Spatio-Temporal Trajectory Similarity Learning in Road Networks

Ziquan Fang
Zhejiang University
Hangzhou, China
zqfang@zju.edu.cn

Yuntao Du
Zhejiang University
Hangzhou, China
ytdu@zju.edu.cn

Xinjun Zhu
Zhejiang University
Ningbo, China
xjzhu@zju.edu.cn

Danlei Hu
Zhejiang University
Hangzhou, China
dlhu@zju.edu.cn

Lu Chen
Zhejiang University
Hangzhou, China
luchen@zju.edu.cn

Yunjun Gao
Zhejiang University
Hangzhou, China
gaoyj@zju.edu.cn

Christian S. Jensen
Aalborg University
Aalborg, Denmark
csj@cs.aau.dk

ABSTRACT

Deep learning based trajectory similarity computation holds the potential for improved efficiency and adaptability over traditional similarity computation. However, existing learning-based trajectory similarity learning solutions prioritize spatial similarity over temporal similarity, making them suboptimal for time-aware analyses. To this end, we propose ST2Vec, a representation learning based solution that considers fine-grained spatial and temporal relations between trajectories to enable spatio-temporal similarity computation in road networks. Specifically, ST2Vec encompasses two steps: (i) spatial and temporal modeling that encode spatial and temporal information of trajectories, where a generic temporal modeling module is proposed for the first time; and (ii) spatio-temporal co-attention fusion, where two fusion strategies are designed to enable the generation of unified spatio-temporal embeddings of trajectories. Further, under the guidance of triplet loss, ST2Vec employs curriculum learning in model optimization to improve convergence and effectiveness. An experimental study offers evidence that ST2Vec outperforms state-of-the-art competitors substantially in terms of effectiveness and efficiency, while showing low parameter sensitivity and good model robustness. Moreover, similarity involved case studies including top-k querying and DBSCAN clustering offer further insight into the capabilities of ST2Vec.

CCS CONCEPTS

• **Information systems → Traffic analysis**.

KEYWORDS

trajectory similarity, spatio-temporal representation, road networks

ACM Reference Format:
Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S. Jensen. 2022. Spatio-Temporal Trajectory Similarity Learning in Road Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539375

1 INTRODUCTION

With the proliferation of geo-positioning capabilities, massive spatio-temporal trajectories of moving objects (e.g., people and vehicles) are collected, which motivates various trajectory analytics [7]. A trajectory T is represented as a time-ordered sequence of n discrete points, i.e., $T = \langle (g_1, t_1), (g_2, t_2), ..., (g_n, t_n) \rangle$, where g denotes a geo-location and t is the corresponding time. Trajectory similarity computation is an ingredient in a range of trajectory analyses that benefit many real-world applications such as ridesharing [20], traffic analysis [7], and social recommendation [18].

A number of traditional measures of the similarity between two trajectories exist, including free-space measures such as DTW [34], LCSS [27], Hausdorff [1], and ERP [5], as well as road network measures such as TP [21], DITA [24], LCRS [36], and NetERP [12]. However, these measures incur high computation costs [10, 14], as they typically rely on pointwise matching, incurring quadratic level computational complexity. As a result, the high computation cost of the above measures becomes a bottleneck when dealing with massive trajectory data [33, 37].

To address this problem, recent studies [10, 32, 33, 37] utilize neural network based models to learn deep representations of trajectories, so that the similarity relations between trajectories are preserved in the low-dimensional embeddings, the use of which has been proved to yield speedups over the use of techniques that operate directly on the original trajectories. While the above approaches thus offer improved efficiency, they still come with several limitations. In particular, they all disregard the temporal dimension of spatio-temporal trajectories. That is, they learn embeddings while considering only the spatial dimensions of trajectories, i.e., $T^{(s)} = \langle g_1, g_2, ..., g_n \rangle$. As a result, they are ineffective in settings where the temporal aspect is important. Overall, considering both

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539375

spatial and temporal similarity is important in time-aware applications such as transportation planning [26] and monitoring [35].

In this paper, we adopt an orthogonal but complementary approach to existing space-driven similarity learning studies–we address the problem of spatio-temporal trajectory similarity learning in road networks. Although existing studies [10, 32, 33, 37] offer guidance for spatial similarity modeling, three non-trivial challenges remain, including temporal similarity learning, spatio-temporal fusion, and model optimization.

Challenge I: How to capture temporal correlations between trajectories for temporal similarity learning? The core task is to generate time-oriented trajectory embeddings that preserve the temporal similarity relations (i.e., close or distant) between trajectories. To achieve this, a natural idea is to feed the time sequences contained in trajectories, i.e., $T^{(t)} = \langle t_1, t_2, ..., t_n\rangle$, to RNN models to capture the sequence information, similarly to existing spatial similarity learning that feeds spatial sequences $T^{(s)}$ to RNNs. However, temporal modeling is more challenging. Unlike spatial locations of trajectories that are discrete and enable the evaluation of spatial distances by specific measures (e.g., L1 and L2), the time information exhibits continuous and periodic patterns. First, time never stops, and it is hard to be discretized with satisfied granularity. Second, trajectories show strong periodicity, resulting in seconds, hours, days, etc. Thus, the time representation must be invariant to time rescaling. Overall, directly feeding time information into RNNs for temporal dimensional embedding is ineffective since it does not contend with temporal characteristics. After twice attempts with non-trivial efforts (Section 4.2), we eventually design a temporal modeling module, termed **TMM**, to achieve effective temporal trajectory representation learning. It is worth mentioning, TMM is flexible and generic, so that it can be integrated with any existing spatial trajectory similarity learning proposals to perform spatio-temporally aware trajectory similarity computation.

Challenge II: How to fuse spatial and temporal characteristics to generate unified trajectory embeddings for spatio-temporal similarity learning? Once spatial and temporal characteristics are captured, we need to fuse these to obtain unified spatio-temporal trajectory embeddings. However, different applications may assign different weights to spatial and temporal similarity. For example, applications such as region function estimation [13] may assign high importance to spatial aspects of trajectories and thus assign high weight to spatial similarity. In contrast, applications such as ridesharing [17] may assign high importance to the temporal aspects and thus assign high weight to temporal similarity. Overall, a preferable fusion approach must be robust to learn different spatial and temporal weights adaptively and not hurt model convergence, especially when the time and spatial dimensions are considered jointly. To address this challenge, we develop a spatio-temporal co-attention fusion module, termed **STCF**, that fuses spatial and temporal characteristics of trajectories to generate unified embeddings.

Challenge III: How to optimize the learning of embeddings to improve effectiveness and efficiency? The two primary goals of the learning of embeddings are effectiveness (quality of embeddings) and efficiency (model convergence). Specifically, the training samples, learning procedure, and network parameters all affect model performance. To improve effectiveness, we utilize triplet loss [19] and then train models with curriculum leaning. To avoid an excess of parameters due to the joint spatial and temporal modeling and to improve efficiency, we present two different fusion approaches in the co-attention fusion module.

To address all three challenges, we propose a representation learning architecture, termed ST2Vec, to enable effective and efficient learning of trajectory embeddings that enable spatio-temporally aware trajectory similarity analytics in road networks. To sum up, the main contributions are as follows.

- To the best of our knowledge, this is the first proposal to study time-aware trajectory representation and spatio-temporal fusion for trajectory similarity learning. ST2Vec accommodates varying spatial and temporal weights and a series of classic measures, enabling flexible analyses.
- After twice attempts in temporal modeling, we propose a generic temporal representation learning module. Further, we propose a spatio-temporal co-attention fusion module with two different fusion strategies (SF and UF) to integrate the spatial and temporal characteristics of trajectories effectively and efficiently.
- In terms of the task of learning-based trajectory similarity computation, we are the first to bring the triplet and curriculum concept to guide the learning process, further improving the model accuracy and convergence.
- Experiments on four popular network-aware trajectory measures show that ST2Vec outperforms all competitors in terms of effectiveness and efficiency, while showing low parameter sensitivity. An ablation study verifies the efficacy of key decisions. In addition, two real case studies demonstrate the downstream capabilities of ST2Vec.

