TrajAgg: Dual-Scale Feature Aggregation with Hybrid Training for Trajectory Similarity Computation in Free Space

Xiao Zhang[1], Xingyu Zhao[1], Yuan Cao[1], Bin Wang[1], Guiyuan Jiang[1], Yanwei Yu[1,2*]

[1]Faculty of Information Science and Engineering, Ocean University of China, Qingdao, Shandong, China
[2]State Key Laboratory of Physical Oceanography, Ocean University of China, Qingdao, Shandong, China
{xiaozhang4549, zhaoxingyu}@stu.ouc.edu.cn, {cy8661, wangbin9545, jiangguiyuan, yuyanwei}@ouc.edu.cn

Abstract

With the widespread use of location-tracking technologies, large volumes of trajectory data are continuously generated. Trajectory similarity computation is a core task in trajectory mining with broad applications. However, existing methods still face two key challenges: (1) difficulty in balancing efficiency and representation quality, and (2) reliance on a single training paradigm, which limits the ability to capture both pairwise similarity and batch-level coherence. To address the challenges mentioned above, we propose a trajectory similarity computation framework, named TrajAgg. Specifically, our framework incorporates a novel Aggregation Transformer that efficiently aggregates GPS and grid features through two stages of direct interaction and enhances the expressiveness of the resulting trajectory embeddings. In addition, by integrating two distinct training paradigms, our model captures both fine-grained pairwise relationships and global structural consistency. We further analyze its effectiveness from the perspective of mutual information. Extensive experiments on three publicly available datasets show that TrajAggconsistently outperforms state-of-the-art baselines. Our method achieves average improvements of 15.11%, 16.49%, 10.41% and 40.15% in HR@1 under four distance measures across three datasets, respectively.

Code — https://github.com/zhx66741/TrajAgg

Introduction

With the widespread use of GPS-enabled devices, vast amounts of trajectory data are generated, each capturing the spatiotemporal movement and dynamic behavior of an object over time and space. Accurately measuring trajectory similarity is fundamental to various practical applications (Chen et al. 2015; Fang et al. 2022; Ma et al. 2024), including trajectory clustering (Lee, Han, and Whang 2007; Wang et al. 2019), anomaly detection (Chen et al. 2021; Zhang et al. 2023), route planning (Yan et al. 2025), and trajectory matching (Zhang et al. 2025; Xia et al. 2025).

Many heuristic trajectory similarity metrics, such as Hausdorff (Alt 2009), Fréchet (Alt and Godau 1995), and EDR (Chen, Özsu, and Oria 2005), adopt point-matching strategies and rely on dynamic programming to compute optimal alignments (Wu et al. 2020). However, this approach incurs a quadratic time complexity of $O(n^2)$ or even higher, where n denotes the average trajectory length, making it inefficient for large-scale trajectory datasets. To address this limitation, recent studies (Li et al. 2018; Yao et al. 2019; Yang et al. 2021; Jiang et al. 2023; Yang et al. 2024) have shifted toward learning-based methods, in which each trajectory is embedded into a d-dimensional embedding by deep neural models, allowing trajectory similarity to be computed efficiently in linear time through simple vector operations.

Existing learning-based methods can be categorized into two groups: ***Pairwise Learning*** and ***Embedding Learning*** (Si et al. 2025). Pairwise learning approaches, such as SIMformer (Yang et al. 2024) and TMN (Yang et al. 2022), typically take two trajectories as input and either utilize neural networks to directly model their similarity or apply a shared encoder to generate embeddings for two trajectories, followed by computing similarity between the two embeddings. In contrast, embedding learning methods, such as TrajCL (Chang et al. 2023), encode a batch of trajectories into embeddings and compute the similarity between all pairs of trajectory embeddings. The former focuses on local matching relationships between trajectories, while the latter emphasizes the global distribution patterns of trajectories.

Although existing methods have made some progress in trajectory similarity computation, they still suffer from two limitations: **(1) Existing trajectory encoders face the trade-off between expressive power and computational efficiency** as they either adopt complex architectures to capture comprehensive movement patterns in trajectories, or compromise representational capacity for the sake of lightweight design. For example, TrajCL employs two separate TransformerEncoders to independently encode structural and spatial information, followed by feature fusion through the attention mask. While this design improves performance, it introduces computational overhead, and the fusion strategy limits effective interaction between dual-scale representations. In addition, SIMformer adopts a single-layer TransformerEncoder to encode trajectories and leverages the Chebyshev distance to compute the distance between embeddings. However, because it relies solely on a single-layer TransformerEncoder, its architecture is simplistic and limits its capacity to model complex spatial-temporal

*Corresponding author.

dependencies in trajectories.

(2) Existing single-paradigm training frameworks limit the model's ability to jointly capture both pairwise similarity and batch-level coherence. Pairwise learning focuses exclusively on local discriminative power by optimizing individual anchor-positive/negative pairs. While effective for fine-grained similarity judgments, this paradigm inherently ignores higher-order relationships and global data topology. Embedding learning, conversely, ensures global consistency by optimizing the complete similarity matrix. While effectively capturing the overall distribution characteristics, this optimization approach often compromises precision in fine-grained pairwise similarity assessment.

To address these limitations, we propose TrajAgg, which integrates a single-layer aggregation transformer to generate highly expressive embeddings by enabling direct interaction between dual-scale trajectory features, while maintaining a lightweight and simple architecture. Meanwhile, by jointly training with both pairwise and embedding learning paradigms, we integrate pairwise similarity information and batch-level coherence into the model training process. Finally, we conducted extensive experiments on three public datasets to validate the effectiveness of our method. Our approach achieved an average improvement of 21.91% in HR@1 on Porto across four metrics. In summary, our main contributions are as follows:

- We propose a novel trajectory encoder, aggregation transformer, that employs dual cross-attention mechanisms to enable direct interaction between dual-scale information, achieving high-quality trajectory embeddings while maintaining efficient encoding.
- We integrate pairwise learning and embedding learning to enable the model to simultaneously ensure local matching accuracy and global structural consistency, and explain from the perspective of mutual information.
- We conduct extensive experimental evaluations on three real-world datasets. Experimental results show that our model achieves an average performance improvement of 22.01% in HR@1 on GeoLife across four metrics.

