2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00221

T3S: Effective Representation Learning for Trajectory Similarity Computation

Peilun Yang[♮], Hanchen Wang[♮*], Ying Zhang[♮], Lu Qin[♮], Wenjie Zhang[†], Xuemin Lin[†]

[♮]*Centre for Artificial Intelligence, University of Technology Sydney, Australia*

[†]*The University of New South Wales, Australia*

[♮]{peilun.yang, hanchen.wang-1}@student.uts.edu.au; {ying.zhang, lu.qin}@uts.edu.au;

[†]{zhangw, lxue}@cse.unsw.edu.au;

***Abstract*—Advances of the sensor and GPS techniques have motivated the proliferation of trajectory data in a wide spectrum of applications. Trajectory similarity computation is one of the most fundamental problems in trajectory analytics. Considering that the high complexity of similarity computation is usually a bottleneck for large-scale trajectory data analysis, there are many research efforts for reducing the complexity such as the approximate algorithms. However, most of them are proposed for only one or two specific similarity measures, and thus cannot support different similarity measures well. In this paper, we propose a deep learning based model, namely T3S, which embeds each trajectory (i.e., a sequence of points) into a vector (point) in a d-dimensional space, and hence can significantly accelerate the similarity computation between the trajectories. By applying recurrent and attention neural networks, T3S can capture various unique characteristics of the trajectories such as the ordering of the points, spatial and structural information. Furthermore, our learning based T3S can easily handle any trajectory similarity measures by adjusting its parameters through the training. Extensive experiments on two real-life datasets demonstrate the effectiveness and efficiency of T3S. T3S outperforms state-of-the-art deep learning based methods under four popular trajectories similarity measures.**

I. Introduction

Driven by advances of location-acquisition techniques and GPS devices, an increasing number of trajectories have been generated and managed in various application domains, such as traffic planning, animal migration analysis, and anomaly detection. Among these applications, measuring the similarity between trajectories is a fundamental problem for trajectory analysis. As shown in [1], a variety of measures have been proposed in the literature to compute the similarity between trajectories such as the Hausdorff distance [2], the Fréchet distance [3], Dynamic Time Warping (DTW) [4], Edit distance with Real Penalty (ERP) [5], and Longest Common SubSequence (LCSS) [6]. These measures aim to capture the similarity of two trajectories from different perspectives such as the spatial and structural information of the trajectories, and the order of the points in each trajectory. As shown in many existing works (e.g., [5], [7], [8]), every similarity function has its own pros and cons, and users should choose a specific similarity measure according to their application scenarios and requirements.

*Corresponding Author

As the pairwise trajectory similarity computation is the building brick in various trajectory analytic tasks, it is essential to develop efficient algorithms. Though a set of exact algorithms have been proposed in the literature for various similarity metrics (e.g., [2]–[4], [7], [9]), the algorithms still suffer from high time complexity. Particularly, given two trajectories T_i and T_j, we usually need to consider all possible pairs between points in T_i and T_j such that we can find a good match or alignment for each point in the trajectories, leading to a quadratic computational complexity which is time-consuming for large trajectories.

To speed-up the trajectory similarity computation, a variety of approximate algorithms (e.g., [10], [11]) have been proposed in the literature. However, as shown in [12], they are still time-consuming in order to achieve a acceptable accuracy. Moreover, they are designed to specific similarity metrics. Since different similarity measures are usually applied in different application scenarios and new similarity measures may emerge as well, it will be desirable if a method can work under different similarity measures.

In this paper, we propose a deep learning based model, namely T3S, for Trajectory Similarity computation which can effectively consider both Structural and Spatial information. Through this model, each trajectory can be mapped into a point (vector) in a low-dimensional space for a given similarity measure such that the trajectory similarity is well preserved in the embedded space; that is, given two trajectories T_i and T_j and their corresponding vector representation $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ in the embedded space, $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ are close to (resp. far from) each other if T_i and T_j are similar (resp. dissimilar) to each other. There are three immediate advantages for our learning based approaches:

- **Efficiency.** Once each trajectory is represented by a vector (point) in a d-dimensional embedded space, it takes only $O(d)$ time to compute the similarity of two trajectories regardless the size and similarity measure of the trajectories.
- **Generality.** Our proposed model can adapt to any similarity measure by adjusting its parameters through the training. Specifically, we can easily obtain the training data which consists of pairs of similar and dissimilar trajectories based on the trajectories and the given similarity measure.