2 RELATED WORK

Non-learning-based methods [12, 21, 24, 29–31, 36] rely on well-defined similarity measures and acceleration techniques. Here, we focus on network-aware similarity measures, while a comprehensive coverage of free-space based measures is available elsewhere [7, 25]. Network-aware similarity computation techniques first map trajectories to road-network paths that consist of vertices or segments. Then, they define similarity measures based on classic distance measures such as Hausdorff [1], DTW [34], LCSS [27], and ERP [5], generally by aggregating the distances between road vertices or segments of two trajectories. For example, Koide et al. [12] propose NetERP by aggregating shortest-path distances between the vertices of two trajectories. Based on LCSS, Wang et al. [29, 30] propose the Longest Overlapping Road Segments (LORS) for similarity computation and trajectory clustering. Further, Yuan et al. [36] propose the direction-aware Longest Common Road Segments (LCRS). These methods typically have high computational complexity, as provided in the Appendix A due to limited space.

Learning-based methods [10, 14, 32, 33, 37] are becoming increasingly popular, as they leverage advances in deep learning technologies such as their increasingly powerful approximation capabilities. The learning-based methods learn distance functions that embed input trajectories and approximate given distance measures. This way, trajectory embeddings are generated that enable fast trajectory similarity computation and downstream analyses. Li et al. [14]

propose t2vec that takes into account low sampling rates and the influence of noisy points in trajectories for deep representation learning, not similarity metric learning, as t2vec evaluates the similarity between a given trajectory and its low-sampled or noisy variants, instead of any pair of trajectories. Yao et al. [33] propose NEUTRAJ, which employs metric learning to approximate trajectory similarity for different free-space based distance measures. Further, Zhang et al. [37] propose Traj2SimVec, which considers sub-trajectory similarity in the learning process. Zhang et al. [32] propose T3S, which utilizes attention to improve the performance. Liu et al. [15] focus on semantic-aware trajectory similarity learning. While these studies all make advances, they all target spatial trajectory similarity in free space. Most recently, Han et al. [10] develop GTS, which targets trajectory similarity learning in road networks and achieves state-of-the-art performance. Specifically, GTS is designed for POI-based spatial trajectory similarity computation. As a result, GTS treats trajectories that share the same or neighboring POIs, but have totally different paths, as similar. Further, GTS learns a single type of distance measure (i.e., TP [21], an extension to the Hausdorff distance), while ST2Vec accommodates a range of popular measures including TP [21], DITA [24], LCRS [36], and NetERP [12]. More importantly, all above approaches ignore the temporal aspect, although time is an essential aspect of trajectories and deserves attention on par with the spatial aspect.

3 PRELIMINARIES

3.1 Road Networks & Trajectories

Definition 3.1. **(Road Network)** *A road network is modeled as a directed graph $G = (L, E)$, where L is a set of road vertices and $E \subseteq L \times L$ is an edge set of road segments. Specifically, a vertex $l_i = (x_i, y_i) \in L$ models a road intersection or a road end, in which x_i and y_i denote the longitude and latitude of l_i, respectively. An edge $e_{l_i,l_j} \in E$ represents a directed road segment from l_i to l_j.*

GPS trajectory T of a moving object is initially captured as a time-ordered sequence of sample points, i.e., $T = \langle (g_1, t_1), (g_2, t_2), ..., (g_n, t_n) \rangle$, where n is the length of T. Each sample point is represented as a 2-dimensional (location, time) tuple, i.e., $(g_i, t_i), i \in [1, n]$. Here, g denotes an observed geo-location that consists of a longitude and a latitude, and t denotes the corresponding time. As we target road-network constrained trajectory similarity learning, we align trajectory points g with road vertices l using an existing map-matching procedure (e.g., [2]). Consequently, each original trajectory T is transformed into a directed path in G from a start vertex to an end vertex, as defined below.

Definition 3.2. **(Trajectory)** *Given a road network $G = (L, E)$, a trajectory T is a directed sequence of m vertices in G, i.e., $T = \langle (l_1, t_1), (l_2, t_2), ..., (l_m, t_m) \rangle$, where $l_i \in L$ denotes a vertex and t_i represents the corresponding time.*

Unless stated otherwise, we assume in the sequel that trajectories are map matched. Given a trajectory T, we use $T^{(s)}$ and $T^{(t)}$ denote its spatial and temporal aspects, respectively, i.e., $T^{(s)} = \langle l_1, l_2, ..., l_m \rangle$ and $T^{(t)} = \langle t_1, t_2, ..., t_m \rangle$. Note that trajectories $T^{(s)}$ and $T^{(t)}$ have the same number of aligned points.

3.2 Spatio-Temporal Similarity

Remark. Before performing similarity learning, a similarity metric must be chosen that serves as the learning target. Most existing studies [32, 33, 37] use free space measures (Hausdorff [1], DTW [34], LCSS [27], and ERP [5]) for trajectory similarity learning in Euclidean space, or they [10] use road-network measures (TP [21]) for trajectory similarity learning in spatial networks. Since we target at general spatio-temporal similarity learning to support a series of classic similarity metrics (instead of a specific similarity metric), to the best of our knowledge, no previous studies exist but a linear combination to meet this target.

Given trajectories T_i and T_j, we define the spatio-temporal similarity function $\mathcal{D}(T_i, T_j)$ as a weighted, linear combination of their spatial and temporal similarity[1]. This definition of spatio-temporal similarity is simple and flexible, and it is used widely in previous non-learning trajectory similarity studies [21–23]. This is natural because most existing similarity metrics support either only spatial or only temporal similarity computation.

$$\mathcal{D}\left(T_i, T_j\right) = \lambda \cdot \mathcal{D}_S\left(T_i^{(s)}, T_j^{(s)}\right) + (1 - \lambda) \cdot \mathcal{D}_T\left(T_i^{(t)}, T_j^{(t)}\right) \quad (1)$$

Since we study trajectory similarity in road networks, $\mathcal{D}$ denotes network-aware distance measures including TP [21], DITA [24], LCRS [36], and NetERP [12]. Further, $\mathcal{D}_S$ and $\mathcal{D}_T$ denote spatial and temporal similarity, respectively. Although these distance measures are predominantly oriented towards spatial proximity, they are also able to support temporal similarity. According to Shang et al. [21], given a trajectory T, its spatial sequence $T^{(s)}$ and temporal sequence $T^{(t)}$ both support distance aggregation between sequences for similarity evaluations. Since we aim to enable similarity learning across different measures without modifying these measures or their implementations, we do not cover their detailed implementations, but instead refer the interested reader to the literature [7]. Further, parameter $\lambda \in [0, 1]$ controls the relative weight of spatial and temporal similarity, providing flexibility that can be used to support different applications as discussed in Section 1.

4 PROBLEM STATEMENT

4.1 Problem Formulation

Same to related similarity learning studies [10, 32, 33, 37], the problem is to learn a neural network $\mathcal{G}(\cdot, \cdot)$ such that $\mathcal{G}\left(v_{T_i}, v_{T_j}\right)$ reflects the actual similarity relation defined by $\mathcal{D}\left(T_i, T_j\right)$. That is, v_{T_i} and v_{T_j} are close (resp. distant) to each other in the embedding space if T_i and T_j are similar (resp. dissimilar) in the original space. Here, v_{T_i} and v_{T_j} are the embeddings of T_i and T_j.

4.2 Alternative Solutions

As none previous studies consider the temporal dimension of trajectory learning, before introducing our methods, we have tried two solutions (i.e., **A1** and **A2**) for temporal modeling to realize spatio-temporal trajectory similarity learning.

A1: This solution first split the time axis into discrete time intervals and then assign trajectories to different intervals using

[1] It is worth mentioning that Equation (1) only serves as the model learning target (i.e., ground-truth) and has no effects on ST2Vec design.