Related Work

Attention-based Methods. T3S (Yang et al. 2021) employs a self-attention-based network to learn the overall structural information of trajectories and combines it with sequential information to obtain trajectory embeddings. TrajGAT (Yao et al. 2022) addresses the limitations of traditional RNN and CNN architectures in capturing long-range dependencies and spatial structures by introducing a Graph Attention Network (GAT) (Veličković et al. 2017). TrajCL (Chang et al. 2023) adopts a contrastive learning framework combined with a dual-feature multi-head self-attention mechanism to learn both structural and spatial characteristics of trajectories. SIMformer (Yang et al. 2024) utilizes a single-layer vanilla TransformerEncoder to learn trajectory embeddings and integrates Chebyshev distance to effectively enhance the performance of similar trajectory retrieval.

RNN-based Methods. NeuTraj (Yao et al. 2019) designs a Memory-Augmented RNN Unit that generates effective trajectory embeddings by memorizing and retrieving information from previously processed trajectories as well as from nearby geographic regions. Traj2SimVec (Zhang et al. 2020) employs a bidirectional LSTM to encode the sequence of point embeddings within a trajectory and aggregates the outputs using mean pooling to obtain the overall trajectory representation. GRLSTM (Zhou et al. 2023) employs a multi-layer LSTM with residual connections to effectively alleviate the gradient vanishing problem without significantly increasing training time. KGTS (Chen et al. 2024) employs a GRU (Cho et al. 2014) to model the sequential order of grids within trajectories, capturing temporal features and producing the final trajectory embedding.

Preliminaries

Definition 1 (Trajectory) *A trajectory is a sequence of spatiotemporal points $\mathcal{T} = [p_1, p_2, \ldots, p_n]$, where each point $p_i = (lon_i, lat_i)$ is a geographic coordinate, and n is the total number of points.*

Definition 2 (Grid partition and Grid Sequence) *Grid Partitioning divides a geographic area into uniform, non-overlapping grids of a given size. Trajectory Grid Sequence maps each trajectory point p_i to its corresponding grid, forming $\mathcal{T}_G = [g_1, g_2, \ldots, g_n]$, where g_i is the grid's 2D coordinate in the x-y plane.*

Problem 1 (Trajectory Similarity Computation) *Given a heuristic metric $d(\cdot,\cdot)$, a neural network $f_\theta()$, and two trajectories $\mathcal{T}_i, \mathcal{T}_j$, our goal is to minimize the discrepancy between true similarity and predicted similarity:*

$$\underset{\theta}{\operatorname{argmin}} \left| \ell\big[f_\theta(\mathcal{T}_i),\ f_\theta(\mathcal{T}_j)\big] - d(\mathcal{T}_i, \mathcal{T}_j) \right|, \tag{1}$$

where $\ell[\cdot]$ denotes the distance between embeddings.

Methodology

As illustrated in Figure. 1, TrajAggtraining process consists of four components: (1) *Trajectory Preparation*, (2) *Trajectory Feature Extraction*, (3) *Dual-branch Trajectory Encoding*, and (4) *Hybrid Training*.

Trajectory Preparation and Feature Extraction.

Given a trajectory dataset $\mathscr{T}$ with M trajectories and the corresponding geographic area, as illustrated in Figure. 1(a) and (b), we randomly sample N trajectories as anchors. For each anchor, we select k target trajectories, resulting in $B = N \times k$ trajectories in total. We then partition the geographic area into uniform grids based on a given cell size and convert each GPS sequence into a corresponding grid sequence through grid mapping.

Subsequently, we apply padding to both the GPS sequences and grid sequences of the anchors and targets to obtain $\mathbf{T}^a_{\text{gps}}, \mathbf{T}^a_{\text{grid}}, \mathbf{T}^t_{\text{gps}}, \mathbf{T}^t_{\text{grid}} \in \mathbb{R}^{B\times n\times 2}$ and $\mathcal{M} \in \mathbb{R}^{B\times n}$, where n represents the padded length (i.e., the length of the longest trajectory in the batch), and $\mathcal{M}$ is the padding mask indicating the valid positions in each trajectory sequence.