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00221

- **Easy for indexing.** For similarity query processing (e.g., nearest neighbor search) on trajectories, the state-of-the-art indexing techniques for multi-dimensional data can be immediately applied to their learned embedded vectors in the d-dimensional space.

Motivation. In order to effectively learn the similarity between trajectories under different similarity measures, it is important to capture the intrinsic properties of the trajectories and similarity measures. Existing distance measures take the locations and structures of the trajectories, and the order of points along the trajectories into consideration, because these information plays an important role in trajectory similarity measurement. We employ the LSTM network to process the points which are usually real-valued coordinate tuples along the trajectory to handle the locations of trajectory. In addition, we exploit grid cells to represent trajectories and apply a self-attention based network to learn embedding vectors for the grid-represented trajectories. By carefully combining the LSTM and the self-attention based networks, our model can capture both spatial and structural information of the trajectories. Consequently, our model is able to work well under various distance measures although these methods consider different information. Through a learnable parameter, our model can also automatically adjust the impacts/importance of the structure and spatial information according to the nature of the given similarity measure, and hence can provide better similarity approximation.

Contribution. Our principle contributions can be summarized as follows:

- We propose a deep learning based model called T3S for trajectory similarity computation, which can accommodate different trajectory similarity measures. T3S can preserve both spatial and structure information of the trajectories for the similarity computation by effectively applying LSTM and self-attention based networks. Moreover, our model can automatically adjust the importance of spatial and structure information for different similarity measures.
- Experimental results on two real-life trajectory datasets and four trajectory similarity measures demonstrate that T3S outperforms the state-of-the-art techniques in terms of trajectory similarity computation efficiency and accuracy.

II. Preliminary

Trajectory is kind of structured data with abundant spatial information. In this paper, a trajectory T is represented by a sequence of points denoted by geographical coordinate locations. Following is a formal definition.

Definition 1. *(**Trajectory**). A trajectory T is denoted by a series of points, i.e., $T = [p_1, p_2, ..., p_i, ..., p_l]$ where $p_i = (lon_i, lat_i)$ is i-th point and l is the length of the trajectory.*

In this paper, we use $d(p, q)$ to denote the Euclidean distance between two points p and q. Following is a general definition of the trajectory similarity.

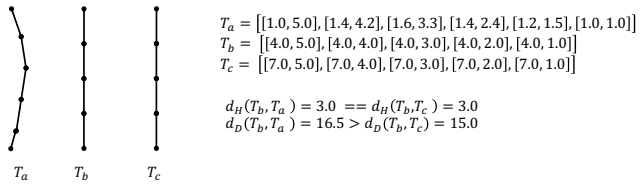

Fig. 1. An example of trajectory similarity under different distance measures.

Definition 2. *(**Trajectory Similarity**). Given two trajectories T_i and T_j, a distance measure $d_M(T_i, T_j)$ is able to measure the similarity between T_i and T_j. T_i and T_j are similar (resp. dissimilar) if their distance is small (resp. large).*

A variety of trajectory similarity measures have been proposed in the literature (See [1] for a comprehensive survey). Though our model can fit any trajectory similarity measure, same as [12], we use the following four popular similarity measures for performance evaluation and examples in this paper: Hausdorff distance, Fréchet distance, DTW distance and ERP distance, which are denoted by $d_H(.)$, $d_F(.)$, $d_D(.)$, and $d_E(.)$ in this paper respectively.

Problem Statement. Given a trajectory similarity measure $d_M(.,.)$, our target is to learn a trajectory similarity approximation function $f(.,.)$ such that for any pair of trajectories T_i and T_j, their difference $|f(T_i, T_j) - d_M(T_i, T_j)|$ is minimized.

III. Our approach

In this section, we give an overview of our approach and then introduce the details of T3S.