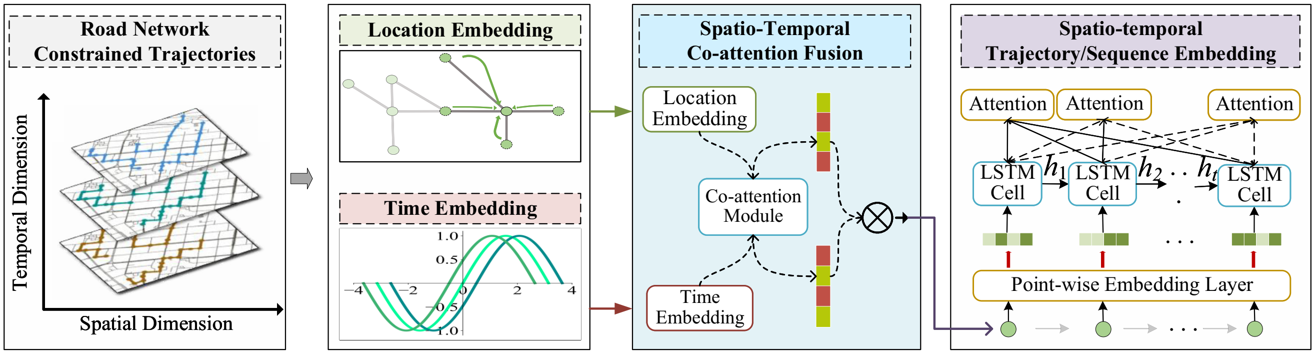


Figure 1: An Overview of the ST2Vec under Unified Fusion

sliding windows. After this pre-processing, one can conduct similarity computations for each time interval using any of the existing spatial proposals. However, this approach is spatially-oriented, is coarse-grained, and is suboptimal. Further, time is continuous and unbounded, making it difficult to determine an appropriate window length, and regardless of the length chosen, inaccurate or incorrect spatio-temporal similarity computations are inevitable. In addition, this approach relies on discrete time and trajectory processing, incurring additional processing costs.

A2: Another solution is to feed the temporal trajectories (i.e., $T^{(t)}$) to RNNs the same way that spatial trajectories (i.e., $T^{(s)}$) are fed to RNNs. Then, the resulting temporal vectors are combined with the spatial vectors using existing methods [10, 32, 33, 37] to achieve spatio-temporal similarity learning. Although this approach is more reasonable than the first, it is also naive. As discussed in Section 1, temporal correlations are more complex than spatial correlations because time exhibits continuous and periodic patterns. Consequently, simply applying spatial embedding methods to embed time is likely to be sub-optimal.

We also study two alternative approaches experimentally and offers detailed insight into their performance.

5 THE ST2VEC APPROACH

ST2Vec enables unified spatio-temporal similarity learning by means of three modules: temporal modeling module (TMM), spatial modeling module (SMM), and spatio-temporal co-attention fusion (STCF) module. Specifically, the TMM and SMM produce spatial and temporal representations of trajectories that are then passed to STCF module that enables separate fusion and unified fusion that fuse space-specific and the time-specific features. Then, these features are aggregated to produce the final spatio-temporal embeddings. Fig. 1 illustrates this using unified fusion. Next, we first detail the three modules of ST2Vec. Then we describe the training of ST2Vec for trajectory similarity learning.

5.1 Temporal Modeling Module (TMM)

Given a set of temporal trajectories, it is natural to use sequence models like RNNs or their variant, to represent the trajectories as vectors. However, this fails to capture periodic and non-periodic temporal patterns in the trajectories.

Basic idea. To achieve fine-grained temporal representation learning, we propose *time embedding* and *temporal sequence embedding* to obtain a trajectory-aware temporal modeling module. Further, we notice that different time points, e.g., rush hour vs. late night, may have different importance. Thus, we further introduce an attention mechanism to enhance the representation of temporal irregularity.

5.1.1 *Time Embedding.*

Inspired by the position embedding in BERT [28], for each time point t in a temporal trajectory, we learn its time embedding t', which is a vector of size $q + 1$.

$$t'[i] = \begin{cases} \omega_i t + \varphi_i, & \text{if } i = 0 \\ \cos(\omega_i t + \varphi_i), & \text{if } 1 \le i \le q \end{cases} \tag{2}$$

Here, $t'[i]$ denotes the i-th element of t', $\omega_0, ..., \omega_q$ and $\varphi_0, ..., \varphi_q$ are learnable parameters, and $\cos(\cdot, \cdot)$ serves as a periodic activation function that helps capture periodic behaviors without the need for feature engineering. For $1 \le i \le q$, ω_i and φ_i are the frequency and the phase-shift of the cos function, and thus the period of the cosine function is $\frac{2\pi}{\omega_i}$, i.e., it has the same value at t and $t + \frac{2\pi}{\omega_i}$. The linear term represents the progression of time and can be used for capturing non-periodic patterns in the input that depend on time. Based on Eq. 2, we embed a temporal trajectory $T^{(t)}$ into a sequence of time vectors, i.e., $\langle t_1, t_2, ..., t_m\rangle \rightarrow \langle t'_1, t'_2, ..., t'_m\rangle$.

5.1.2 *Temporal Sequence Embedding.*

As shown in Fig. 1, if we remove the spatio-temporal co-attention fusion module, after embedding each time point in a trajectory, we can feed $\langle t'_1, t'_2, ..., t'_m\rangle$ to an LSTM architecture to model its temporal dependence. The recurrent step of an LSTM is performed as follows. At each step i, an LSTM cell takes as input the current input vector x_i and the state vector of the previous step h_{i-1}, and it outputs the state vector of the current step h_i.

$$h_i = \text{LSTM}\left(t'_i, h_{i-1}, i_i, f_i, o_i, m_i\right), \tag{3}$$

where i_i, f_i, o_i, and s_i represent an input gate, a forget gate, an output gate, and a memory cell, respectively. More details on LSTMs are available elsewhere [3]. In the context of our LSTM layer, t'_i is the learned time embedding that corresponds to the original time t_i. The LSTM unit exploits the embedded time, the hidden state, and the cell state from the previous step to compute the new hidden state and to update the cell state. Finally, we use the last hidden state vector h_t as the deep temporal representation because it contains all temporal information of the trajectory. Overall, a temporal information preserving representation is learned by the recurrent procedure that processes the time points and captures the correlations among them.

5.1.3 ***Decoupled Attention.*** Different time points in a temporal trajectory may carry different weights in subsquent temporal similarity computations. To support this and improve model effectiveness, we employ attention mechanisms to capture the correlations between trajectory points. Specifically, we propose a self-attention mechanism to compute attention scores between different time points in the same trajectory as follows.

$$\tilde{h}_i^{(p)} = \sum_{k=1}^{i} \operatorname{att}\left(h_i^{(p)}, h_k^{(p)}\right) \cdot h_k^{(p)} \tag{4}$$

Here, $\tilde{h}_i^{(p)}$ denotes the improved state representation, and $\operatorname{att}(\cdot,\cdot)$ is an attention function:

$$\operatorname{att}\left(h_i^{(p)}, h_k^{(p)}\right) = \frac{\alpha_{i,k}}{\sum_{k'=1}^{i} \exp\left(\alpha_{i,k'}\right)}, \tag{5}$$

where, $\alpha_{i,k} = w_1^{\top} \cdot \tanh\left(W_1 \cdot h_k^{(p)} + W_2 \cdot h_i^{(p)}\right)$ and w_1, W_1, and W_2 are the parameter vector and matrices to learn. By integrating the attention mechanism into the temporal embedding, we can discover more important time points, in turn improving model performance. Note that we also use the hidden representation of the last step to encode the full temporal trajectory.