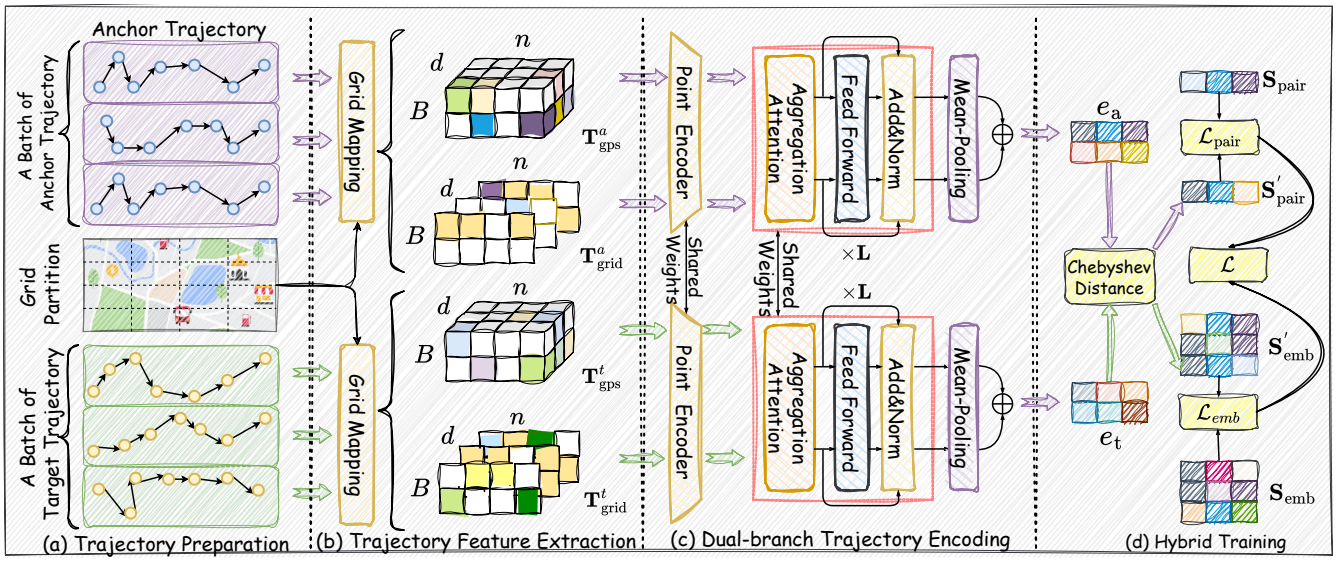


Figure 1: Overall framework of the proposed TrajAgg.

Dual-branch Trajectory Encoding

Point Encoder. Before applying the aggregation transformer, we first perform point embedding to project the raw GPS and grid features into the embedding space. We choose LSTM (Hochreiter and Schmidhuber 1997) as point encoder because it effectively captures the sequential dependencies in trajectories, which is crucial for heuristic metrics that rely on recursive computations based on previous positions. The process is formulated as follows:

$$\begin{aligned}\mathbf{H}^a_{\text{gps}} &= [\mathbf{h}^1_{\text{gps}}, \mathbf{h}^2_{\text{gps}}, \ldots, \mathbf{h}^n_{\text{gps}}] = \text{LSTM}(\mathbf{T}^a_{\text{gps}}),\\ \mathbf{H}^a_{\text{grid}} &= [\mathbf{h}^1_{\text{grid}}, \mathbf{h}^2_{\text{grid}}, \ldots, \mathbf{h}^n_{\text{grid}}] = \text{LSTM}(\mathbf{T}^a_{\text{grid}}),\end{aligned} \tag{2}$$

where $\mathbf{H}^a_{\text{gps}}$ and $\mathbf{H}^a_{\text{grid}} \in \mathbb{R}^{B\times n\times d}$ denote the output feature matrices of LSTM, which are composed of the hidden states $\mathbf{h}^i_{\text{gps}}$ and $\mathbf{h}^i_{\text{grid}}$ at i-th time step. Similarly, $\mathbf{H}^t_{\text{gps}}$ and $\mathbf{H}^t_{\text{grid}} \in \mathbb{R}^{B\times n\times d}$ can be obtained using the same equations.

Aggregation Transformer. As illustrated in Figure. 2, each aggregation transformer layer comprises an aggregation attention module followed by a feed-forward network with residual connections.

The aggregation attention applies cross-attention twice to model the dependencies between coarse-grained grid features and fine-grained GPS features, thereby integrating complementary information for more expressive representations. In the following, we use $\mathbf{H}^t_{\text{gps}}$ and $\mathbf{H}^t_{\text{grid}}$ as illustrative examples to demonstrate the working mechanism of aggregation attention.

Specifically, we treat $\mathbf{H}^t_{\text{gps}}$ as the query and $\mathbf{H}^t_{\text{grid}}$ as the key and value. Linear projections are then applied to obtain the matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, representing the transformed query, key, and value, respectively, as shown below:

$$\begin{aligned}\mathbf{Q} &= \mathbf{W}_q \cdot \mathbf{H}^t_{\text{gps}} + b_q,\\ \mathbf{K} &= \mathbf{W}_k \cdot \mathbf{H}^t_{\text{grid}} + b_k,\\ \mathbf{V} &= \mathbf{W}_v \cdot \mathbf{H}^t_{\text{grid}} + b_v,\end{aligned} \tag{3}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ are learnable weight matrices and b_q, b_k, b_v are the corresponding bias vectors.

Subsequently, the matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are used in scaled multi-head dot-product attention. For each attention head i, the attention scores are first computed using $\mathbf{Q}_i$ and $\mathbf{K}_i$, which represent the query and key projections for the i-th head. The padding mask $\mathcal{M}$ is then applied to suppress the influence of padded positions, and the resulting masked scores are subsequently used to weight the corresponding value matrix $\mathbf{V}_i$, producing the head-specific output $\mathbf{E}^t_i$. Finally, outputs from all heads are concatenated to produce the cross-attention output $\mathbf{E}^t_{\text{grid}}$. This is computed as follows:

$$\begin{aligned}\mathbf{E}^t_i &= \left(\text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_k}}\right)\odot\mathcal{M}\right)\mathbf{V}_i,\\ \mathbf{E}^t_{\text{grid}} &= \text{Concat}(\mathbf{E}^t_1, \ldots, \mathbf{E}^t_h),\end{aligned} \tag{4}$$

where h denotes the number of attention heads, and d_k is the dimension of each head.

The above operation completes the first of the two cross-attention, where the output $\mathbf{E}^t_{\text{grid}}$ primarily captures coarse-grained structural characteristics. To further incorporate fine-grained information from GPS side, we perform an additional cross-attention in the reverse direction. Specifically, we use $\mathbf{H}^t_{\text{gps}}$ as the key and value, and take $\mathbf{H}^t_{\text{grid}}$ as the query. This setup enables the resulting embedding to capture richer detailed information. Following the same computation steps as in Eqs. (3) and (4), we obtain the output $\mathbf{E}^t_{\text{gps}}$.