A. Overview

Trajectory similarity aims to find the correlations between trajectories. As we mentioned in Section II, trajectories are represented by coordinate tuples following the time order. Usually, the coordinate tuples contain location information. To learn the similarity between trajectories, a model should not only preserve the overall geographical distance between the trajectories, but also capture the structural information of the trajectories, especially find the deterministic nodes for similarity computation under different distance metrics. Fig. 1 shows an instance of similarity computation between trajectories under different distance measures. Given three trajectories T_a, T_b and T_c. Intuitively, T_b is more similar with T_c than T_a because T_b and T_c are both straight lines, while T_a is a polyline. If Hausdorff distance is applied to compute the similarity, according to the distance definition, $d_H(T_b, T_a) = 3.0$ and $d_H(T_b, T_c) = 3.0$. If DTW distance is used, then $d_D(T_b, T_a) = 16.5$ and $d_D(T_b, T_c) = 15.0$. In this example, the distances $d(T_b, T_a)$ and $d(T_b, T_c)$ can be distinguished under DTW metric, while they are exactly the same under Hausdorff metric. DTW compares all point pairs between trajectories and accumulates the discrepancies of matched point pairs following the sequential order of trajectory. Hausdorff distance aims to find a maximum of a set of distances between the closest point pairs to represent the distance between two trajectories. Therefore, DTW distance

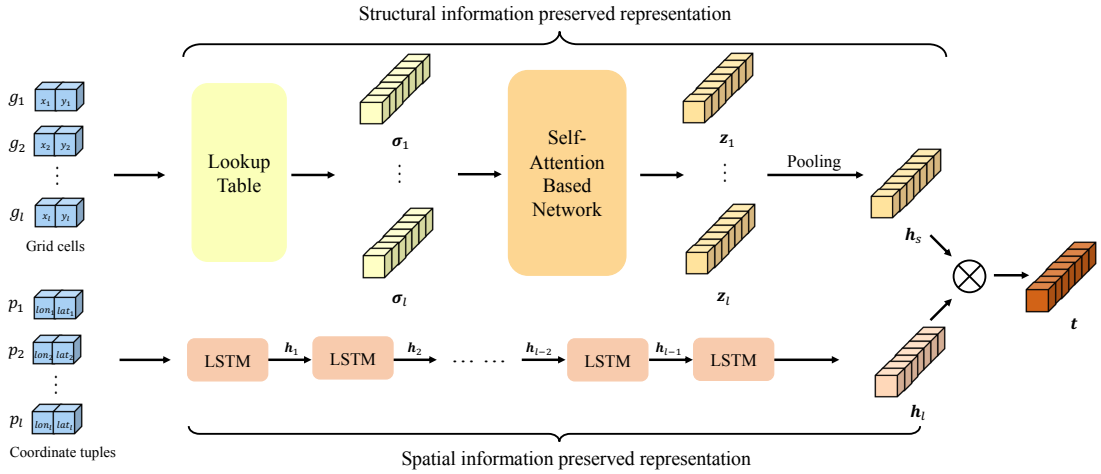

Fig. 2. The framework of T3S. T3S is composed of two parts, *i.e.,* a structural information-preserved network and a spatial information-preserved network. These networks are adjustably combined in T3S.

distinguishes trajectories of different shapes in this case, while Hausdorff distance mainly focuses on the spatial information which is only determined by the most dissimilar point pair in the trajectories. To generalize the learning-based model to approximate trajectory similarities under different distance measures, T3S is proposed, which can flexibly adjust the proportion of spatial information and structural information.

Fig. 2 shows the architecture of T3S. As shown, our model is composed of two parts, *i.e.,* a self-attention based network and LSTM network. T3S exploits self-attention based network to process the grid represented trajectories, thus captures the structural information. Meanwhile, the coordinates series, which contains the spatial information, is encoded to a low-dimensional representation by the LSTM network. These networks are adjustably applied in T3S to embed a trajectory into a low-dimensional vector, which can be applied to downstream tasks.

B. Input Representation

Trajectories in the real world are often represented by sequences of geographical locations. Similarity measures employ these locations to compute trajectory similarity, so that it can obtain the spatial information between trajectories. In this paper, each location is represented by a coordinate tuple, as we mentioned in Section II. T3S exploits the coordinate tuples of trajectories when learning representations for them.