5.2 Spatial Modeling Module (SMM)

5.2.1 SMM vs. Existing Studies. Since several studies [10, 32, 33, 37] address spatial trajectory modeling, we first detail the main difference between them and ST2Vec. Most of the studies [32, 33, 37] consider trajectory similarity in free space and adopt RNN-type models to capture the sequence information for spatial representation learning. However, objects such as people and vehicles move in road networks [20], these studies do not reflect the real distances between trajectories caused by the movement restrictions imposed by road networks. A recent study [10] combines GNNs with LSTMs to enable road network constrained trajectory representation learning and achieves state-of-the-art similarity learning performance. However, the study assumes POI-based similarity computation. This has the effect of giving more significance to POIs while ignoring detailed travel paths, which can yield inaccuracies when measuring the similarity between trajectories that share same POIs but have different travel paths. In contrast to all of the above studies, we target fine-grained spatial similarity learning in road networks, which considers both the locations (i.e., sample points) and paths when evaluating the similarity between trajectories.

Basic idea. Given a spatial trajectory $T^{(s)} = \langle l_1, l_2, ..., l_m \rangle$ (the l_i denote vertices in road network G), we aim to embed $T^{(s)}$ into a vector $v_{T^{(s)}}$ in the low-dimensional space while capturing the trajectory's road-network constrained spatial information. It is natural to utilize GNNs, which have been used successfully in road-network settings for tasks such as traffic prediction [38] to take into account the structure of G. Hence, to achieve spatial similarity oriented representation learning, we develop a spatial modeling module (SMM), which also encompasses three phases, i.e., location embedding, spatial sequence embedding, and spatial attention. Next, we detail three phases in the order.

5.2.2 ***Location Embedding.*** The trajectories of objects moving in a road network are constrained by the topology of the road network. Thus, the road-network distance between two sample points that are close in Euclidean space can still be large. To capture the structural information, we first utilize Node2Vec [9] that aims to capture the co-occurrence of the adjacent locations in road networks. Specifically, given a vertex l_i, we use Node2Vec to approximate the spatial conditional probability of vertices in its neighborhood, i.e., we perform the mapping $l_i \rightarrow n_i$, where l_i and n_i denote the original and embedded locations, respectively. Then, locations sharing similar neighborhoods tend to have similar embeddings. Next, we feed the embedded locations (i.e., the n_i) to a GCN step by step to obtain locally smoothed location embeddings, where spatially close locations tend to be close in the latent space. Given a road network G and a low-dimensional representation n_i of location $l_i \in G$, we define the GCN function as follows.

$$l_i' = \operatorname{GCN}(n_i) = \sigma\left(\left(\sum_{j \in N_i} c_{ij} W_s n_j\right) || n_i\right) \tag{6}$$

Here, l_i' is the location representation of location l_i, σ is a non-linear activation function, c_{ij} is an adjacency weight, $W_s \in \mathcal{R}^{d \times d}$ is a learnable matrix shared by all vertices in G, $||$ denotes the concatenation operation, and $\mathcal{N}_i$ is the set of neighbor vertices of n_i in G. Based on Node2Vec and Eq. 6, we obtain a fine-grained representation of each spatial trajectory $T^{(s)}$, i.e., $\langle l_1, l_2, ..., l_m \rangle \rightarrow \langle l_1', l_2', ..., l_m' \rangle$.

5.2.3 ***Spatial Sequence Embedding*** *&* ***Attention.*** As illustrated in Fig. 1, if we remove the spatio-temporal co-attention fusion module, we obtain a sequence of location vectors as input for the LSTM model. This spatial sequence embedding is similar to the temporal sequence embedding in Eq. 3.

Overall, given a spatial trajectory, based on Node2Vec and Eq. 6, we first obtain an initial sequence representation and feed that to an LSTM model to encode the spatial information. Further, the self-attention mechanism used in the TMM is also applied to capture different contributions of the different locations in the learning process. We do this because different location points in a trajectory also contribute differently to similarity computations. For instance, noisy location points with obvious deviations from other points typically have high influence on the similarity computation. Finally, we use the hidden state of the last step of the LSTM model as the deep spatial trajectory representation.

5.3 Spatio-Temporal Co-attention Fusion (STCF)

We proceed to fuse the extracted spatial and temporal characteristics of trajectories. We propose a spatio-temporal co-attention fusion module that features two fusion strategies with different purposes. Fig. 1 shows ST2Vec using the UF strategy.

5.3.1 Separate Fusion (SF). We embed temporal trajectories and spatial trajectories using *temporal sequence embedding* (Section 5.1.2) and *spatial sequence embedding* (Section 5.2.3), respectively. Thus, a straightforward approach is to first generate spatial and temporal embeddings with two separate LSTM models and then combine the two types of embeddings. More specifically, given a trajectory T

with temporal embedding $\tau^{(t)} = (t'_1, t'_2, ..., t'_m)$ and spatial embedding $\tau^{(s)} = (l'_1, l'_2, ..., l'_m)$, we define the spatio-temporal trajectory embedding using separate fusion follows.

$$v_T = LSTM_t(t'_1, t'_2, ..., t'_m) + LSTM_s(l'_1, l'_2, ..., l'_m) \tag{7}$$

Although this approach is simple, it requires two LSTM models to separately train and capture the temporal and spatial information, doubling the number of parameters that need to be determined in LSTMs. To improve model convergence and efficiency, we propose another fusion strategy.

5.3.2 Unified Fusion (UF). Given a trajectory, using the procedure of *time embedding* and *location embedding*, we could obtain its temporal trajectory embedding, denoted by $\tau^{(t)} = \langle t'_1, t'_2, ..., t'_m \rangle$, and its spatial trajectory embedding, denoted by $\tau^{(s)} = \langle l'_1, l'_2, ..., l'_m \rangle$. Since these representations capture different aspects of trajectory properties, we enhance them by letting them interact with each other, as depicted in Fig. 1. Specifically, we first make a transformation for the temporal and spatial features via a matrix W_F.

$$z^1_\tau = W_F \tau^{(t)}, \quad z^2_\tau = W_F \tau^{(s)} \tag{8}$$

The spatio-temporal interaction is calculated as follows.

$$\begin{aligned} \beta_{i,j} &= \frac{\exp\left(W_{Q'} z^i_\tau \cdot W'_K z^{j^T}_\tau\right)}{\sum_{j' \in \{1,2\}} \exp\left(W_{Q'} z^i_\tau \cdot W'_K z^{j'T}_\tau\right)}, \\ \tau^{(\hat{t})} &= \text{Norm}\left(FFN'\left(\beta_{1,1} z^1_\tau + \beta_{1,2} z^2_\tau\right) + \tau^{(t)}\right), \\ \tau^{(\hat{s})} &= \text{Norm}\left(FFN'\left(\beta_{2,1} z^1_\tau + \beta_{2,2} z^2_\tau\right) + \tau^{(s)}\right), \end{aligned} \tag{9}$$

Here, W'_Q and W'_K are matrices of the same shape as W_F, and $\tau^{(\hat{t})}$ and $\tau^{(\hat{s})}$ are the enhanced representations of $\tau^{(t)}$ and $\tau^{(s)}$. As shown in Fig 1, we then feed the enhanced initial temporal and spatial sequence embeddings into a single LSTM model for unified spatio-temporal trajectory embedding. This fusion strategy is formally defined as follows, where + denotes the Concat operation.

$$v_T = LSTM(\tau^{(\hat{t})} + \tau^{(\hat{s})}) \tag{10}$$

5.4 Training and Model Optimization

5.4.1 Similarity Construction. Following previous studies, we use the dot product between the embedding vectors of trajectories to denote the similarity between them. With the embedding vectors of trajectories T_i and T_j being v_{T_i} and v_{T_j}, the similarity score $\mathcal{G}(v_{T_i}, v_{T_j})$ is given by $\mathcal{D}\left(T_i, T_j\right) = v^{\mathrm{T}}_{T_i} v_{T_j}$.

