectory pairs. We adopt a Siamese network to simultaneously embed two trajectory samples, which can be positive or negative, and optimize the network via contrastive loss. This strategy enables the model to perceive global-level (or trajectory-level) features, making learned trajectory representations uniformly distributed over the hypersphere.

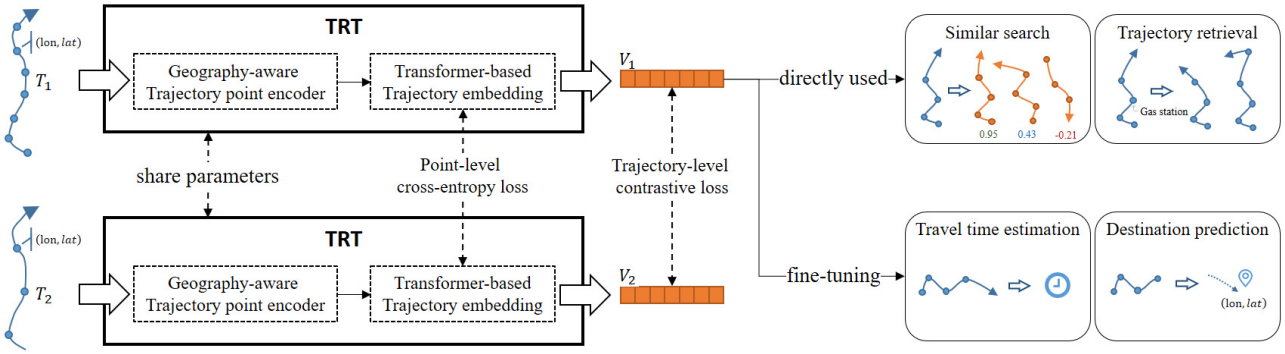


Figure 1. Overview of TRT model.

To evaluate the performance of the proposed TRT, we design two categories of downstream tasks: first, we directly calculate the distance (i.e., cosine distance) between vectorized representations for trajectory similarity measurement and trajectory retrieval; second, we train two very simple networks, using trajectory representations as input, for travel

time estimation and destination prediction. Comprehensive experiments prove that our TRT supports various applications and achieves promising performance.

The contributions of this work can be summarized as follows:

1. We propose a trajectory representation transformer (TRT) to capture both local and global features from trajectories and to make effective use of geographical information. The learned trajectory representations are universal and can be applied to multiple traffic-related tasks;
2. We propose a novel geography encoder to embed the GPS sampling points, such that the spatial proximity between points can be fully modeled. In this way, the geographical information contained in trajectories can be clearly learned;
3. It is the first to apply a Siamese transformer to trajectory representation learning for generating both alignment and uniformity representation. We propose a joint training strategy to train the proposed model, such that the representations of similar trajectories are drawn closer together, and dissimilar trajectories are easily distinguished;
4. We evaluate TRT with two real-world datasets and four traffic-related downstream tasks. Experiment results show that our TRT outperforms state-of-the-art models, demonstrating the effectiveness of incorporating geographical information and the scalability of trajectory representation learning methods.

2. Related Work

We briefly review recent studies of representation models for sequence data, representation models for trajectory data, and universal representation learning.

2.1. Learning Representations from Sequence Data

Sequence data are the data points that are organized in a specified order, such that earlier data points provide information about later data points. Text data, trajectory data, DNA and protein sequences, and time-series observations are typical sequence data. Recently, inspired by the success of NLP, the idea of learning representations from sequence data has been widely applied. The most commonly used is the Recurrent Neural Network-based encoder–decoder model [13], such as sequence-to-sequence learning [14,15]. The encoder embeds an input sequence to a low-dimension vector, and the decoder reconstructs the original sequence by giving the vector. The learned vector can be applied in specific mining algorithms as a representation of high-dimensional data. Another promising method for sequence modeling is the transformer [16], whose self-attention mechanism has the capacity to capture long-range dependencies. For example, Gao et al. [17] propose the simCSE for generating sentence embedding, which takes BERT [18], a variant of the transformer, as the encoder and applies the contrastive learning framework. Different from the above representation learning models for NLP, which takes discrete tokens (i.e., words) as input, our task requires processing trajectory points with spatial proximity relation. To this end, we propose a geography-aware encoder for embedding trajectory points and modify the loss function for perceiving spatial proximity.

2.2. Learning Representations from Trajectory Data

Some works have attempted to learn task-specific trajectory representations by means of word2vec and encoder–decoder models. An enlightening work is t2vec [10], which trains a seq2seq model by reconstructing the original trajectory from its low sampling rate trajectory, making the learned trajectory representations robust to noise and non-uniform sampling rates. Yao et al. [11] use a seq2seq model to generate representations of moving behavior sequences and address the spatio-temporal shifts problem occurring in trajectory clustering. Different from these reconstructive models, which capture only global-level features of trajectory, our work employs the transformer encoder and designs a masked point prediction task to capture local-level features between trajectory points.

Learning trajectory representations aim to improve the effectiveness of tasks such as destination prediction, travel time estimation, etc. Most of the traditional methods

directly process the coordinates of trajectory points, which is inefficient and ineffective. The emerging approaches tend to implement trajectory embedding. For example, Cao et al. [19] combined trajectory data and POI information to recognize human living patterns. Crivellari et al. [20] built a skip-gram word2vec model to construct location embeddings and then obtain trajectory representation by averaging all location embeddings in a trajectory. Fang et al. [21] proposed an end-to-end trajectory clustering framework to embed trajectory and perform trajectory clustering simultaneously. Liao et al. [22] proposed two GPS embedding methods and an attention-based BiLSTM network for destination prediction. Zhang et al. [23] proposed a model for travel time estimation, DeepTravel. It maps trajectory data to grids and extracts features such as geographic location, average speed, and number of trajectory points within grids. The trajectory representation is learned using an LSTM-based model. Although these works have designed trajectory embedding methods for specific tasks, they are not available for other tasks. Our work focuses on learning universal trajectory representations that can be applied to a variety of downstream tasks.

2.3. Learning Universal Representations

The goal of universal representation is to train a single model that maps raw data into low-dimensional and useful vectors. The learned representations should incorporate latent features and can be effectively applied to various applications [24]. Many scientific research fields have started to investigate universal representation learning, such as face recognition [25], computer vision [26,27], and time series analysis [28]. However, only a few works have explored universal trajectory representation learning. Fu et al. [29] mapped trajectories into road segments based on a map-matching technique and then used a seq2seq model to learn trajectory representations. It is the first time that trajectory representation has been applied to multiple downstream applications. Chen et al. [30] proposed the Toast to learn both road segment representations and route representations and apply them to different transport planning tasks. These works incorporate road networks and other external data, whereas our work focuses on trajectory representation learning and does not include such rich information. Furthermore, these methods fail to consider spatial correlations between trajectory points, whereas our work incorporates a geography encoder to model them. The differences between our method and theirs also lie in the Siamese architecture and the new training strategy. It is possible to combine them to achieve further improvements in multiple traffic-related tasks.