Following the previous studies [13], [14], we evenly partition the entire geographical space into discrete grid cells and map the coordinate tuples to the grid cells. Concretely speaking, we regard the whole area as a large rectangle. All the trajectories belong to this area. The entire area is then divided into grid cells with same size. Formally, assume the entire area is divided into a matrix of $m \times n$, that is, the whole area is mapped into $m \times n$ grid cells. We build a function $\mathcal{G} : p_i \rightarrow (x_i, y_i)$ where p_i is the coordinate tuple made up of longitude lon_i and latitude lat_i, x_i and y_i are the indices along the longitude and the latitude. Then we can map the geographical coordinate tuples to the indices of the grid cells. Trajectory $T^c = [p_1, p_2, ..., p_i, ..., p_l]$ can be mapped into a grid sequence $T^g = [g_1, g_2, ..., g_i, ..., g_l]$, where $g_i = (x_i, y_i)$. We now can not only exploit the geographical coordinate tuples in our model, but also employ the indices of grid cells. In next subsections, we introduce how T3S learns representations for trajectories with both coordinate tuples and grid representations.

C. Structural information preserved representation

To capture the structural information of trajectories, we pay attention to the structure of trajectories. Therefore, our model should capture the importance of each point in the trajectory. Inspired by [15], which is mainly based on the attention mechanism, we develop a self-attention based network to capture the structural information of trajectories and encode the trajectories to low-dimensional representations. Fig. 3 shows the architecture of our self-attention based network.

Given a trajectory T with grid representations $T^g = [g_1, g_2, ..., g_i, ..., g_l]$ where $g_i = (x_i, y_i)$. The input embedding $\boldsymbol{m}_i$ is formulated using the following equations,

$$\boldsymbol{\sigma}_i = lookup(x_i, y_i) \tag{1}$$

$$\boldsymbol{m}_i = \boldsymbol{\sigma}_i + \boldsymbol{pos}_i \tag{2}$$

where $\boldsymbol{m}_i, \boldsymbol{\sigma}_i, \boldsymbol{pos}_i \in \mathbb{R}^{d \times 1}$ and d is the dimension of hidden vector.

We initialize the embedding $\boldsymbol{\sigma}_i \in \mathbb{R}^d$ for every grid following the normal distribution. The grid embedding matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n \times d}$ is stored in a lookup table for further retrieval. To record the positional information of elements in the sequence, we combine each grid embedding $\boldsymbol{\sigma}_i$ with a positional encoding vector $\boldsymbol{pos}_i$ to obtain the input embedding $\boldsymbol{m}_i$. Positional encoding vectors follow a learned or pre-defined pattern, *e.g.,* cosine positional function and sine positional function, which determines the position of each element. The formulated vectors are then fed to the following layers.

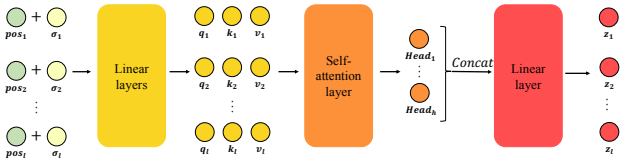

Fig. 3. Self-attention based network. It employs multi-head mechanism when computing attention scores for the input sequence.

After obtaining the input embeddings, the procedures are summarized in following equations,

$$(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{M} \times (\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V) \tag{3}$$

$$\boldsymbol{A} = Softmax(\frac{\boldsymbol{Q} \times \boldsymbol{K}^T}{\sqrt{d}}) \tag{4}$$

$$\boldsymbol{Z} = \boldsymbol{A} \times \boldsymbol{V} \tag{5}$$

where $\boldsymbol{M} \in \mathbb{R}^{b \times l \times d}$ is the input matrix, $\boldsymbol{A} \in \mathbb{R}^{l \times l}$, $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{b \times l \times d}$ are the query, key and value matrices. $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V \in \mathbb{R}^{d \times d}$ represents the weight matrix to produce $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$. b is the batch size, $\boldsymbol{Z} \in \mathbb{R}^{l \times d}$ is the representation of the grids after calculating the self-attention.