5.4.2 Objective Function. Existing studies (e.g., [10, 37]) typically use pair-wise loss as the objective function. Inspired by the triple learning in image recognition and clustering [19], we propose and use a simple but effective triplet-based pair-wise loss. Given a dataset $\mathcal{T}$, we randomly select one trajectory as an anchor T_a and sample a relatively similar (resp. relatively dissimilar) trajectory as its positive T_p (resp. its negative T_n) trajectory. Such a triple of an anchor, a positive, and a negative trajectory form a similarity triplet (T_a, T_p, T_n). The triplets provide trajectory samples in terms of similarities and dissimilarities, which contributes to making the trained model effective and robust.

$$\max \sum_{T_p \in \mathcal{T}, T_n \in \mathcal{T} \setminus \{T_a\}} \mathbb{I}\left(\mathcal{D}\left(T_a, T_p\right) < \mathcal{D}\left(T_a, T_n\right)\right), \tag{11}$$

where T_p (resp. T_n) is the relatively similar (resp. relatively dissimilar) trajectory for trajectory T_a and $\mathbb{I}$ is the indicator function that evaluates to one if the condition holds and otherwise evaluates to zero. To reduce the training time, following state-of-the-art study [10], we randomly sample one trajectory instead of traversing all trajectories for the given anchor trajectory.

5.4.3 Training Optimization. We observe that all existing trajectory similarity learning methods use random training instances for learning and often converge slowly. Recently studies in text generation [4] and translation [16] suggest that organizing training samples from easy to hard accelerates the learning. Such an organization in human learning is referred to as a curriculum. In view of this, given a trajectory anchor T_a and its similar (dissimilar) ones, we can order the triplets with the easy ones first (i.e., the ones most dissimilar to T_a), followed by the hard ones (i.e., the ones most similar to T_a). Then, we use the those triplets from easy to hard. This way, we expect ST2Vec to achieve faster convergence and higher accuracy, to be validated experimentally. Since the characteristics of time (i.e., continuousness and periodicity) are irrelevant to the specific trajectory samples, we pretrain the time embeddings using Eq. 2 to further accelerate the convergence of training.

6 EXPERIMENTAL STUDY

6.1 Experimental Settings

Datasets. Two public real-life trajectory data sets are used, including Beijing[2] and Rome[3].

- **Beijing** contains 15 million taxi trajectory points from Beijing, China, collected from Feb. 2 to Feb. 8, 2008.
- **Rome** contains 367,052 trajectories from taxis in Rome, Italy, covering 30+ days.

Pre-processing. We first map match [2] all trajectories to the corresponding road networks from OpenStreetMap. This way, the raw GPS trajectory data is transformed into time-ordered vertex sequences, in accordance with Definition 3.2. Further, we acquire trajectories from urban areas and remove trajectories with fewer than 10 sampling points. This preporcessing yields 348,210 trajectories in Beijing and 45,157 trajectories in Rome.

Evaluation Metrics. Same to existing similarity learning studies [10, 32, 33, 37], we utilize top-k similarity search for validation, adopting HR@10, HR@50, and R10@50 as evaluation metrics. The ground-truth results of top-k similarity search are the exact top-k similarity search results obtained when using traditional non-leaning based similarity measures, including TP [21], DITA [24], LCRS [36], and NetERP [12]. Then, to evaluate the effectiveness of similarity learning, we compare the top-k results returned by the leaning-based methods with the top-k results produced by the non-learning methods. Specifically, HR@k denotes the top-k hitting ratio that captures the degree of overlap between a top-k result and

[2] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

[3] https://crawdad.org/roma/taxi/20140717/

Table 1: ST2Vec vs. Window-guided baselines, LSTM-guided baselines, and our TMM-guided baselines on Beijing and Rome.

Datasets	Methods	TP Distance			DITA Distance			LCRS Distance			NetERP Distance		
		HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50
Beijing	NEUTRAJ^w	0.0978	0.1373	0.1582	0.0805	0.1243	0.1442	0.0357	0.0419	0.0861	0.0054	0.0173	0.0198
	Traj2SimVec^w	0.0827	0.1261	0.1397	0.053	0.0682	0.1151	0.016	0.098	0.1861	0.0209	0.0986	0.1010
	T3S^w	0.1295	0.1733	0.2045	0.0838	0.1266	0.1489	0.0435	0.0678	0.1187	0.0125	0.0292	0.0388
	GTS^w	0.3034	0.3980	0.6975	0.1178	0.2223	0.3991	0.0188	0.0538	0.0652	0.0252	0.0408	0.0505
	NEUTRAJ^l	0.1765	0.2221	0.2703	0.0767	0.1103	0.1340	0.0533	0.1126	0.1694	0.0259	0.0502	0.0736
	Traj2SimVec^l	0.1446	0.1902	0.2263	0.0526	0.0642	0.1071	0.0329	0.1397	0.2257	0.0328	0.1050	0.1244
	T3S^l	0.1535	0.1984	0.2382	0.0806	0.1191	0.1422	0.0486	0.0904	0.1445	0.0193	0.0398	0.0563
	GTS^l	0.3709	0.4756	0.7965	0.1277	0.2321	0.4143	0.0360	0.1074	0.1342	0.0398	0.0655	0.0894
	NEUTRAJ^t	0.3371	0.4091	0.7001	0.1412	0.2719	0.4892	0.0924	0.2848	0.3632	0.1086	0.1832	0.2841
	Traj2SimVec^t	0.3987	0.5364	0.6593	0.1321	0.3072	0.3643	0.0968	0.2826	0.3741	0.2128	0.3212	0.5553
	T3S^t	0.3944	0.5011	0.7917	0.1284	0.2288	0.4073	0.1442	0.4331	0.5672	0.1464	0.2767	0.4077
	GTS^t	0.4243	0.5640	0.8026	0.3244	0.4370	0.6381	0.1643	0.4427	0.6242	0.2154	0.3477	0.5343
	ST2Vec	**0.4624**	**0.5868**	**0.8361**	**0.3861**	**0.5251**	**0.7356**	**0.1806**	**0.5469**	**0.7293**	**0.3152**	**0.4225**	**0.6468**
Rome	NEUTRAJ^w	0.0976	0.1499	0.1775	0.0898	0.1417	0.1756	0.0405	0.1552	0.2488	0.0053	0.0397	0.0723
	Traj2SimVec^w	0.0552	0.089	0.0973	0.0363	0.0391	0.0753	0.0057	0.026	0.0314	0.1191	0.2235	0.2728
	T3S^w	0.1098	0.1863	0.2228	0.0893	0.1426	0.1823	0.0669	0.1766	0.2910	0.0123	0.0512	0.0871
	GTS^w	0.1738	0.3775	0.4952	0.0872	0.1612	0.2636	0.1915	0.2677	0.4798	0.0697	0.1508	0.1869
	NEUTRAJ^l	0.1225	0.2177	0.2613	0.0932	0.1499	0.1950	0.0864	0.1981	0.3308	0.0172	0.0608	0.1004
	Traj2SimVec^l	0.1108	0.2287	0.2712	0.0544	0.0772	0.1336	0.0992	0.1350	0.2331	0.1151	0.2205	0.2787
	T3S^l	0.1195	0.2092	0.2508	0.0931	0.1494	0.1931	0.0805	0.1930	0.3209	0.0156	0.0582	0.0969
	GTS^l	0.1891	0.4188	0.5405	0.0896	0.1644	0.2678	0.2217	0.2985	0.5361	0.0732	0.1566	0.2001
	NEUTRAJ^t	0.2092	0.4725	0.5986	0.0931	0.1692	0.2743	0.2606	0.3372	0.6088	0.0606	0.1254	0.2763
	Traj2SimVec^t	0.2065	0.4654	0.5821	0.0891	0.1573	0.2477	0.2383	0.2899	0.5299	0.2067	0.2921	0.4711
	T3S^t	0.2473	0.4994	0.5171	0.1876	0.2652	0.4729	0.2278	0.3098	0.4711	0.1217	0.2458	0.4608
	GTS^t	0.3191	0.4229	0.6467	0.2148	0.3538	0.5226	0.2878	0.3185	0.5562	0.1935	0.2746	0.4177
	ST2Vec	**0.3834**	**0.5051**	**0.7221**	**0.2576**	**0.3916**	**0.5953**	**0.3178**	**0.3942**	**0.7244**	**0.2117**	**0.2967**	**0.5117**