3. Geography-Aware Trajectory Representation Learning

This work aims to learn a low-dimensional, uniformly distributed, and robust representation $V \in \mathbb{R}^d$ (d is the dimension of the vector) for each trajectory, T, where T is composed of a series of GPS sampling points $P = (longitude, latitude)$. During the training process, as shown in Figure 2, we take a randomly masked trajectory as an input sequence; the representation model predicts the masked point at position i based on the context points $P_{1,\cdots,i-1,i+1,\cdots,m}$. The Siamese architecture is also applied here for training the model by distinguishing the difference between anchor and candidate trajectories. The trajectory representation transformer (TRT) is illustrated in Figure 2. Each part will be elaborated in the following sections.

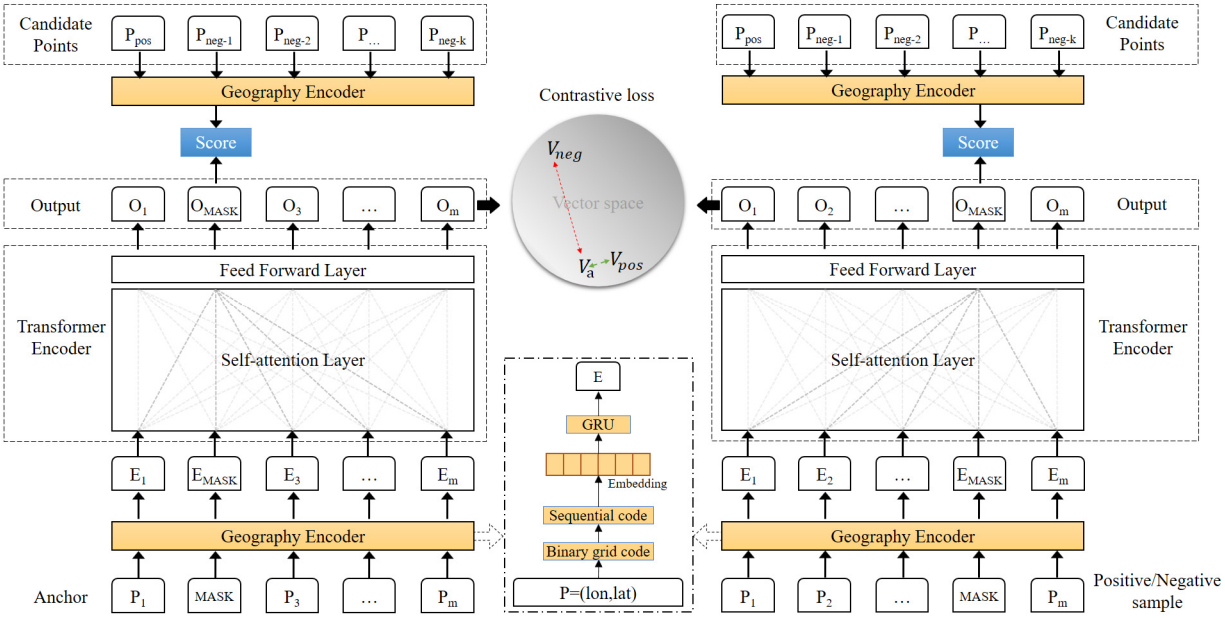


Figure 2. Framework of the proposed TRT.

3.1. Geography-Aware Trajectory Point Encoder

Latitude and longitude are commonly used to define the precise location of each trajectory point, but they are not suitable for direct feeding into the learning system. One reason is that trajectory points cover only a small region of the Earth's surface, directly feeding latitude and longitude into the learning system, which may suffer from the sparsity problem. Furthermore, the exact position of a trajectory point is identified by making joint use of latitude and longitude. If they are fed into the learning system as two discrete values, the interaction between them will be difficult to capture [31]. To better model the geographical information, we propose a geography encoder, which first maps the latitude and longitude of trajectory points into a hierarchical grid and then embeds its sequence code with a gated recurrent unit (GRU) network.

3.1.1. Trajectory Point Mapping

Spatial discretization is widely applied in trajectory feature engineering. Trajectory points are mapped into spatial grids and represented with numeric grid IDs to overcome the sparsity issue. However, the common numeric ID may result in the omission of geospatial proximity between GPS points. To this end, a novel map-gridding scheme is introduced. We first divide the world map into hierarchical grids by using a recurrent four-partition of space. And then, the GPS point located inside the grid can be represented by a binary Morton code [32]. When the number of partitions increases, the grid size decreases, and the code length increases. As illustrated in Figure 3, the Big Ben clock tower (longitude = -0.12454 and latitude = 51.50066) is mapped into a 75 m $\times$ 75 m grid and represented by the code "01111010111010111010110111010110011 1". In this way, the strong interaction between latitude and longitude can be fully modeled during the trajectory point mapping. Furthermore, spatially adjacent grids share the same coding prefix, such that the geospatial proximity between trajectory points can be preserved as much as possible.

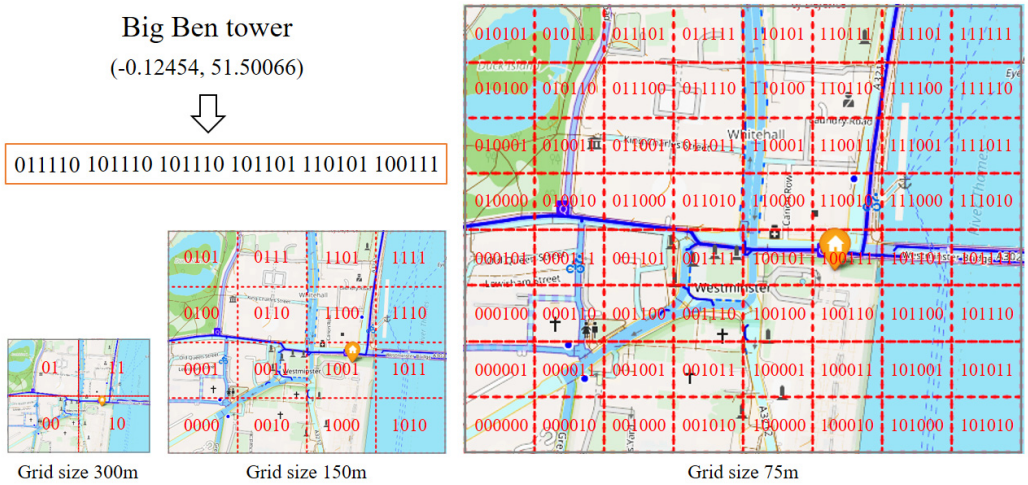


Figure 3. Sketch map of hierarchical map gridding.