Attention network first obtains the query, key and value vectors $\boldsymbol{q}_i$, $\boldsymbol{k}_i$ and $\boldsymbol{v}_i$ by multiplying each input $\boldsymbol{m}_i$ with the corresponding weight matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$ and $\boldsymbol{W}^V$. For grid cell at position i, we calculate the attention scores by taking the dot product of i-th grid's query vector $\boldsymbol{q}_i$ with the key matrix $\boldsymbol{K}$ which contains the key vectors for all grids in the trajectory. Then these scores are passed through a softmax layer, which normalizes the scores so that the sum of these scores equals to 1. The normalized attention score $a_{i,j}$ measures the relevance between i-th grid and j-th grid. A high $a_{i,j}$ score indicates that these two grids may have tight connection or j-th grid may have a obvious impact on similarity computation. Next, we multiply value matrix $\boldsymbol{V}$ with the computed attention score matrix $\boldsymbol{A}$ to aggregate the neighboring representations for all grids in the trajectory. Finally, we obtain a weighted value matrix as the representation of grids, denoted by $\boldsymbol{Z}$.

We also employ the multi-head mechanism to further improve the robustness and generalization of T3S. Instead of computing attention results with d-dimensional queries, keys and values only once as above equations, the multi-head mechanism linearly projects the queries, keys and values multi-times. Each of projected versions of queries, keys and values can be used to compute attention scores for every grid in the trajectory. In this paper, we calculate queries, keys and values h times and combine the results of different projected versions to get the representations of grids by the following equations.

$$(\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i) = \boldsymbol{M} \times (\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V) \tag{6}$$

$$\boldsymbol{Head}_i = Softmax(\frac{\boldsymbol{Q}_i \times \boldsymbol{K}_i^T}{\sqrt{d_k}}) \times \boldsymbol{V}_i \tag{7}$$

$$\boldsymbol{Z} = Concat(\boldsymbol{Head}_1, \boldsymbol{Head}_2, ..., \boldsymbol{Head}_h)\boldsymbol{W}^O \tag{8}$$

where $\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i \in \mathbb{R}^{b \times l \times d_k}$ are query, key and value matrices of i-th attention head and $i \in [1, 2, ..., h]$. d_k is the size of each part's hidden size where $d_k = d/h$, $\boldsymbol{Head}_i$ is the result of i-th attention head, $Concat$ is the concatenation operation, $\boldsymbol{W}^O \in \mathbb{R}^{d \times d}$ is a linear layer to process the concatenated result.

In order to learn the representation for a trajectory from the grid embeddings, we employ mean pooling to process the representations of grid cells. The row vector $\boldsymbol{z}_i$ in the outputs $\boldsymbol{Z}$ is the encoding result of i-th input grid, the mean value of these row vectors is obtained as a new d-dimensional vector $\boldsymbol{h}_s$ to represent the entire trajectory.

$$\boldsymbol{h}_s = mean(\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_l) \tag{9}$$

where $\boldsymbol{z}_i$ is the i-th row vector of the output matrix $\boldsymbol{Z}$. Finally, the trajectory representation $\boldsymbol{h}_s$ that preserves the structural information is obtained.

D. Spatial information preserved representation

Since trajectories can be viewed as geometric curves [16], almost all measures consider the coordinate information in their computation. To grasp spatial information of trajectories, we exploit LSTM network to further process the coordinate tuples.

Given a trajectory $T = [(p_1, g_1), (p_2, g_2), ..., (p_i, g_i), ...]$ where $p_i = (lon_i, lat_i), g_i = (x_i, y_i)$, LSTM is used to preserve the spatial information directly from the geographical coordinate tuples (lon_i, lat_i). At each time step, we feed the coordinate tuple (lon_t, lat_t) to LSTM according to their order in the trajectory. Then LSTM unit will exploit the coordinate tuple, the hidden state and the cell state from the previous step to compute the new hidden state and update the cell state. We use the hidden state vector $\boldsymbol{h}_l$ at final time step l as the representation of a trajectory because it contains spatial information of all the coordinate tuples. The spatial information preserved representation is learned by the recurrent procedures that process the coordinate tuples and capture the the correlation among these tuples.