the corresponding ground-truth result; and Rk@t is the top-t recall for the top-k ground truth that captures the fraction of the top-k ground truth in the corresponding top-t result. The closer HR@10, HR@50, and R10@50 are to 1, the higher the model effectiveness.
Competitors/Baselines. We compare ST2Vec with all existing trajectory similarity learning methods, including NEUTRAJ [33], Traj2SimVec [37], T3S [32], and GTS [10], where GTS has state-of-the-art performance. Since all existing competitors ignore the temporal aspect and cannot handle temporal embedding, to enable fair comparisons, we extend them (i.e., integrating them with the two alternative methods proposed in Sec. 4.2 and our designed TMM) under SF strategy, resulting in 12 baselines in three categories. By doing this, we can evaluate the effectiveness of TMM.

- **Window-guided baselines ($*^w$):** Here, we assign trajectories to time slots and perform top-k queries in each slot, resulting in NEUTRAJ^w, Traj2SimVec^w, T3S^w, and GTS^w.
- **LSTM-guided baselines ($*^l$):** Here, we feed temporal trajectories directly to an LSTM model, resulting in NEUTRAJ^l, Traj2SimVec^l, T3S^l, and GTS^l.
- **Our TMM-guided baselines ($*^t$):** Here, we integrate our temporal trajectory representation module (i.e., TMM) into the all existing spatially oriented competitors, resulting in NEUTRAJ^t, Traj2SimVec^t, T3S^t, and GTS^t.

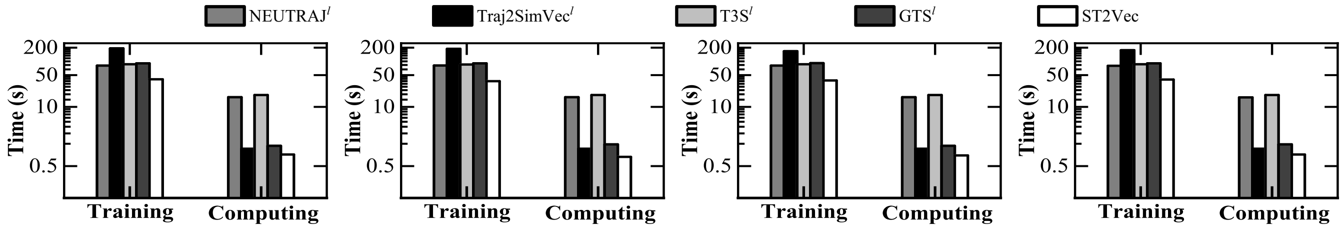
The symbols w, l, and t indicate the categories of baselines.

Hyperparameters. We split each data set into training, validation, and test sets with ratios 3:1:6. The default value of λ is set to 0.5. We set the spatial and temporal embedding dimensionalities to 128. The number of hidden LSTM units is 128. We set the batch size to 50. We tune parameters to obtain the best performance. Moreover, we use Adam [11] for training with an initial learning rate of 0.001. Finally, ST2Vec is implemented in Python and Pytorch. All experiments are conducted on a server with an Intel Silver 4210R, 2.40GHz CPU, 64-GB RAM, and a GeForce GTX-2080 Ti 11G GPU. Code is available online[4] for further studies.

6.2 Comparison with Existing Models

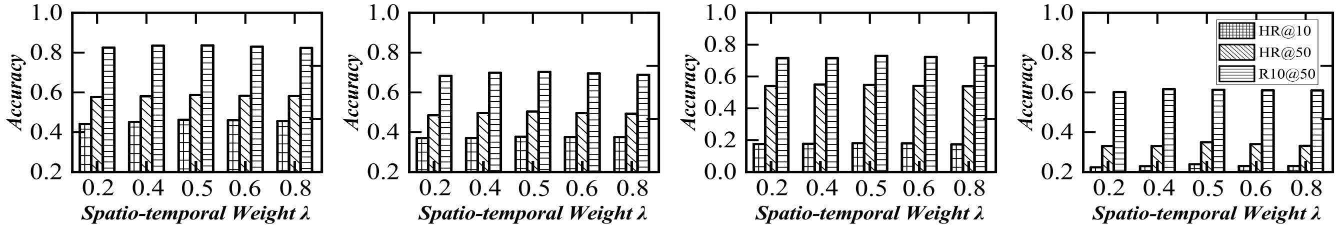
6.2.1 Effectiveness Comparison. To investigate model (i.e., similarity learning) effectiveness, we compare the performance of ST2Vec and all 12 baselines. Table 1 lists the results. From these results, we provide observations and analyses as follows. i) We first observe that, the TMM-guided baselines significantly outperform the window-guided and LSTM-guided baselines, indicating that the proposed TMM is effective for temporal representation learning. Indeed, although the window-based and LTSM-based methods might capture the temporal information to some extent, they ignore the continuous nature of time and periodic patterns, restricting their effectiveness. ii) Second, within the same category, GTS and ST2Vec

[4] https://github.com/ZJU-DAILY/ST2Vec



(a) Beijing/TP (b) Beijing/DITA (c) Beijing/LCRS (d) Beijing/NetERP

Figure 2: Model Efficiency Evaluation: Offline Model Training and Online Similarity Computing



(a) Beijing/TP (b) Beijing/DITA (c) Beijing/LCRS (d) Beijing/NetERP

Figure 3: Performance of ST2Vec vs. Spatio-Temporal Weight λ

outperform the other methods. The main reason is that GTS and ST2Vec consider the road network in spatial correlation modeling, while the other methods only capture the sequence features in free space and disregard the structural dependencies in road networks. iii) Third, ST2Vec consistently achieves better accuracy than GTS. This reflects the fact that GTS targets POI-based trajectory similarity computation that disregards the actual travel paths.

6.2.2 Efficiency Comparison. Next, we study model efficiency in terms of offline model training (denoted as training, with the unit seconds/epoch) and online computing (denoted as computing, with the unit seconds/4k trajectories). Fig. 2 shows results on Beijing. Note that the scale of the y-axis is logarithmic due to the significant differences. We compare ST2Vec with the LSTM-guided baselines, i.e., NEUTRAJ^l, Traj2SimVec^l, T3S^l, and GTS^l, because they outperform the window-guided baselines and because the TMM-guided baselines are essentially based on our designed TMM. As can be seen, ST2Vec outperforms the competitors at both training and computing. During the training phase, ST2Vec finishes each epoch within 40 seconds and runs two times faster than NEUTRAJ, T3S, and GTS, and five times faster than Traj2SimVec. In terms of similarity computation, we measure the total running time of each method on the test data. Here, ST2Vec also exhibits superior performance (i.e., within 1 second) and is 20 times faster than NEUTRAJ and T3S and two times faster than Traj2Sim and GTS. The reason is that ST2Vec adopts unified fusion strategy while the competitors adopt separate fusion strategy .

6.3 Model Robustness Study

Further, we evaluate ST2Vec performance when varying the spatio-temporal weight λ in Eq. 1. When $\lambda = 1$, only the spatial domain is considered, and when $\lambda = 0$, the similar computation considers the temporal domain only. We report results for Beijing only; Rome yields similar observations. Fig. 3 shows that HR@10, HR@50, and R10@50 are stable when varying λ, indicating that ST2Vec is able to support a variety of applications discussed in Section 1. In addition, we give experiments to study the effect of training data size on the performance (Fig. 7 in Appendix B).