Then, we apply mean pooling to $\mathbf{E}^t_{\text{gps}}$ and $\mathbf{E}^t_{\text{grid}}$ separately, while masking out the padding positions using $\mathcal{M}$ to ensure

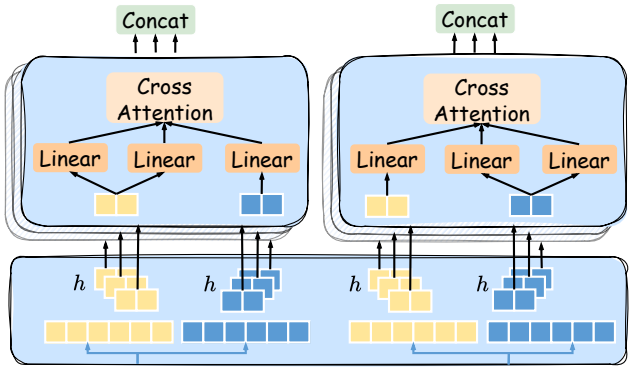


Figure 2: Aggregation Transformer.

robustness. This yields the GPS feature embedding $\mathbf{e}^t_{\text{gps}} \in \mathbb{R}^{B\times d}$ and the grid feature embedding $\mathbf{e}^t_{\text{grid}} \in \mathbb{R}^{B\times d}$ for the target trajectories, as formalized below:

$$\begin{aligned}\mathbf{e}^t_{\text{gps}} &= \text{Pooling}(\mathbf{E}^t_{\text{gps}}, \mathcal{M}),\\ \mathbf{e}^t_{\text{grid}} &= \text{Pooling}(\mathbf{E}^t_{\text{grid}}, \mathcal{M}).\end{aligned} \tag{5}$$

The final target trajectories embedding $\mathbf{e}^t$ is obtained through a weighted combination of $\mathbf{e}^t_{\text{gps}}$ and $\mathbf{e}^t_{\text{grid}}$, governed by the hyperparameter μ:

$$\mathbf{e}^t = \mu \cdot \mathbf{e}^t_{\text{grid}} + (1-\mu)\cdot \mathbf{e}^t_{\text{gps}}. \tag{6}$$

Similarly, following the above procedure, we can also obtain the corresponding embeddings for the anchor trajectories, denoted as $\mathbf{e}^a \in \mathbb{R}^{B\times d}$.

The aggregation transformer enables effective communication between fine-grained and coarse-grained representations, allowing the model to capture complex spatial-temporal dependencies and integrate dual-scale trajectory information in a unified manner.

Hybrid Training.

Hybrid Training Workflow. First, for the given trajectory dataset $\mathscr{T}$ containing M trajectories, we compute ground-truth distance matrix $\mathbf{D} \in \mathbb{R}^{M\times M}$ using heuristic metrics such as Hausdorff, DTW, and others. This distance matrix is then transformed into a normalized similarity matrix:

$$\mathbf{S} = \exp(-\alpha \cdot \mathbf{D}), \tag{7}$$

where α is a scaling factor. We then extract the ground-truth supervision for pairwise learning and embedding learning from $\mathbf{S}$, denoted as $\mathbf{S}_{\text{pair}}$ and $\mathbf{S}_{\text{emb}}$, respectively.

Next, as shown in Figure. 1(d), given the anchor trajectory embedding $\mathbf{e}^a$ and the target trajectory embedding $\mathbf{e}^t$, we compute the predicted similarity based on the selected heuristic metric. Specifically, following SIMformer, we adopt the Chebyshev distance for the Hausdorff, Discrete Fréchet, and SSPD metrics, and the Manhattan distance for DTW. The corresponding formulations are as follows:

$$\mathbf{S}'_{\text{pair}}[i] = \begin{cases}\exp\left(-\max\limits_{j=1}^{d}\left|(e^a)_{i,j} - (e^t)_{i,j}\right|\right)\\ \exp\left(-\sum\limits_{j=1}^{d}\left|(e^a)_{i,j} - (e^t)_{i,j}\right|\right)\end{cases} \tag{8}$$

$$\mathbf{S}'_{\text{emb}}[i,j] = \begin{cases}\exp\left(-\max\limits_{k=1}^{d}\left|(e^t)_{i,k} - (e^t)_{j,k}\right|\right)\\ \exp\left(-\sum\limits_{k=1}^{d}\left|(e^t)_{i,k} - (e^t)_{j,k}\right|\right)\end{cases} \tag{9}$$

where $i \in \{1,\ldots,B\}$ in Eq (8), $i,j \in \{1,\ldots,B\}$ in Eq (9) and $\mathbf{S}'_{\text{pair}}$ represents the predicted similarity in pairwise learning, while $\mathbf{S}'_{\text{emb}}$ denotes the predicted similarity in embedding learning.

Finally, we then compute the hybrid training losses using the Mean Squared Error (MSE) criterion:

$$\mathcal{L}_{\text{pair}} = \frac{1}{B}\sum_{i=1}^{B}\left(\mathbf{S}'_{\text{pair}}[i] - \mathbf{S}_{\text{pair}}[i]\right)^2, \tag{10}$$

$$\mathcal{L}_{\text{emb}} = \frac{1}{B(B-1)/2}\sum_{i=1}^{B}\sum_{j=i+1}^{B}\left(\mathbf{S}'_{\text{emb}}[i,j] - \mathbf{S}_{\text{emb}}[i,j]\right)^2, \tag{11}$$

$$\mathcal{L} = \epsilon_1 \cdot \mathcal{L}_{\text{pair}} + \epsilon_2 \cdot \mathcal{L}_{\text{emb}}, \tag{12}$$

where ϵ_1 and ϵ_2 are hyperparameters that balance the two loss terms. We empirically set both ϵ_1 and ϵ_2 to 1, as this configuration yields the best performance in our experiments.

The final hybrid loss $\mathcal{L}$ jointly optimizes both pairwise and embedding objectives. By jointly considering pairwise similarity and batch-level coherence, it provides more informative training signals and enhances the model's representation capability and generalization performance.