Based on the structural information preserved representation and spatial information preserved representation, the final representation is obtain by a learning-based combination. In our model, we use a learnable parameter γ to balance the weights of results of attention based network and LSTM.

$$\boldsymbol{t} = \gamma \cdot \boldsymbol{h}_l + (1 - \gamma) \cdot \boldsymbol{h}_s \tag{10}$$

where $\boldsymbol{h}_l$ is the output of LSTM at the final time step and $\boldsymbol{h}_s$ is the output of the self-attention based network. By fusing $\boldsymbol{h}_l$ with $\boldsymbol{h}_s$, the trajectory representation $\boldsymbol{t}$ takes both structural and spatial information into consideration and balances these two kinds of information. Therefore, T3S is more generic and outperforms other methods on most distance measures.

IV. Experiments

In this section, we introduce the extensive experiment results that demonstrate the effectiveness and efficiency of T3S.

TABLE I
PERFORMANCE EVALUATION FOR METHODS UNDER DIFFERENT DISTANCE MEASURES.

Dataset	Method	Fréchet distance				DTW distance				ERP distance				Hausdorff distance			
		HR-10	HR-50	R10@50	R10@100	HR-10	HR-50	R10@50	R10@100	HR-10	HR-50	R10@50	R10@100	HR-10	HR-50	R10@50	R10@100
Geolife	SRN	0.4670	0.6094	0.8152	0.8998	0.2778	0.4009	0.6255	0.7670	0.7378	0.7543	0.9700	0.9873	0.3075	0.4324	0.6662	0.7880
	NeuTraj	0.5079	0.6617	0.8433	0.9189	0.3002	0.4262	**0.6619**	**0.7961**	0.7625	0.7907	0.9767	0.9895	0.3416	0.4761	0.6802	0.7913
	T3S	**0.5231**	**0.6732**	**0.8667**	**0.9363**	**0.3208**	**0.4316**	0.6601	0.7912	**0.7808**	**0.8053**	**0.9837**	**0.9938**	**0.3807**	**0.5463**	**0.7690**	**0.8650**
Porto	SRN	0.4720	0.5828	0.7749	0.8525	0.3810	0.4952	0.7727	0.8767	0.7177	0.7180	0.9819	0.9961	0.3800	0.4998	0.7421	0.8513
	NeuTraj	0.5466	0.6524	0.8453	0.9065	0.4341	0.5625	**0.8390**	0.9229	0.7552	0.7788	0.9891	0.9973	0.4645	**0.6009**	0.8288	0.9089
	T3S	**0.5518**	**0.6560**	**0.8550**	**0.9141**	**0.4345**	**0.5809**	0.8350	**0.9239**	**0.8080**	**0.8133**	**0.9965**	**0.9988**	**0.4672**	0.5977	**0.8344**	**0.9140**

[1] HR-k is the hitting ratio for top-k similarity search task.
[2] $R10@k$ is the top-k recall for the top-10 ground truth.

A. Experimental Settings

Datasets. The following two datasets are used in our experiments:

- Geolife [17] is a GPS-based trajectory dataset that was collected by 182 users from April 2007 to August 2012 in Beijing, China. This dataset contains 17,621 trajectories and records a broad range of human outdoor movements.
- Porto [18] is a dataset contains more than 1.7 millions vehicle route trajectories that were collected by 442 taxis in Porto, Portugal. This dataset serves as a benchmark to evaluate traffic monitoring models.

Compared models. To evaluate the effectiveness of T3S, we compare our model with the following state-of-the-art baselines:

- SRN [19] is a Siamese recurrent network (SRN)-based model which solves the time series similarity computation problem. Following the previous work [12], we instantiate the Siamese network with LSTM in our experiments.
- NeuTraj [12] is a neural metric learning method for trajectory similarity computation. NeuTraj augments LSTM network with a memory unit that retrieves historical information of processed trajectories, and achieves the state-of-the-art performance on trajectory similarity approximation task.
- T3S is our proposed model which exploits the LSTM network and self-attention mechanism to obtain the spatial and structural information of trajectories for representation learning.