6.4 Effectiveness of Key Designs

6.4.1 ST2Vec vs. Fusion Strategy. To evaluate the effect of the fusion strategy, we train ST2Vec using separate fusion (SF) and unified fusion (UF). Fig. 5 shows that ST2Vec using UF achieves similar effectiveness to that of using SF. However, ST2Vec-UF achieves fast convergence than ST2Vec-SF, as SF features two separate LSTMs, which doubles the number of parameters to tune.

6.4.2 ST2Vec vs. Curriculum/Random. To evaluate the effect of curriculum, we perform TP similarity learning using curriculum and random strategies. Fig. 6 shows the use of curriculum yields faster convergence and higher quality (i.e., HR@50) than the use of random. This is because the curriculum strategy feeds training samples directionally, which is effective for neural network learning.

6.5 Case Study

We proceed to perform top-k querying and DBSCAN clustering [8] using Beijing to examine the capabilities of ST2Vec. In terms of top-k querying, we randomly choose one trajectory as the query trajectory. Then we plot the top-2 ground-truth trajectories according to TP as well as the top-2 trajectories returned by ST2Vec. The left part of Fig. 4 shows that the trajectories returned by ST2Vec match the ground-truth trajectories very well. Next, we perform DBSCAN clustering when fixing parameter *minPts* at 10 and varying parameter ϵ. We compare the clustering results generated by TP and ST2Vec. As shown in Fig. 4, the numbers of clusters in the two results share similar trends as ϵ grows.

7 CONCLUSIONS

This is the first study of time-aware representation for spatio-temporal similarity learning in road networks. In future research, it is of interest to extend ST2Vec to support similarity range queries.

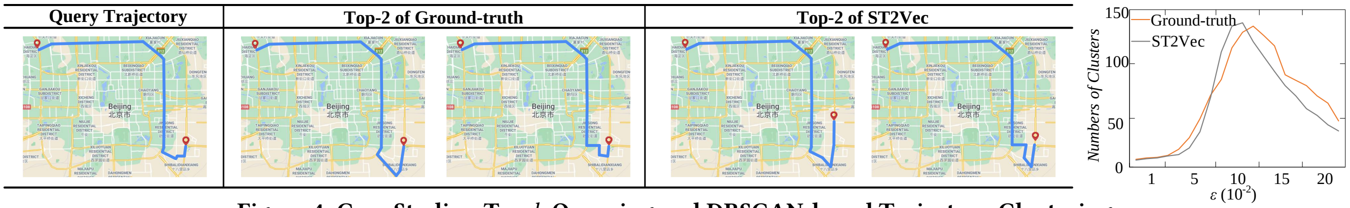
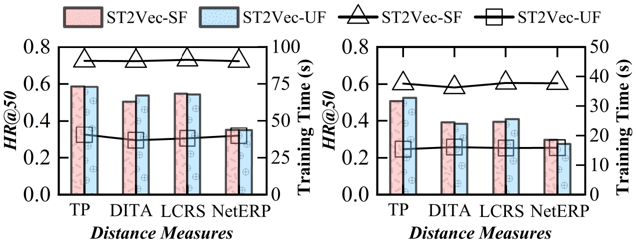
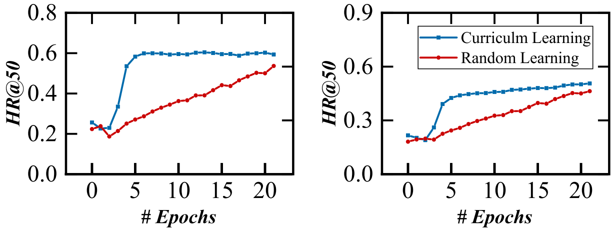


Figure 4: Case Studies: Top-k Querying and DBSCAN-based Trajectory Clustering



(a) Beijing (b) Rome

Figure 5: ST2Vec Performance vs. Fusion Strategy



(a) Beijing (b) Rome

Figure 6: Convergence of ST2Vec

ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2021YFC3300303, the NSFC under Grants No. (62025206, 61972338, and 62102351). Yunjun Gao is the corresponding author of the work.

REFERENCES

[1] Stefan Atev, Grant Miller, and Nikolaos P. Papanikolopoulos. 2010. Clustering of Vehicle Trajectories. *TITS* 11, 3 (2010), 647–657.

[2] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On Map-Matching Vehicle Tracking Data. In *VLDB*. 853–864.

[3] Antonia Breuer, Sven Elflein, Tim Joseph, Jan-Aike Bolte, Silviu Homoceanu, and Tim Fingscheidt. 2019. Analysis of the Effect of Various Input Representations for LSTM-Based Trajectory Prediction. In *ITSC*. 2728–2735.

[4] Ernie Chang, Hui-Syuan Yeh, and Vera Demberg. 2021. Does the Order of Training Samples Matter? Improving Neural Data-to-Text Generation with Curriculum Learning. *CoRR* abs/2102.03554 (2021).

[5] Lei Chen and Raymond T. Ng. 2004. On The Marriage of Lp-norms and Edit Distance. In *VLDB*. 792–803.

[6] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *SIGMOD*. 491–502.

[7] Roniel S. de Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. 2020. Vehicle Trajectory Similarity: Models, Methods, and Applications. *ACM Comput. Surv.* 53, 5 (2020), 94:1–94:32.

[8] Junhao Gan and Yufei Tao. 2015. DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation. In *SIGMOD*. 519–530.

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. 855–864.

[10] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. 2021. A graph approach for trajectory similarity computation in networks. In *KDD*. 556–564.

[11] Diederik P. Kingma. 2015. A method for stochastic pptimization. In *ICLR*.

[12] Satoshi Koide, Chuan Xiao, and Yoshiharu Ishikawa. 2020. Fast Subtrajectory Similarity Search in Road Networks under Weighted Edit Distance Constraints. *VLDB* 13, 11 (2020), 2188–2201.

[13] Xiangjie Kong, Menglin Li, Jianxin Li, Kaiqi Tian, Xiping Hu, and Feng Xia. 2019. CoPFun: an urban co-occurrence pattern mining scheme based on regional function discovery. *WWW* 22, 3 (2019), 1029–1054.

[14] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *ICDE*. 617–628.

[15] An Liu, Yifan Zhang, Xiangliang Zhang, Guanfeng Liu, Yanan Zhang, Zhixu Li, Lei Zhao, Qing Li, and Xiaofang Zhou. 2020. Representation learning with multi-level attention for activity trajectory similarity computation. *TKDE* (2020).

[16] Xuebo Liu, Houtim Lai, Derek F. Wong, and Lidia S. Chao. 2020. Norm-based curriculum learning for neural machine translation. In *ACL*. 427–436.

[17] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. 2021. Zone pAth Construction (ZAC) based Approaches for Effective Real-Time Ridesharing. *J. Artif. Intell. Res.* 70 (2021), 119–167.

[18] Guang Xing Lye, Wai-Khuen Cheng, Teik-Boon Tan, Chen-Wei Hung, and Yen-Lin Chen. 2020. Creating Personalized Recommendations in a Smart Community by Performing User Trajectory Analysis through Social Internet of Things Deployment. *Sensors* 20, 7 (2020), 2098.

[19] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. 815–823.

[20] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, and Kai Zheng. 2018. Trajectory similarity joins in spatial networks. *VLDB J.* 27, 3 (2018), 395–420.

[21] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Kai Zheng, and Panos Kalnis. 2017. Trajectory similarity join in spatial networks. In *VLDB*. 1178–1189.

[22] Shuo Shang, Ruogu Ding, Kai Zheng, Christian S. Jensen, Panos Kalnis, and Xiaofang Zhou. 2014. Personalized trajectory matching in spatial networks. *VLDB J.* 23, 3 (2014), 449–468.

[23] Shuo Shang, Kai Zheng, Christian S. Jensen, Bin Yang, Panos Kalnis, Guohe Li, and Ji-Rong Wen. 2015. Discovery of Path Nearby Clusters in Spatial Networks. *TKDE* 27, 6 (2015), 1505–1518.