Mutual Information. Trajectory similarity computation can be viewed from the Information Theory by modeling the true similarity as a random variable $\mathbf{X}$ and the learned similarity as $\mathbf{Y}$. The goal is to align them by maximizing their mutual information $\mathbf{I}(\mathbf{X};\mathbf{Y})$.

a) Pairwise learning loss function Eq. (10) essentially corresponds to maximizing the conditional mutual information between the anchor and target embeddings given similarity $\mathbf{S}[i,j]$, which is defined as:

$$\mathbf{I}(e^i_{\text{a}}; e^j_{\text{t}} \mid \mathbf{S}[i,j]) = \mathbf{H}(e^i_{\text{a}} \mid \mathbf{S}[i,j]) - \mathbf{H}(e^i_{\text{a}} \mid e^j_{\text{t}}, \mathbf{S}[i,j]), \tag{13}$$

where e^i_{a} and e^j_{t} denote the embeddings of the i-th anchor and j-th target trajectories, respectively, and $\mathbf{S}[i,j]$ indicates their ground-truth similarity. $\mathbf{H}(e^i_{\text{a}} \mid \mathbf{S}[i,j])$ is the conditional entropy of the anchor embedding given the similarity label, and $\mathbf{H}(e^i_{\text{a}} \mid e^j_{\text{t}}, \mathbf{S}[i,j])$ is the conditional entropy of the anchor embedding given both the target embedding and the similarity label.

First, maximizing the discrimination term $\mathbf{H}(e^i_a \mid \mathbf{S}[i,j])$ encourages the distribution of e^i_a to be more diverse or dispersed under each given similarity label, which in turn drives

the model to retain trajectory features that are critical for similarity judgment, such as sharp turns or stay points.

Second, minimizing the conditional entropy term $\mathbf{H}(e_a^i \mid e_t^j, \mathbf{S}[i,j])$ means that the model is encouraged to accurately predict the anchor embedding e_a^i given the target embedding e_t^j and the similarity label $\mathbf{S}[i,j]$. This constraint drives the model to learn a *Lipschitz continuous mapping*, where trajectories that are similar in the original space are also mapped to nearby points in the embedding space:

$$\begin{aligned} \forall (e_a^i, e_t^j) \in \{(e_a, e_t) \mid \mathbf{S}[i,j] = 1\}, \\ |e_a^i - e_t^j| \leq \lambda |\mathcal{T}_a^i - \mathcal{T}_t^j|, \end{aligned} \tag{14}$$

where λ is the Lipschitz constant that controls the smoothness of the mapping function. This constraint ensures that similar trajectory pairs in the original space (e.g., $\mathcal{T}_a^i$ and $\mathcal{T}_t^j$) remain close in the embedding space, thereby preserving the local structural consistency.

b) Embedding Learning loss function Eq. (11) seeks to maximize the mutual information across the batch of learned representations, thereby preserving the global structural relationships among trajectories. This objective can be formally expressed as:

$$\mathbf{I}\left(\{e_t^i\}\right) = \sum_{i=1}^{B} \mathbf{H}(e_t^i) - \mathbf{H}(e_t^1, \ldots, e_t^B), \tag{15}$$

where $\mathbf{H}(e_t^i)$ denotes the marginal entropy of the i-th target trajectory embedding, and $\mathbf{H}(e_t^1, \ldots, e_t^B)$ represents the joint entropy of the entire batch.

First, maximizing the marginal entropy $\mathbf{H}(e_t^i)$ promotes the generation of informative and diverse embeddings by preventing over-concentration. Compared to maximizing the conditional term $\mathbf{H}(e_a^i \mid \mathbf{S}[i,j])$ in pairwise learning, the former enhances global expressiveness and prevents mode collapse by avoiding embedding concentration, while the latter focuses on maintaining the discriminability of anchor representations under different similarity labels to improve local precision.

Second, minimizing the joint entropy of trajectory embeddings naturally promotes the formation of compact and stable clusters for similar trajectories. Specifically, for each trajectory cluster C_k, the pairwise embedding distance satisfies the following upper bound:

$$\|e_t^i - e_t^j\| \leq \frac{2\mathbf{H}(C_k)}{\lambda_{\min}(J^\top J)}, \quad \forall \mathcal{T}_i, \mathcal{T}_j \in C_k, \tag{16}$$

where $\mathbf{H}(C_k)$ denotes the entropy of cluster C_k, and $\lambda_{\min}(J^\top J)$ is the minimum eigenvalue of the Jacobian-based local metric tensor. This inequality indicates that lower cluster entropy and well-conditioned Jacobians jointly lead to more compact embeddings. As a result, minimizing joint entropy not only mitigates mode collapse but also facilitates clearer structural separation in the embedding space, benefiting downstream tasks such as similarity learning and clustering.

Although both minimizing joint entropy and conditional entropy encourage similar trajectories to be mapped closer in the embedding space, they operate through different mechanisms. Joint entropy serves as a global unsupervised constraint that compresses the embedding distribution and enhances overall structural consistency, promoting the natural clustering of similar trajectories on a low-dimensional manifold. In contrast, conditional entropy provides label-driven local supervision, where the Lipschitz constraint enforces semantic alignment and improves local discriminability. By combining both objectives, the model can capture global structure and local semantics simultaneously, leading to more effective trajectory representations.

Experiments

Datasets

We evaluate our method on three publicly available datasets: Proto (O'Connell, moreiraMatias, and Kan 2015),GeoLife (Zheng et al. 2010) and T-Driver (Zheng 2011). In line with prior work (Chang et al. 2023; Yang et al. 2024), we remove trajectories that fall outside the geographic boundaries or do not meet the required length criteria. For detailed settings, please refer to Table 1. For each dataset, we randomly sample 10,000 trajectories and split them into train, eval, and test datasets in a ratio of 2:1:7.