Evaluation Metrics. Following previous work [12], we conduct the top-k similarity search experiments on the datasets to evaluate the performance of compared methods. The hitting ratio of top-k task is used to evaluate different methods which examines the overlap percentage of the top-k results and the ground truth results, *i.e.*, the percentage of ground truth trajectories that are recovered by the learned top-k results. Besides, we also employ the top-50 recall ($R10@50$) and the top-100 recall ($R10@100$) for the top-10 ground truth. These metrics count how many top-10 ground-truth trajectories are recovered by the top-50 lists or top-100 lists produced by different methods. Higher recall value indicates better performance. All evaluations are performed for the compared methods under four common trajectory distance measures: the Fréchet distance, the Hausdorff distance, the ERP distance and the DTW distance. We also evaluate the efficiency of the models by reporting the similarity computation time and the model training time.

Parameter Settings. The geographical space in both datasets is divided into a 1100×1100 sized grid matrix by every 0.001 longitude and latitude. The training ratio tr is set to 0.2, *i.e.*, 20% data will be used for training. Trajectory embedding dimension d and learning rate lr are set to 128 and 5×10^{-3} respectively. The attention-based network contains two layers, each of which has 16 attention heads. All experiments are conducted on a machine with Intel Xeon E-2288G @3.7GHz 8 cores CPU and NVIDIA Quadro RTX 6000 (4,608 Cores, 576 Tensor Cores, 24GB Memory) GPU.

B. Performance Evaluation

In this subsection, we report the evaluation results on top-k similarity search task for compared methods under the Hausdorff, the Fréchet, the DTW and the ERP distances. As shown in Table I, our model outperforms other models in almost all the evaluation metrics under these 4 distance measures. For the Hausdorff distance and the Fréchet distance, which consider the locations and ordering of the points in the trajectories, our model has a significant improvement against SRN and NeuTraj. Especially, our model has a great improvement under Hausdorff distance. As we can see, the hitting ratio of our model is almost 10% higher than SRN on the $R10@50$ and $R10@100$ tasks. Under these measures, the LSTM network takes a large proportion of the learnable weight γ because of it superiority on preserving the spatial information in the coordinate tuples. For the DTW distance, which solves the local time shift issue when computing the distance between trajectories, our model still improves the performance compared to SRN and NeuTraj, especially on the top-k hitting ratio. Under this distance function, T3S assigns weight γ almost equally to the self-attention based network and LSTM network. By taking advantage of these two networks, our model preserves both structural and spatial information, thus produces high-quality trajectory representations. Performance results in Table I prove the superior effectiveness of our proposed T3S. By capturing the structural and spatial information, T3S achieves better learning ability on trajectory similarity computation.

C. Efficiency Comparison

We evaluate the query time cost of different methods on similarity computation to compare the efficiency. We also report the training time under different similarity measures.

Online Similarity Computation Time. Given a trajectory, we compare the time cost on similarity computation under different distance measures. We randomly sample $1K$, $5K$, $10K$ and $200K$ trajectories from the Porto dataset as base

TABLE II
TIME COST OF COMPUTING SIMILARITY UNDER PORTO DATASET WITH DIFFERENT SIZE.

Method	1k	5k	10k	200k
	Fréchet distance			
BruteForce	6430.45s	154320.87s	620840.11s	N/A
NeuTraj	6.06s	30.67s	61.73s	1232.61s
Ours	**2.20s**	**11.28s**	**23.26s**	**462.07s**
	DTW distance			
BruteForce	238.04s	5640.29s	22637.15s	N/A
NeuTraj	6.09s	31.28s	61.72s	1234.10s
Ours	**2.25s**	**11.44s**	**23.03s**	**461.42s**
	ERP distance			
BruteForce	674.38s	16161.47s	64369.99s	N/A
NeuTraj	5.86s	30.76s	60.88s	1219.72s
Ours	**2.16s**	**11.32s**	**23.19s**	**461.23s**
	Hausdorff distance			
BruteForce	203.02s	4919.61s	19561.86s	N/A
NeuTraj	6.11s	30.29s	60.82s	1218.86s
Ours	**2.23s**	**11.48s**	**23.12s**	**460.84s**