3.1.2. Embedding Grid Code

It is intuitive to feed the binary code into a dense layer for linear transformation. However, this method still suffers from the sparsity issue. The reason is that as we keep discretizing the space, the number of grids will increase exponentially, and most of them are without trajectory points. The linear layer maps binary code to continuous space, making all grids predictable, including those without points. To address this problem, we split the binary code into several sub-codes with length l and convert these sub-codes to decimal numbers. Taking the aforementioned binary code "011110101110101110101101110101100111" as an example, the sub-codes with $l = 6$ are (011110, 101110, 101110, 101101, 110101, 100111) $\rightarrow$ (30,39,45,46,53). The binary code is transformed into a character sequence, where each character has a vocabulary of size 2^l. Hence, we can consider the sequence characters as category variables and embed them with an embedding matrix. In this manner, we turn the continuous regression task into a discrete classification task. The learning model will directly ignore grids without trajectory points when embedding, so the sparsity issue is addressed to some extent. The character sequence still preserves the geospatial proximity due to the adjacent grids having the same prefix, so we apply a GRU network to capture the sequential dependency.

3.2. Transformer-Based Trajectory Embedding

Currently, available trajectory representation models are based on the seq2seq approach, where an RNN-based encoder transforms the input trajectory into a vector of specified dimensions, and a decoder reconstructs this vector into the original trajectory. Although this approach achieves dimensionality reduction, the reconstruction strategy makes the learning model focus only on the global feature and ignores the local correlations among trajectory points. Meanwhile, the RNN-based encoder has difficulty in modeling long-range dependencies for long sequences, which may constrain the effectiveness of trajectory representation. To this end, we apply the transformer encoder for embedding the trajectory sequence.

The transformer encoder consists of a self-attention layer and a feed-forward layer. The self-attentive layer takes the sequence tensor $E \in \mathbb{R}^{m \times d}$ (m denotes the length of the sequence) as input and feeds it into an attention module,

$$\mathrm{A} = Attn\left(EW_Q, EW_k, EW_v\right) \tag{1}$$

where W_Q, W_K, and $W_v \in \mathbb{R}^{d \times d}$ denote projection matrices that are used in generating different subspace representations of the query, key, and value matrices. The attention module is the scaled dot-product attention, i.e.,

$$Attn(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{2}$$

The attention module implements self-attention by capturing the relationships between the different elements (in this case, the points) of the same sequence (or trajectory). The output of the self-attention layer will be fed into a feed-forward network layer, which consists of two fully connected layers for encoding intra-dimensional correlations and a ReLU activation function for nonlinear transformations. The formula is written as follows:

$$O_i = ReLU(A_i W_1 + b_1)W_2 + b_2 \tag{3}$$

where $W_1 \in \mathbb{R}^{d \times d_h}$, $W_2 \in \mathbb{R}^{d_h \times d}$, s.t. $d_h > d$. The output $O_i \in \mathbb{R}^d$ denotes the representation vector of the trajectory point at position i. To obtain a fixed-shape trajectory representation, we add a pooling layer to point-wise vectors. In this work, we use three pooling strategies: (1) FIRST: using the first vector in the trajectory sequence. (2) AVERAGE: using the average of all vectors. (3) MAX: using the largest vector in the trajectory sequence. The pooling strategy may affect the performance to some extent, an effect which will be analyzed in the experiments.

3.3. Joint Training Strategy

As mentioned above, this work aims to learn a uniformly distributed and robust representation of trajectories. Ideally, the learned representation shall be robust to non-uniform sampling and noise points. Meanwhile, the learned trajectory representation should be roughly uniformly distributed on the hypersphere, preserving as much information about the original trajectory as possible such that the representation can be applied to various downstream tasks. To meet these two requirements, we design a masked point prediction task and a contrastive representation learning task for jointly training the TRT.

3.3.1. Masked Point Prediction

Sampled trajectories often suffer from uniform sampling frequency, GPS drift, and other problems, which may significantly affect the performance of data mining models. To address this problem, we design a masked trajectory point training task with reference to [16]. Specifically, we generate a noisy trajectory from the original trajectory sequence by randomly masking or distorting the trajectory points. Then, the model is expected to restore these masked points based on the contextual points. However, it is infeasible to directly output the exact position of masked points due to the sparsity issue, so we create a candidate set N and treat the distance of candidate examples as output probabilities. Now, we can train the model by optimizing the cross-entropy loss, which is usually used in sequence modeling [33–35], written as follows:

$$\mathcal{L}_{mask_1} = -\sum_{i=1}^{n}\left(log\sigma(y_{i,O_i}) + \sum_{j=1}^{k} log\left(1 - \sigma\left(y_{i,N_{i,j}}\right)\right)\right) \tag{4}$$

where n denotes the number of masked items, and $y_{i,\blacksquare}$ denotes the preference score for a candidate example at position i. Here, we consider the dot product as the preference score. O_i is the target grid, and k is the size of the i-th set $N_{i,\blacksquare}$. In order to efficiently optimize the loss, parameter k should be limited. We will discuss its sensitivity in the experiments.

However, simply adopting this loss function may lead to two problems. The first one is that simple negative samples (e.g., grids far away from the target) contribute less to the gradient of the loss. It is also infeasible to consider top-k nearest grids as negative due to the false negative problem [31]. To this end, we propose to perform negative sampling based on geographical information. Recall that the grid is represented by a sequence code,

where the sequence position represents a hierarchical spatial scale. We randomly sample from the last two scales, such that the sampled grids will neither be too far to contribute to the gradient nor be false negative samples. Hence, we solve the problem of negative sampling, but the other problem looms. The loss function in Equation 4 penalizes the negative grids with equal weight, which is obviously inappropriate. Intuitively, grids close to the target should be more acceptable than grids far from it. Accordingly, we weight each negative grid with its spatial distance to the target grid. To be specific, the loss function is reformulated as follows:

$$\mathcal{L}_{mask_2} = -\sum_{i=1}^{n}\left(log\sigma\left(y_{i,O_i}\right) + \sum_{j=1}^{k} W_j log\left(1-\sigma\left(y_{i,N_{i,j}}\right)\right)\right) \tag{5}$$

where $W_j = \frac{exp\left(-Dist_{i,j}/\theta\right)}{\sum_{j'=1}^{k} exp\left(-Dist_{i,j'}/\theta\right)}$ is the distance-aware weight for each negative grid. $Dist_{i,j}$ denotes the Euclidean distance between the center of grids, and θ is a scale parameter used to adjust the penalty (magnitude of the gradient) for negative grids.