[24] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *SIGMOD*. 725–740.

[25] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory measures and evaluation. *VLDB J.* 29, 1 (2020), 3–32.

[26] Luan Tran, Minyoung Mun, Matthew Lim, Jonah Yamato, Nathan Huh, and Cyrus Shahabi. 2020. DeepTRANS: A Deep Learning System for Public Bus Travel Time Estimation using Traffic Forecasting. *VLDB J.* 13, 12 (2020), 2957–2960.

[27] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. 2002. Discovering Similar Multidimensional Trajectories. In *ICDE*. 673–684.

[28] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On Position Embeddings in BERT. In *ICLR*.

[29] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast Large-Scale Trajectory Clustering. *VLDB* 13, 1 (2019), 29–42.

[30] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Zizhe Xie, Qizhi Liu, and Xiaolin Qin. 2018. Torch: A Search Engine for Trajectory Data. In *SIGIR*. 535–544.

[31] Dong Xie, Feifei Li, and Jeff M. Phillips. 2017. Distributed Trajectory Similarity Search. *VLDB* 10, 11 (2017), 1478–1489.

[32] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *ICDE*. 2183–2188.

[33] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. A generic seed-guided neural metric learning approach. In *ICDE*. 1358–1369.

[34] Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. 1998. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *ICDE*. 201–208.

[35] Qingying Yu, Yonglong Luo, Chuanming Chen, and Xiaoyao Zheng. 2019. Road congestion detection based on trajectory clustering. *ISPRS* 8, 6 (2019), 264.

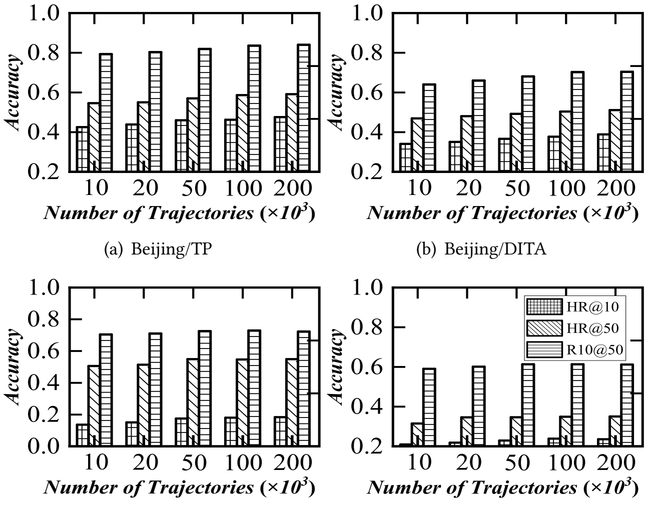
[36] Haitao Yuan and Guoliang Li. 2019. Distributed In-memory Trajectory Similarity Search and Join on Road Network. In *ICDE*. 1262–1273.

[37] Hanyuan Zhang, Xinyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. 2020. Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching. In *IJCAI*. 3209–3215.

[38] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. In *AAAI*. 1234–1241.

Table 2: Time Cost of Online Similarity Search on Beijing

Measures	Methods	1k	5k	10k	200k
TP	Non-learning	1.492s	3.127s	5.893s	117.832s
	ST2Vec	0.004s	0.014s	0.028s	0.521s
DITA	Non-learning	0.921s	3.301s	6.291s	125.826s
	ST2Vec	0.004s	0.015s	0.028s	0.522s
LCRS	Non-learning	1.292s	4.614s	8.784s	175.824s
	ST2Vec	0.004s	0.014s	0.028s	0.525s
NetERP	Non-learning	1.535s	6.246s	12.674s	253.481s
	ST2Vec	0.004s	0.015s	0.028s	0.522s



(a) Beijing/TP (b) Beijing/DITA

(c) Beijing/LCRS (d) Beijing/NetERP

Figure 7: Performance of ST2Vec vs. Training Data Size

A APPROACH ANALYSES

In this section, we compare the time complexity of traditional similarity computation methods and our learning-based ST2Vec approach, when evaluating the similarity between two trajectories.

A.1 Traditional Similarity Measures

Free-space measures. To evaluate the similarity between two trajectories, many metrics are proposed in previous studies, such as Hausdorff [1], Dynamic Time Warping (DTW) [34], longest common subsequence (LCSS) [27], and edit distance with real penalty (ERP) [5]. Specifically, they typically rely on pointwise matching computation [10, 14], meaning that they need to scan all point pairs from two trajectories to calculate the inter-trajectory similarity scores, which incurs quadratic time complexity $O(\hat{n}^2)$, where $\hat{n}$ is the average trajectory length (i.e., the number of sample points). Note that, these similarity metrics are designed for free-space based trajectory similarity computation, and thus they are widely used for bird, storm, and vessel trajectory analyses. In many real application scenarios, objects (e.g., people and vehicles) are moving in spatial networks rather than in Euclidean space. In a spatial network, Euclidean distance might lead to errors when calculating the real distance between objects. This paper focuses on road-network based similarity learning, although it is straightforward to train ST2Vec for free-space similarity learning.

Network-aware metrics. Also, many studies are proposed to extend these free-space similarity metrics into road-network aware similarity metrics, resulting in TP [21] (an extension to Hausdorff), DITA [24] (an extension to DTW), LORS [30] and LCRS [36] (extensions to LCSS), and NetERP [12] (an extension to ERP). To compute the similarity between two trajectories in a road network, the time complexity of TP, DITA, and NetERP is $O((E + L \lg L) \cdot \hat{m}^2)$, where $O(E + L \lg L)$ is the cost of finding a shortest path between two vertices and $\hat{m}$ is the average trajectory length. E and L are defined in Definition 3.1. The complexity of LORS and LCRS using the overlap between two trajectory paths as the similarity scores is $O(\hat{m}^2)$, which computation relies on dynamic programming. Although speedup techniques such as indexing and pruning [29, 30], approximation [6, 21], and distributed processing [24, 36] exist, these performance enhancements are specific to particular similarity notions. As a result, they cannot be adapted easily to other similarity notions. Rather, new similarity notions always call for the invention of new performance enhancing techniques.

A.2 Learning-based ST2Vec

Once ST2Vec is well-trained on a dataset using few training samples (Fig. 7), it enables i) computing the similarity between two trajectories within $O(d)$, where d is a constant dimension; ii) measuring the similarity relations even the dataset may update in the future. Thus, ST2Vec is efficient for large-scale trajectory data analysis, as trajectories can be embedded into low-dimensional vectors.

B MORE EXPERIMENTAL RESULTS

B.1 Acceleration Study

Following previous studies [32, 33], we give an acceleration study to show the speedup capability of ST2Vec. We perform top-50 similarity querying for each trajectory using ST2Vec and a traditional method. Table 2 reports the average time cost with different data sizes. As observed, ST2Vec achieves 200–400x speeds up over the non-learning based method. Although ST2Vec requires model training time, it is acceptable (Fig. 2) and this process need only once.

B.2 Sensitivity to Training Volumes

Second, we investigate the effect of the number of training samples on the performance of ST2Vec, to prove the training cost is acceptable in real-world applications. Fig. 7 shows the similarity learning performance (i.e., HR@10, HR@50, R10@50) for the four measures when varying the training data size from 10k to 200k. As can be observed, ST2Vec exhibits stable performance. That is, ST2Vce can be trained using few training samples, which is also lightweight (Fig. 2). Even if given a new trajectory dataset of a new city, users only need to select part of the trajectories for model training.

C DBSCAN CLUSTERING

Trajectory clustering is an essential problem in trajectory analyses, which groups similar trajectories and distinguishs dissimilar trajectories. DBSCAN stands for density-based spatial clustering with noise. There are two parameters: *minPts* denotes the minimum number of trajectories to define a cluster; and ϵ denotes the distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them are less than or equal to ϵ.