* N/A means the computation cannot be finished in a week.

data trajectories. For the learning-based models, we evaluate the time cost of obtaining the representations of trajectories and computing the similarities between trajectories using the learned representations. The model that computes the distance measures according to their definitions introduced in Section II, namely BruteForce, is also compared. The results are summarized in Table II. Our model T3S is the most efficient among all compared methods. Overall, T3S achieves 2.5x - 3x speedup over NeuTraj and 100x speedup over BruteForce method. Deep learning based methods improve the efficiency significantly compared to the BruteForce method since T3S and NeuTraj both exploit neural network which enjoys lower time complexity. NeuTraj has to calculate attention scores at every recurrent step in LSTM, while T3S only computes the attention scores once during the trajectory similarity query. Therefore, T3S is more efficient to learn trajectory representations for trajectory similarity computation.

TABLE III
TRAINING TIME FOR DEEP LEARNING BASED METHODS.

Method	t_{epoch}	n_{epoch}	t_{total}
NeuTraj-Fré	110.92s	300	33276s
NeuTraj-DTW	109.28s	229	25025.12s
Ours-Fré	49.39s	164	8099.96s
Ours-DTW	50.68s	970	49159.6s

Offline Training Time. Beside trajectory similarity computation time, the training time costs of T3S and NeuTraj are also evaluated. Table III includes the training time costs for both methods. The results indicate that our method needs less time for each epoch. T3S employs LSTM and a self-attention based network. These networks are both efficient when they are used to process sequences. LSTM is simple-structured and easy to converge. The self-attention based network is also efficient, because it computes attention scores in parallel. Benefiting from concise structure and less parameters, T3S is faster to obtain the trajectory representations. Suffering from SAM module, NeuTraj needs more time for each epoch. For each grid cell, NeuTraj needs to find historical information of its surrounding $(2w+1)^2$ grid cells. With these historical information and cell state of LSTM network, NeuTraj needs to compute attention scores for every grid cell. Then the weighted historical information will be used to update the cell state. These operations will lead to a high complexity which makes NeuTraj inefficient for representation learning and parameter optimization. Besides, the read and write operations will also make NeuTraj not efficient enough.

REFERENCES

[1] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *The VLDB Journal*, vol. 29, no. 1, pp. 3–32, 2020.

[2] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 647–657, 2010.

[3] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *Int. J. Comput. Geom. Appl.*, vol. 5, pp. 75–91, 1995.

[4] Byoung-Kee Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings 14th International Conference on Data Engineering*, 1998, pp. 201–208.

[5] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '05, 2005, p. 491–502.

[6] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th International Conference on Data Engineering*, 2002, pp. 673–684.

[7] L. Chen and 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*, ser. VLDB '04, 2004, p. 792–803.

[8] H. Yuan and G. Li, "Distributed in-memory trajectory similarity search and join on road network," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 1262–1273.

[9] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "Ftw: Fast similarity search under the time warping distance," 01 2005, pp. 326–337.

[10] A. Backurs and A. Sidiropoulos, "Constant-distortion embeddings of hausdorff metrics into constant-dimensional l_p spaces," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, ser. LIPIcs, K. Jansen, C. Mathieu, J. D. P. Rolim, and C. Umans, Eds., vol. 60, 2016, pp. 1:1–1:15.

[11] A. Driemel and F. Silvestri, "Locality-sensitive hashing of curves," *CoRR*, vol. abs/1703.04040, 2017.

[12] D. Yao, G. Cong, C. Zhang, and 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)*, 2019, pp. 1358–1369.

[13] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "Ibat: Detecting anomalous taxi trajectories from gps traces," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, 2011, p. 99–108.

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

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 5998–6008.

[16] S. Wang, Z. Bao, J. S. Culpepper, and G. Cong, "A survey on trajectory data management, analytics, and learning," 2020.

[17] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Data Eng. Bull.*, vol. 33, pp. 32–39, 06 2010.

[18] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, 2013.

[19] W. Pei, D. M. J. Tax, and L. van der Maaten, "Modeling time series similarity with siamese recurrent networks," 2016.