Dataset	Porto	Geolife	T-Driver
Trajectories (n)	1,372,725	10,940	50,685
min length	20	20	20
max length	200	300	200
mean length	48	106	46
longitude	[-8.7005, -8.5192]	[116.25, 116.5]	[115.9, 117]
latitude	[41.1001, 41.2086]	[39.8, 40.1]	[39.6, 40.7]

Table 1: Detailed statistics of the dataset.

Baselines

We compare TrajAggwith seven representative baselines covering different trajectory modeling paradigms: NeuTraj (Yao et al. 2019), Traj2SimVec (Zhang et al. 2020), T3S (Yang et al. 2021), TMN (Yang et al. 2022), TrajGAT (Yao et al. 2022), TrajCL (Chang et al. 2023), and SIMformer (Yang et al. 2024), including LSTM-based, attention-based, graph-based, and contrastive pretraining approaches.

Evaluation Metrics and Experiment Settings

We use HR@1, HR@5, and HR@20 as evaluation metrics, and approximate the heuristic distances Hausdorff, Discrete Fréchet, SSPD, and DTW. The formula for HR@k is as follows:

$$\text{HR@}k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{|\text{Pred}_i^k \cap \text{GT}_i^k|}{k}, \tag{17}$$

where Q denotes the set of query trajectories, Pred_i^k is the set of top-k predicted trajectories for the i-th query, and GT_i^k or GT_i^t denotes the set of top-k or top-t ground-truth trajectories, respectively.

Dataset	Method	Hausdorff			Discrete Frećhet			SSPD			DTW		
		HR@1	HR@5	HR@20	HR@1	HR@5	HR@20	HR@1	HR@5	HR@20	HR@1	HR@5	HR@20
Porto	NeuTraj	29.91%	37.91%	50.18%	39.16%	46.31%	60.71%	24.83%	33.99%	45.07%	24.38%	30.01%	42.81%
	Traj2SimVec	35.71%	44.29%	55.72%	45.67%	49.83%	62.01%	23.13%	35.88%	45.70%	23.16%	35.76%	43.17%
	T3S	36.47%	50.01%	61.71%	47.98%	51.58%	64.73%	25.38%	34.91%	46.18%	22.18%	30.18%	44.19%
	TMN	39.39%	52.96%	63.16%	46.18%	52.78%	65.59%	26.98%	37.46%	46.19%	23.38%	33.78%	44.87%
	TrajGAT	42.76%	55.18%	63.56%	41.28%	47.62%	58.22%	25.09%	36,91%	45.91%	22.20%	31.38%	39.09%
	TrajCL	41.80%	53.10%	62.60%	43.10%	53.90%	62.70%	30.50%	40.60%	48.50%	23.00%	32.00%	40.40%
	SIMformer	54.96%	66.33%	75.54%	57.47%	68.73%	78.23%	31.04%	42.08%	51.05%	33.86%	47.53%	60.74%
	Ours $w/\ Emb$	60.26%	72.09%	**81.33%**	64.23%	75.21%	83.44%	36.01%	45.95%	53.35%	44.86%	57.72%	65.80%
	Ours $w/\ Pair$	60.34%	70.74%	78.85%	60.69%	71.06%	79.54%	36.49%	46.45%	53.39%	47.39%	**59.19%**	65.87%
	Ours $w/\ Hybrid$	**61.87%**	**72.87%**	81.13%	**65.74%**	**75.85%**	**83.90%**	**37.29%**	**46.60%**	**53.68%**	**47.59%**	58.96%	**65.97%**
Geolife	NeuTraj	20.39%	30.17%	42.92%	30.64%	35.11%	52.63%	15.29%	29.79%	45.83%	19.76%	31.28%	42.48%
	Traj2SimVec	22.67%	32.60%	45.08%	31.09%	34.60%	50.97%	17.82%	32.17%	44.58%	22.64%	34.71%	44.14%
	T3S	17.85%	29.66%	47.28%	28.16%	41.67%	57.20%	18.34%	33.66%	45.20%	18.01%	28.70%	43.90%
	TMN	26.91%	40.20%	59.10%	32.96%	43.18%	59.19%	20.18%	33.90%	47.10%	19.20%	30.29%	41.18%
	TrajGAT	30.77%	46.08%	60.17%	28.75%	41.18%	55.41%	18.32%	31.56%	41.46%	15.19%	26.19%	36.88%
	TrajCL	28.90%	44.71%	61.16%	30.76%	42.61%	58.27%	22.17%	34.84%	45.27%	19.64%	29.74%	38.82%
	SIMformer	38.11%	53.87%	68.02%	35.54%	51.23%	65.98%	24.30%	37.33%	49.59%	25.13%	36.72%	48.26%
	Ours $w/\ Emb$	45.01%	**62.01%**	**75.54%**	**44.96%**	60.69%	**74.88%**	24.67%	37.51%	**50.31%**	34.74%	48.08%	**58.52%**
	Ours $w/\ Pair$	**45.50%**	61.41%	73.35%	44.14%	59.89%	72.47%	24.77%	37.57%	50.01%	34.93%	**48.43%**	58.28%
	Ours $w/\ Hybrid$	44.47%	60.43%	73.15%	44.87%	**61.04%**	74.01%	**24.94%**	**37.86%**	50.26%	**35.81%**	48.35%	57.61%
T-Driver	NeuTraj	13.06%	25.85%	43.86%	14.32%	42.87%	54.50%	11.09%	23.74%	25.67%	7.92%	16.29%	25.61%
	Traj2SimVec	15.67%	24.40%	44.17%	15.47%	40.81%	52.10%	15.55%	25.15%	36.30%	8.83%	18.67%	27.06%
	T3S	20.27%	35.38%	53.31%	19.75%	47.58%	57.91%	14.29%	22.17%	33.71%	6.18%	15.78%	24.10%
	TMN	22.19%	38.66%	55.47%	22.75%	48.66%	58.72%	18.23%	24.60%	35.84%	6.85%	17.56%	25.89%
	TrajGAT	27.19%	37.99%	56.17%	17.87%	41.86%	55.71%	16.76%	27.62%	36.10%	5.88%	16.72%	23.64%
	TrajCL	25.09%	39.88%	57.62%	23.07%	50.69%	57.89%	15.77%	30.90%	40.31%	9.07%	20.58%	24.73%
	SIMformer	45.40%	60.69%	72.86%	43.34%	58.28%	69.15%	**20.21%**	**32.53%**	**44.27%**	16.98%	27.38%	37.44%
	Ours $w/\ Emb$	42.97%	63.19%	**77.39%**	46.81%	**64.14%**	**75.66%**	18.48%	31.44%	43.89%	22.32%	32.93%	41.36%
	Ours $w/\ Pair$	**48.27%**	**63.81%**	75.32%	43.34%	57.72%	68.46%	16.80%	29.22%	41.48%	14.01%	22.48%	29.12%
	Ours $w/\ Hybrid$	46.64%	63.34%	75.68%	**47.17%**	63.15%	74.28%	17.24%	30.03%	43.17%	**23.34%**	**34.09%**	**42.55%**