3.3.2. Contrastive Representation Learning

Although the masked point prediction task enables the model to learn local-level correlations between points, it fails to model global-level features from trajectories. The reason is that the representation generated by the transformer encoder suffers from anisotropy [36]. In other words, trajectory representations occupy a narrow cone in the vector space, which means any two of them have a high cosine similarity. This is obviously not appropriate. A good representation should capture the most shared information from positive pairs and remain invariant to the other noise factors [37]. To this end, we propose to adopt a Siamese network to encode two mini-batch trajectory samples simultaneously and optimize it with the contrastive loss, which encourages positive pairs to be similar and push apart negative pairs. One critical question in contrastive learning is how to construct positive pairs. We sample a mini-batch of C trajectory examples and perform twice randomly dropping with a rate from 0.0 to 0.6, resulting in $2C$ data points. In this way, two downsampled trajectories generated from the same original trajectory can be treated as a positive pair. The other $2(C-1)$ trajectories within a mini-batch are treated as negative examples. Let V and V_+ denote the representations of positive pairs. Then, we follow the contrastive framework in [25] and take the infoNCE loss:

$$\mathcal{L}_{contrastive} = -log\frac{e^{sim(V,V_+)/\tau}}{e^{sim(V,V_+)/\tau} + \sum_{i=0}^{2(C-1)} e^{sim(V,V_i)/\tau}} \tag{6}$$

where τ is a scalar temperature hyperparameter, and $sim(V_1, V_2) = V_1{}^{\intercal}V_2/||V_1||\,||V_2||$ denotes the cosine similarity between normalized V_1 and V_2.

4. Experiments

In this section, we evaluate our method using two real-world trajectory datasets on four different tasks: trajectory similarity measure, trajectory retrieval, destination prediction, and travel time estimation. Section 4.1 introduces the datasets we used, the benchmark models, and the parameter settings. The setup and experimental results of the four tasks are given in Sections 4.2–4.5. To further investigate the validity of the proposed model, we present ablation studies and parameter sensitivity analyses in Sections 4.6 and 4.7, respectively.

4.1. Datasets, Baselines, and Settings

4.1.1. Datasets

We use datasets from two cities to evaluate our model: the Porto taxi dataset from a Kaggle competition [38] and the Wuhan taxi dataset. The first dataset, collected in the city of Porto, contains the trajectories of 442 taxis operating over a year. GPS coordinates

are recorded at a fixed interval of 15 s. We remove trajectories with less than 20 sampling points and yield 1 million trajectories. The dataset is divided into training set, validation set, and test set with a ratio of 8:1:1. The second dataset contains complete trajectories collected from 1835 cabs in Wuhan city over 10 days. We split the original trajectories according to the operation status to obtain the on-load (i.e., carrying passenger) trajectories. Other processing is the same as the first one. The details of these two datasets are shown in Table 1.

Table 1. Dataset statistics.

Dataset	Trajectories	Points	Mean Distance (km)	Mean Travel Time (s)
Porto	1,039,605	49,581,741	4.7	735
Wuhan	359,888	8,833,230	7.8	1423

4.1.2. Baselines

To demonstrate the effectiveness of our proposed TRT, we compare it with the following baselines.

LCSS [39] is a traditional but powerful method that is widely adopted for trajectory similarity measures.

Seq2seq [14] and vanilla transformer (vTrans) serve as embedding baseline methods. Both methods use encoder–ecoder architectures, but the former is implemented based on the RNN, and the latter is implemented based on self-attention.

Trembr [29] learns trajectory embedding via road networks for use in a variety of applications. It transforms trajectory as road segment sequences through the map-matching technique and trains two models for modeling trajectory properties and road segment relationships to facilitate trajectory representation learning.

t2vec [10] is a state-of-the-art method of learning trajectory representation for similarity measures. It maps trajectory points to grids and then feeds the grid ID sequences into a seq2seq framework for generating trajectory representation.

MLP-C [40], a simple multi-layer perceptron for destination prediction, won the Taxi Service Trajectory Prediction Challenge@ ECML/PKDD 2015 [35]. This method uses the first and last five trajectory points as input and considers the weighted center of destination clusters as the predicted destination.

DeepTTE [40] is a well-known baseline for travel time estimation (TTE). This method uses GPS trajectory and external factors to learn spatial and temporal feature representations.

4.1.3. Settings

In our method, we use 6-character sequence codes to represent spatial grids. The grid size is about 75 m. The dropping and distorting rate of input trajectories is set from 0.0 to 0.6. In the geography encoder, both the hidden layer size of GRU network and the output trajectory point representation are set to 128. To make a fair comparison, the default hidden layer size is also set to 128 for all methods. Two scale parameters θ and τ in loss functions are set to 0.1 and 0.05. During the training process, we use the Adam optimizer with a learning rate of 1×10^{-3}, and the number of training epochs is set to 30 for the Porto dataset and 20 for the Wuhan dataset. All models are implemented on a machine with two GTX 2080Ti GPUs and an Intel i7 7700K CPU using the open-source framework PyTorch 1.8.1.

4.2. Trajectory Similarity Measure

Similarity computation is fundamental and essential in many trajectory data mining tasks. In this section, we first introduce the experimental setup, including the dataset preparation and the evaluation method, and then analyze the experimental results.

4.2.1. Setup

Due to the lack of ground truth labels for evaluating the performance of models, we follow the evaluation methodology in [41], which has been widely adopted in related works. Specifically, we randomly selected 100,000 trajectories from the test set, denoted as D. For each trajectory $T \in D$, we orderly extract the points from it in odd and even positions, yielding two sub-trajectories, denoted as T_a and T_b, such that we can construct two the dataset $D_a = \{T_a\}$ and $D_b = \{T_b\}$. Next, we treat the first 10,000 trajectories from D_a as query dataset, denoted as Q_a. For each query $T_a \in Q_a$, we measure its distance to trajectories in D_b using different methods and calculate the rank of T_b. Ideally, trajectory T_b should be ranked first due to T_a and T_b being generated from the same trajectory. Finally, we calculate the mean rank of 10,000 queries in Q_a as the result of similarity computation. Note that we use the cosine distance to measure the similarity between trajectory embeddings.

4.2.2. Evaluation

Table 2 shows the mean rank of query trajectories using different methods when using Porto and Wuhan datasets. As the dropping rate increases, the performance of all methods degrades. This is a common phenomenon, as trajectories with low sampling rates contain much less information than original trajectories. LCSS performs the worst since it is too sensitive to noise. vTrans performs slightly better than seq2seq, as the attention-based model is more capable of capturing features compared to the RNN-based model. However, the sparsity issue (we have discussed in Section 3.1) of directly embedding GPS coordinates restricts the performance of both models. t2vec and Trembr show significant improvements over other methods, demonstrating that transforming raw trajectories into grid (or road segment) sequences is useful. Our TRT performs the best among all baselines.

Table 2. Results for trajectory similarity research.