Table 2: Performance comparison of the proposed model and different baselines.

Experimental Results

Table 2 summarizes our experimental results. Ours $w/\ Emb$ and Ours $w/\ Pair$ denote variants trained with embedding learning and pairwise learning alone, respectively. Ours $w/\ Hybrid$ uses the hybrid training strategy. The best results are highlighted in **bold**, and the second-best are underlined. From the table, we can draw the following conclusions:

First, across all three datasets, our model demonstrates significant improvements over state-of-the-art methods on four heuristic evaluation metrics. Specifically, it achieves average improvements in HR@1 of 15.11%, 16.49%, 10.41% and 40.15% on the Hausdorff, Discrete Fréchet, SSPD and DTW metrics, respectively. These results clearly demonstrate the effectiveness of our proposed method.

Second, TrajAggperforms exceptionally well on Hausdorff, Discrete Fréchet, and DTW, which emphasize global trajectory structures effectively captured by our model. In contrast, SSPD focuses on fine-grained local distances, which are better preserved by SIMformer and TrajAgg. Moreover, unlike the shorter and more regular trajectories in Porto, the varying sampling frequencies in T-Drive and GeoLife hinder the learning of local features, resulting in relatively lower SSPD performance on these two datasets.

Third, from the results of the Embedding, Pairwise, and Hybrid training modes, we can see that our model supports not only single training strategies but also hybrid ones. It notably outperforms the SOTA methods under hybrid training modes. Overall, hybrid training consistently achieves the best or second-best performance, further demonstrating the strong representational capacity of our proposed aggregation transformer in modeling both fine-grained discriminative features and global trajectory structures.

Fourth, from the perspective of balancing efficiency and expressive power, TrajAggsignificantly outperforms other methods, except for comparable results on T-Drive's SSPD. With minimal additional training cost, it achieves notable improvements, indicating that we attain a better trade-off.

Finally, compared with SIMFormer, our method offers two key advantages: (1) the dual-scale input provides richer spatial information, and (2) our encoder is more effective than the vanilla Transformer, as demonstrated in the ablation study. In addition, compared with methods like TrajCL and T3S that also leverage grid and GPS features, our aggregation transformer adopts a more direct and effective fusion strategy using cross-attention, which outperforms their use of attention masks or simple feature addition.

Ablation Study. To validate key components, we perform ablation by removing specific modules from TrajAggon the Porto dataset, as shown in Figure 3. Moreover, we omit ablation for Hybrid training since its performance is already

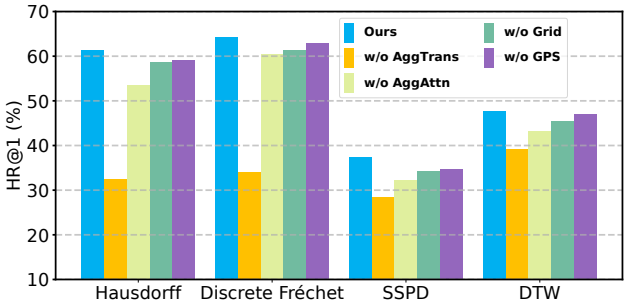


Figure 3: Ablation experiments on the Porto dataset.

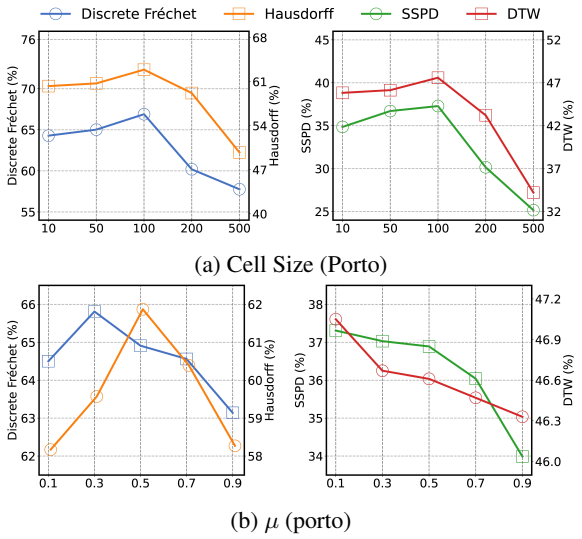
reported in Table 2.

The resulting model variants are defined as follows: ***w/o* AggTrans**: we remove the aggregation transformer and encode trajectories using only the point encoder. ***w/o* AggAttn**: we replace the aggregation attention with a vanilla multi-head self-attention mechanism. ***w/o* Grid**: we use only the grid features as the model input. ***w/o* GPS**: we use only the GPS features as the model input.