	Porto					**Wuhan**			
D_{rate}	0	0.2	0.4	0.6	D_{rate}	0	0.2	0.4	0.6
LCSS	113.46	150.58	181.24	192.17	LCSS	162.31	188.85	241.56	380.19
seq2seq	87.1	98.33	117.94	163.5	seq2seq	113.1	141.83	196.97	305.8
vTrans	51.28	60.14	86.63	127.98	vTrans	99.31	128.51	174.79	253.86
Trembr	6.21	8.23	18.07	25.33	Trembr	41.75	52.31	67.24	99.33
t2vec	6.17	9.28	15.59	25.02	t2vec	44.32	49.22	70.03	96.81
TRT	3.92	5.61	8.4	13.45	TRT	15.8	19.53	36.24	56

4.3. Trajectory Retrieval

Trajectory retrieval, a branch of trajectory similarity computation, is a typical task in the field of data mining and analysis. For example, during the pandemic, health workers use a partial trajectory of virus carriers to retrieve trajectories of other moving individuals for estimating the spread of virus. In this section, we introduce the preparation of datasets and the evaluation metric and then compare the performance of TRT with other methods.

4.3.1. Setup

In this task, we first choose 1000 short trajectories, with a length between 20 and 30, as the query and 10,000 other trajectories as the database. Using short trajectories for retrieval is meaningful since short trajectories record the user's activity in a certain region. To overcome the lack of ground truth, we find top-k most similar trajectories in the database as the ground truth for each query. Next, we randomly drop some points in query trajectories according to the dropping rate. For each down-sampled query, we retrieve its k-nearest neighbors in the database and calculate the Hit Ratio$_{@}k$ (the proportion of true k-nearest neighbors) for evaluation. Similar to the first task, we directly use the trajectory representation to calculate similarity scores between trajectories in the query and the database.

4.3.2. Evaluation

We evaluate all methods with $k = 10,\ 20$ in both Porto and Wuhan datasets. The dropping rate is varied from 0.2 to 0.4. In Table 3, we can observe that the performance of all methods degrades when the dropping rate becomes larger. Conversely, the Hit Ratio grows when we enlarge the number of neighbors to retrieve. LCSS is significantly affected by low sampling rates. Seq2seq and vTrans are robust to low sampling rates due to their representation learning methods, but they fail to retrieve true neighbors. Trembr and t2vec outperform other baselines by a significant margin. Our TRT consistently performs the best, demonstrating our method adapts to limited information (i.e., short trajectory) and low sampling rate.

Table 3. Results for trajectory retrieval.

	Porto					**Wuhan**			
D_{rate}	$HR_{@10}$		$HR_{@20}$		D_{rate}	$HR_{@10}$		$HR_{@20}$	
	0.2	0.4	0.2	0.4		0.2	0.4	0.2	0.4
LCSS	0.34	0.15	0.37	0.16	LCSS	0.28	0.11	0.33	0.12
seq2seq	0.37	0.32	0.39	0.34	seq2seq	0.29	0.22	0.36	0.31
vTrans	0.41	0.38	0.46	0.43	vTrans	0.49	0.31	0.54	0.37
Trembr	0.74	0.66	0.75	0.69	Trembr	0.68	0.59	0.7	0.63
t2vec	0.76	0.65	0.77	0.67	t2vec	0.7	0.61	0.71	0.63
TRT	0.82	0.71	0.86	0.74	TRT	0.75	0.69	0.79	0.72

4.4. Destination Prediction

The prediction of moving object's destination with partial trajectory (previous part of a completed trajectory) is an important yet challenging research issue. In this section, we demonstrate that learned trajectory representations can be used for destination prediction. We introduce the experimental setup and show the experimental results in following subsections.

4.4.1. Setup

In this task, given a trajectory prefix, models need to predict the destination position, which is composed of two scalar values (latitude and longitude). To verify the effectiveness and scalability of trajectory representations, we adopt a simple multi-layer perceptron as a task-specific model and take learned trajectory embeddings as input vectors. However, it is infeasible to directly predict the destination due to the sparsity issue. To this end, we follow the methodology used in [42]. Specifically, we predefine a set C of destination cluster centers and use a softmax layer to calculate the probability p of each cluster. As the output $\hat{l}$ must be a single position, we compute a weighted average of the destination cluster centers:

$$\hat{l} = \sum_{i=1}^{C} p_i(l_i) \tag{7}$$

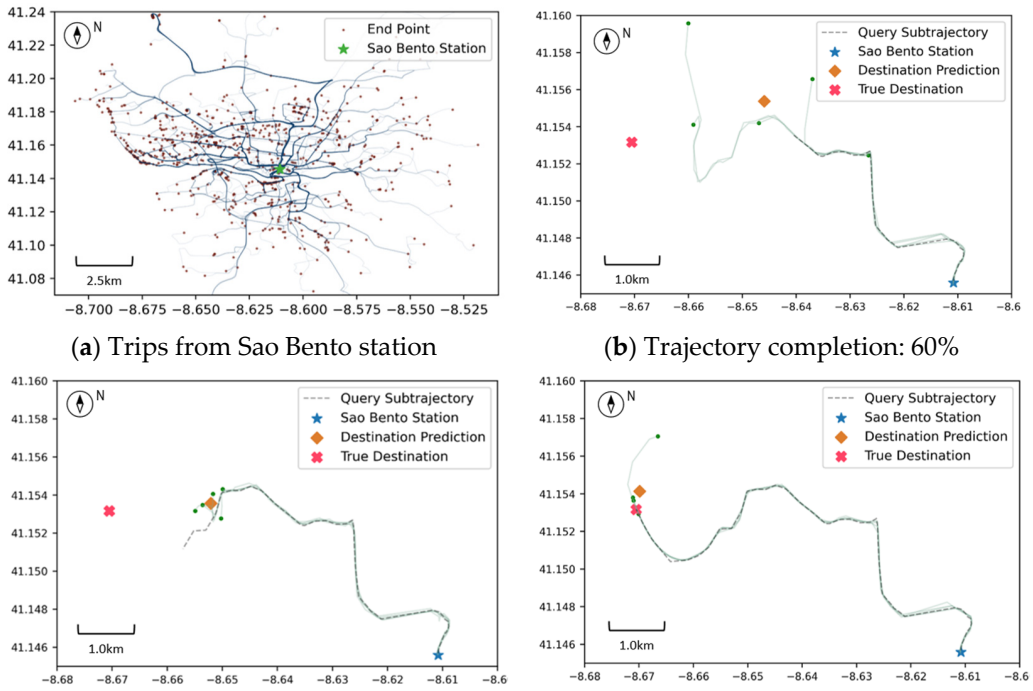
where $p_i = \frac{exp(V_i)}{\sum_{j=1}^{C} exp(V_j)}$.

During the fine-tuning process, we choose 10,000 trajectories from Porto dataset, all of them starting from Sao Bento Station. In the same way, 4000 trajectories starting from Hankou station are selected in Wuhan dataset. We train the aforementioned model, which uses trajectory representations learned by different methods as input, with fivefold cross-validation. For evaluation, we calculate the mean Haversine distance between true destinations and predicted locations as metric.

4.4.2. Evaluation

We compare the prediction results at 40%, 60, 80%, and 99% percentage of trajectory completion. Figure 4 illustrates an example of how our model for final destination prediction works for a trajectory selected randomly among the Sao Bento dataset. We find that the prediction accuracy is proportional to the trajectory completion. This is expected

since the more information we know about a trajectory, the more accurately we can identify which cluster it belongs to. The performance of all methods is summarized in Table 4. The prediction error of seq2seq and vTrans is larger. Trember and t2vec show similar performance. MLP-C, a model specifically designed for destination forecasting, surpasses other baseline methods. Our method achieves competitive performance, demonstrating that the trajectory representation learned by TRT preserves adequate trajectory information. Thanks to contrastive learning, learned representations are uniformly distributed in the vector space. The representations of similar trajectories are close, and the representations of dissimilar trajectories are distant, making it easier for the fine-tuned model to classify them.



(**a**) Trips from Sao Bento station (**b**) Trajectory completion: 60%

(**c**) Trajectory completion: 80% (**d**) Trajectory completion: 99%

Figure 4. Example of destination prediction in Porto.

Table 4. Results for destination prediction.

	Porto					**Wuhan**			
	40%	60%	80%	99%		40%	60%	80%	99%
seq2seq	3.74	2.26	1.98	1.5	seq2seq	6.51	4.63	3.9	2.14
vTrans	3.06	2.57	1.94	1.44	vTrans	6.15	4.33	3.81	2.02
Trembr	2.58	1.84	1.43	1.09	Trembr	4.59	3.37	2.71	1.88
t2vec	2.34	1.76	1.44	1.12	t2vec	4.66	3.51	2.59	2.04
MLP-C	2.05	1.62	0.93	0.61	MLP-C	4.05	3.19	2.07	1.54
TRT	1.97	1.46	0.9	0.58	TRT	3.44	2.67	1.97	1.53

4.5. Travel Time Estimation

Travel time estimation (TTE) is a crucial problem in route planning and traffic dispatching. In this section, we demonstrate the learned trajectory representations can be used for travel time prediction. We introduce the experimental plan and evaluate the performance of various methods in the following subsections.

4.5.1. Setup

We randomly choose 10,000 trajectories from test dataset and calculate the travel time for each trajectory as ground truth. Then, we adopt a simple neural network with two linear

layers for fine-tuning. The fine-tuned network takes trajectory representations learned by different methods as input and predicts the travel time for each trajectory. We train the network with fivefold cross-validation, use the mean absolute error (in seconds) between ground truths, and predict results as the metric of travel time estimation.

4.5.2. Evaluation

We compare our TRT to various methods on Porto and Wuhan datasets. Similar to the experiment of trajectory similarity measure, we vary the dropping rate to investigate the impact of low sampling rates on model performance. We even remove all trajectory points except the origin and the destination because it is more challenging to predict travel time by using the origin–destination (OD) approach. The experimental results are shown in Table 5. The fine-tuned network trained with random vectors serves as a baseline. Performance comparisons between four benchmark models are similar to those in the destination prediction task. DeepTTE achieves the highest accuracy when using complete trajectories. However, this method requires using a specific path (high sampling rate trajectory) to predict. When we remove points from the trajectory, the travel path becomes non-unique, so DeepTTE can no longer make predictions. In contrast, our TRT is robust to low-frequency sampled trajectories, including OD trajectories. One more thing worth noting is that we did not add any external factors, such as weather conditions, temporal trends (time of day, day of week), etc., when training TRT, which are necessary information for TTE-specific methods.

Table 5. Results for travel time estimation.

	Porto					**Wuhan**			
D_{rate}	0	0.3	0.6	OD	D_{rate}	0	0.3	0.6	OD
Random	170.68	-	-	-	Random	476.35	-	-	-
seq2seq	138.03	141.85	158.05	168.79	seq2seq	386.42	393.57	418.79	434.49
vTrans	127.36	130.66	150.17	163.24	vTrans	351.29	363.44	392.03	433.61
Trembr	98.12	98.97	109.01	123.43	Trembr	266.67	268.31	304.62	310.05
t2vec	97.86	98.87	107.8	124.55	t2vec	272.79	274.51	298.02	311.96
DeepTTE	84.29	-	-	-	DeepTTE	220.86	-	-	-
TRT	86.5	91.83	95.25	120.74	TRT	223.1	227.53	234.01	245.49

4.6. Ablation Study

We conduct an ablation study by independently removing different components of TRT to explore their impact on model performance. Model variants are as follows: (1) TRT_M: the model that removes the contrastive learning and is trained only on masked point prediction; (2) TRT_C: the model that is trained only on contrastive objective; (3) w/o GE: the model that replaces the geography encoder by common grid mapping. (4) w/o L_2: the model that does not use geography-aware loss function and applies the original cross-entropy loss function.

We also use two metrics, alignment and uniformity [37], to measure the quality of trajectory representations. Alignment calculates the expected distance between embeddings of positive pairs, which is written as follows:

$$\ell_{align} \triangleq \mathbb{E}_{(x,x^+)} \left|\left| f(x) - f\left(x^+\right) \right|\right|^2 \tag{8}$$

On the other hand, uniformity measures how uniformly the embeddings are distributed:

$$\ell_{uniform} \triangleq log\left(\mathbb{E}_{(x,y)} e^{-2||f(x)-f(y)||^2}\right) \tag{9}$$

Table 6 shows the results of all model variants on four downstream tasks. We find that missing either geography encoder or geography-aware loss function harms performance, proving the effectiveness of these two components.

Table 6. Ablation study of TRT.

Model	ℓ_{align}	$\ell_{uniform}$	Porto				Wuhan			
			MR	$HR_{@10}$	$DP_{80\%}$	TTE	MR	$HR_{@10}$	$DP_{80\%}$	TTE
t2vec	0.35	−2.48	6.17	0.76	1.44	97.86	44.32	0.70	2.59	272.79
TRT_M	0.02	−1.05	5.80	0.77	1.23	96.61	41.53	0.70	2.24	249.08
TRT_C	0.67	−3.91	165.13	0.35	2.57	139.44	480.07	0.31	3.74	363.67
w/o GE	0.33	−3.63	8.77	0.66	1.83	106.25	50.29	0.62	2.88	299.13
w/o L_2	0.29	−3.54	5.42	0.80	1.09	91.34	40.42	0.72	2.11	235.64
TRT	0.28	−3.83	3.92	0.82	0.90	86.50	15.8	0.75	1.97	223.10

Degradation in TRT_M is also observed. Although its alignment is good, the representation learned by TRT_M is anisotropic, leading to poor performance in downstream tasks. The performance of TRT_C shows significant degradation, demonstrating that using contrast learning alone does not work. Compared to t2vec, our TRT with lower ℓ_{align} and $\ell_{uniform}$ performs better, indicating that both alignment and uniformity are necessary for a good representation.

4.7. Parameter Sensitivity Analysis

In this section, we perform further experiments to investigate how hyper-parameters affect TRT. The following experiments are conducted on Porto and Wuhan datasets. We evaluate the impact of the parameters through the trajectory similarity measure.

4.7.1. Effect of Pooling Strategy and Representation Dimension

The pooling strategy is adopted here for generating fixed-shape representations. A good pooling strategy should collect discriminative information and remove irrelevant details. Figure 5 shows that the average strategy performs better than max and first strategies. One possible reason is that average pooling takes the mean value of all contextual vectors; thus, the noise can be further reduced. Another important parameter that determines the quality of learned representations is the dimension size. From Figure 5, it can be seen that the model performs the worst when the dimension is 32. This is because the dimension size determines how much information the representation contains. When the dimension size is small, information may be lost in the process of representation learning, leading to the degradation of performance. In contrast, a high dimension of representation is typically much more expressive but requires more training data to avoid overfitting. In this experiment, our model performs better when the dimension size is between 128 and 256.

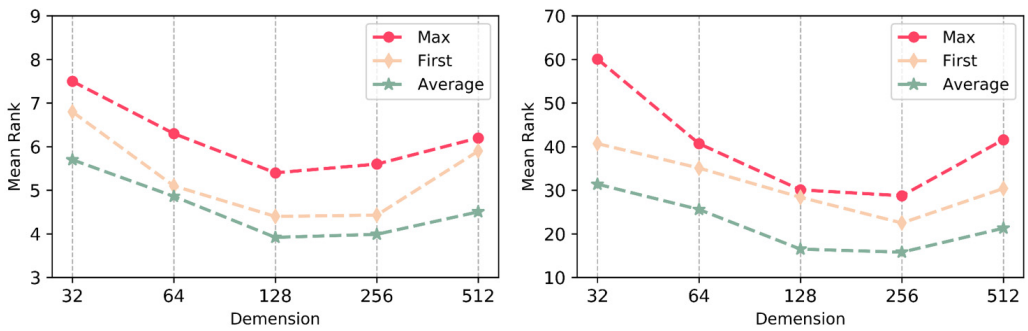


Figure 5. Performance w.r.t. pooling strategy and representation dimension.

4.7.2. Effect of Negative Sampling

Recall the training objective in masked point prediction—we aim to accurately predict masked points by teaching the model to distinguish positive and negative samples. In this experiment, we investigate the effect of different negative sampling methods by varying the number of negative samples from 1 to 20. The results are shown in Figure 6. We find that K-nn sampling performs consistently better than random sampling. This verifies that hard negative samples (nearby grids) contribute more gradient during training. We also discovered that when using K-nn sampling, a small number of negative samples could lead to significant performance improvements. In contrast, random sampling requires more negative samples to improve performance, followed by an exponential increase in memory usage.

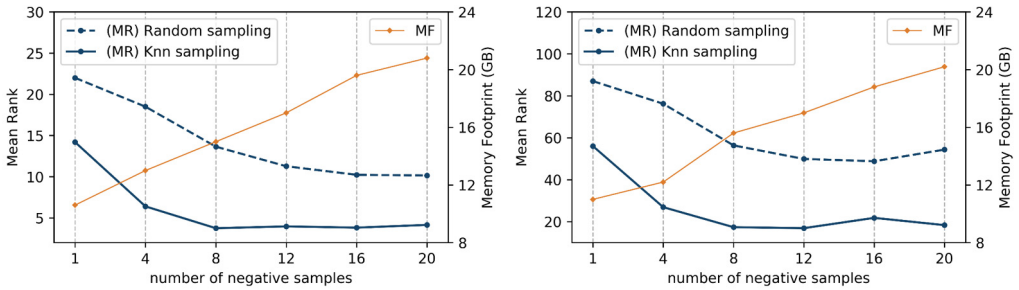


Figure 6. The impact of negative sampling.

5. Conclusions

This study focuses on learning universal trajectory representations that are well-suited for use in multiple traffic-related tasks. Most existing methods are designed for specific tasks and have not been validated for other tasks. To this end, we propose a trajectory representation transformer (TRT), a universal trajectory representation learning framework. First, we propose a novel geography encoder to model the spatial proximity between trajectory points and use a transformer to capture local-level features of the trajectory. Then, we adopt a Siamese network to simultaneously embed positive (or negative) trajectory pairs, which can not only facilitate global-level feature modeling but also address the anisotropy of representations. Moreover, we designed a joint training strategy to train TRT, which consists of a masked point prediction task and a contrastive learning task. Empirical studies show that the proposed TRT framework is able to comprehensively learn the representation of trajectory in two real-world datasets to support various downstream tasks, including trajectory similarity measure, trajectory retrieval, destination prediction, and travel time estimation. An ablation study and parameter sensitivity analysis are also conducted to demonstrate the effectiveness of each component in TRT.

Although this paper provides a new idea for learning universal trajectory representation, there are still some points that deserve continued research in the future. Firstly, a road network plays a crucial role in enhancing trajectory representation by providing a structured context that accounts for spatial relationships, patterns, and constraints. We plan to develop a road network-aware encoder that takes into account the topology of road networks. Furthermore, traffic data contain valuable information about the dynamics and behavior of vehicles, which can be leveraged to enhance trajectory representation learning. By incorporating traffic information, we expect to gain insights into traffic patterns and improve the accuracy of trajectory prediction and analysis tasks.

Author Contributions: Conceptualization, Longgang Xiang; Methodology, Chenhao Wu; Software, Chenhao Wu and Xiongwei Wu; Validation, Qingcen Zhong; Formal analysis, Libiao Chen; Investigation, Qingcen Zhong and Xiongwei Wu; Data curation, Chenhao Wu; Writing—original draft, Chenhao Wu; Writing—review & editing, Longgang Xiang; Visualization, Xiongwei Wu; Supervision, Longgang Xiang; Project administration, Libiao Chen; Funding acquisition, Libiao Chen. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Transport Key Science and Technology Project in Transportation, 2022-ZD6-074; and the National Natural Science Foundation of China, 42071432.

Data Availability Statement: The data that support the findings of this study are openly available in Kaggle, ECML/PKDD 15.

Acknowledgments: We would like to express our sincere thanks to the reviewers and editors for their valuable comments that helped improve this paper.

Conflicts of Interest: Libiao Chen and Xiongwei Wu are working in Fujian Expressway Science & Technology Innovation Research Institute, and declare no conflicts of interest.

References

1. Zheng, Y.; Xie, X. Learning travel recommendations from user-generated GPS traces. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1–29. [CrossRef]
2. Huang, L.; Ma, Y.; Wang, S.; Liu, Y. An Attention-Based Spatiotemporal LSTM Network for Next POI Recommendation. *IEEE Trans. Serv. Comput.* **2021**, *14*, 1585–1597. [CrossRef]
3. Hong, Z.; Chen, Y.; Mahmassani, H.S. Recognizing Network Trip Patterns Using a Spatio-Temporal Vehicle Trajectory Clustering Algorithm. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 2548–2557. [CrossRef]
4. Liu, H.; Jin, C.; Zhou, A. Popular route planning with travel cost estimation from trajectories. *Front. Comput. Sci.* **2020**, *14*, 191–207. [CrossRef]
5. Wu, C.H.; Xiang, L.G.; Yan, J.L.; Zhang, Y.T. Spatio-temporal neural network for taxi demand prediction using multisource urban data. *Trans. GIS* **2022**, *26*, 2166–2187. [CrossRef]
6. Srivastava, S.; Ng, K.K.; Delp, E.J. Co-ordinate mapping and analysis of vehicle trajectory for anomaly detection. In Proceedings of the 2011 IEEE International Conference on Multimedia and Expo, Barcelona, Spain, 11–15 July 2011; pp. 1–6. [CrossRef]
7. Hao, F.; Jin, J. A Broad Learning Ensemble System Using Bagging for Typhoon Trajectory Forecasting. In Proceedings of the 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, 14–16 January 2022; pp. 729–733. [CrossRef]
8. Bengio, Y.; Réjean, D.; Vincent, P. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
9. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
10. Li, X.; Zhao, K.; Cong, G.; Jensen, C.S.; Wei, W. Deep Representation Learning for Trajectory Similarity Computation. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 16–19 April 2018; pp. 617–628. [CrossRef]
11. Yao, D.; Zhang, C.; Zhu, Z.; Huang, J.; Bi, J. Trajectory clustering via deep representation learning. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3880–3887. [CrossRef]
12. Tedjopurnomo, D.A.; Li, X.; Bao, Z.; Cong, G.; Choudhury, F.; Qin, A.K. Similar Trajectory Search with Spatio-Temporal Deep Representation Learning. *ACM Trans. Intell. Syst.* **2021**, *77*, 26. [CrossRef]
13. Hinton, G.; Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef]
14. Cho, K.; Van Merri, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
15. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3104–3112.
16. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.
17. Gao, T.; Yao, X.; Chen, D. SimCSE: Simple Contras-tive Learning of Sentence Embeddings. *arXiv* **2021**, arXiv:2104.08821.
18. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2019**, arXiv:1810.04805.
19. Cao, H.; Xu, F.; Sankaranarayanan, J.; Li, Y.; Samet, H. Habit2vec: Trajectory Semantic Embedding for Living Pattern Recognition in Population. *IEEE Trans. Mob. Comput.* **2020**, *19*, 1096–1108. [CrossRef]
20. Crivellari, A.; Beinat, E. From Motion Activity to Geo-Embeddings: Generating and Exploring Vector Representations of Locations, Traces and Visitors through Large-Scale Mobility Data. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 134. [CrossRef]

21. Fang, Z.; Du, Y.; Chen, L.; Hu, Y.; Gao, Y.; Chen, G. E2DTC: An End to End Deep Trajectory Clustering Framework via Self-Training. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 696–707. [CrossRef]
22. Liao, C.; Chen, C.; Xiang, C.; Huang, H.; Xie, H.; Guo, S. Taxi-Passenger's Destination Prediction via GPS Embedding and Attention-Based BiLSTM Model. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 4460–4473. [CrossRef]
23. Zhang, H.; Wu, H.; Sun, W.; Zheng, B. Deeptravel: A neural network based travel time estimation model with auxiliary supervision. *arXiv* **2018**, arXiv:1802.02147.
24. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
25. Shi, Y.; Yu, X.; Sohn, K.; Chandraker, M.; Jain, A.K. Towards Universal Representation Learning for Deep Face Recognition. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 6816–6825. [CrossRef]
26. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 9726–9735. [CrossRef]
27. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the 37th International Conference on Machine Learning (ICML'20), Virtual, 13–18 July 2020; pp. 1597–1607.
28. Yue, Z.; Wang, Y.; Duan, J.; Yang, T.; Huang, C.; Tong, Y.; Xu, B. TS2Vec: Towards Universal Representation of Time Series. *Proceeding AAAI* **2022**, *2022*, 8980–8987. [CrossRef]
29. Fu, T.; Lee, W. Trembr: Exploring Road Networks for Trajectory Representation Learning. *ACM Trans. Intell. Syst. Technol.* **2020**, *11*, 1–25. [CrossRef]
30. Chen, Y.; Li, X.; Cong, G.; Bao, Z.; Long, C.; Liu, Y.; Chandran, A.K.; Ellison, R. Robust Road Network Representation Learning: When Traffic Patterns Meet Traveling Semantics. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Virtual, 30 October 2021; pp. 211–220. [CrossRef]
31. 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 (KDD '20), New York, NY, USA, 6–10 July 2020. [CrossRef]
32. Morton, G.M. *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*; Tech. Rep.; IBM Ltd.: Ottawa, ON, Canada, 1966.
33. Kang, W.; McAuley, J. Self-Attentive Sequential Recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206. [CrossRef]
34. Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; Kaiser, Ł. Universal Transformers. *arXiv* **2018**, arXiv:1807.03819.
35. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. *arXiv* **2022**, arXiv:2201.12740.
36. Li, B.; Zhou, H.; He, J.; Wang, M.; Yang, Y.; Li, L. On the Sentence Embeddings from Pre-trained Language Models. *arXiv* **2020**, arXiv:2011.05864.
37. Wang, T.; Isola, P. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In Proceeding of International Conference on Machine Learning (ICML), Virtual, 13–18 July 2020. [CrossRef]
38. Kaggle. ECML/PKDD 15. 2015. Available online: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i (accessed on 2 July 2015).
39. Vlachos, M.; Kollios, G.; Gunopulos, D. Discovering similar multidimensional trajectories. In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, 26 February–1 March 2002; pp. 673–684.
40. Wang, D.; Zhang, J.; Cao, W.; Li, J.; Zheng, Y. When will you arrive? Estimating travel time based on deep neural networks. *Proc. AAAI Conf. Artif. Intell.* **2018**, *18*, 1–8. [CrossRef]
41. Ranu, S.; Deepak, P.; Telang, A.D.; Deshpande, P.; Raghavan, S. Indexing and matching trajectories under inconsistent sampling rates. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 999–1010.
42. Brébisson, A.D.; Simon, É.; Auvolat, A.; Vincent, P.; Bengio, Y. Artificial Neural Networks Applied to Taxi Destination Prediction. *arXiv* **2015**, arXiv:1508.00021.