As shown in Figure 3, ***w/o* AggTrans** results in a substantial performance drop across all four heuristic metrics, with an average degradation of approximately 31.67%. This clearly demonstrates that the strong encoding capability of our model does not arise solely from the point encoder, but primarily from the Aggregation Transformer itself, which plays a crucial and irreplaceable role in capturing complex spatiotemporal patterns. ***w/o* AggAttn** exhibits notable performance drops on the Hausdorff and Discrete Fréchet metrics, with degradations of 15.85% and 8.82% respectively. In contrast, the declines on SSPD and DTW are less significant, this is because the former two metrics rely more heavily on the information provided by grid-level features, whereas our model is better at integrating multi-scale features. As a result, replacing the attention module with a vanilla Transformer leads to a more significant performance degradation. Finally, both ***w/o* GPS** and ***w/o* Grid** lead to a decline in model performance, indicating that integrating the two types of data enhances the model's effectiveness. Our performance gains stem not only from the combination of the two data sources but also from our effective fusion approach.

Parameter Sensitivity Analysis. We evaluate the sensitivity of TrajAggto the number of weights μ in Eq.(6) and the grid size. As shown in Figure. 4(a), a large grid results in overly coarse trajectory representations, leading to the loss of fine-grained details and reduced similarity accuracy; in contrast, a smaller grid preserves more spatial information but incurs higher computational costs. Furthermore, as shown in Figure. 4(b), better similarity results are generally obtained when μ is small, indicating that GPS trajectories contain rich information, while the grid sequences serve more as a complementary aid.

Efficiency Comparison. We compare the inference speed and per-epoch training time of our method and baselines on the Porto dataset. As shown in Figure 5, SIMformer is



(a) Cell Size (Porto)

(b) μ (porto)

Figure 4: Impact of hyperparameter on model performance.

the fastest at inference because it uses a single-layer Transformer encoder and benefits from PyTorch's optimized parallel implementation. Our model ranks third, slightly slower than Traj2simvec due to the LSTM and dual-attention operations in each layer. TrajCL is slower than our method because it relies on two separate two-layer TransformerEncoders, further reducing inference efficiency.

For training efficiency, our model also ranks third. Traj2simvec trains the fastest thanks to its lightweight sampling strategy, where each anchor is paired with only one positive and one negative sample. In contrast, our method requires more extensive sampling, resulting in longer training time.

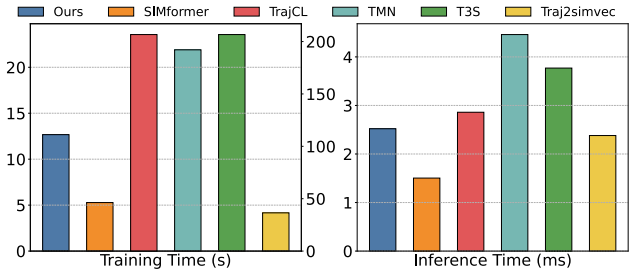


Figure 5: Inference time comparison on the Porto dataset. TrajCL's training time uses the right y-axis; all other methods use the left y-axis.

Conclusion

In this paper, we propose aggregation transformer, an efficient and accurate trajectory encoder that uses dual cross-attention to capture information across two scales. By combining different training paradigms, it learns both local and global patterns. Experiments on three real-world datasets show that our model outperforms state-of-the-art methods.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under grant Nos. 62176243, 62472263, and 62276079, and the Fundamental Research Funds for the Central Universities under grant No. 202442005.

References

Alt, H. 2009. The Computational Geometry of Comparing Shapes. *Efficient Algorithms*, 5760: 235–248.

Alt, H.; and Godau, M. 1995. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02): 75–91.

Chang, Y.; Qi, J.; Liang, Y.; and Tanin, E. 2023. Contrastive trajectory similarity learning with dual-feature attention. In *2023 IEEE 39th International conference on data engineering (ICDE)*, 2933–2945. IEEE.

Chen, J.; Ding, G.; Yang, Y.; Han, W.; Xu, K.; Gao, T.; Zhang, Z.; Ouyang, W.; Cai, H.; and Chen, Z. 2021. Dual-modality vehicle anomaly detection via bilateral trajectory tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4016–4025.

Chen, L.; Gao, Y.; Li, X.; Jensen, C. S.; and Chen, G. 2015. Efficient metric indexing for similarity search. In *2015 IEEE 31st international conference on data engineering*, 591–602. IEEE.

Chen, L.; Özsu, M. T.; and Oria, V. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 491–502.

Chen, Z.; Zhang, D.; Feng, S.; Chen, K.; Chen, L.; Han, P.; and Shang, S. 2024. KGTS: contrastive trajectory similarity learning over prompt knowledge graph embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 8311–8319.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Fang, Z.; Du, Y.; Zhu, X.; Hu, D.; Chen, L.; Gao, Y.; and Jensen, C. S. 2022. Spatio-temporal trajectory similarity learning in road networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 347–356.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jiang, J.; Pan, D.; Ren, H.; Jiang, X.; Li, C.; and Wang, J. 2023. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *2023 IEEE 39th international conference on data engineering (ICDE)*, 843–855. IEEE.

Lee, J.-G.; Han, J.; and Whang, K.-Y. 2007. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, 593–604.

Li, X.; Zhao, K.; Cong, G.; Jensen, C. S.; and Wei, W. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th international conference on data engineering (ICDE)*, 617–628. IEEE.

Ma, Z.; Tu, Z.; Chen, X.; Zhang, Y.; Xia, D.; Zhou, G.; Chen, Y.; Zheng, Y.; and Gong, J. 2024. More than routing: Joint GPS and route modeling for refine trajectory representation learning. In *Proceedings of the ACM Web Conference 2024*, 3064–3075.

O'Connell, M.; moreiraMatias; and Kan, W. 2015. ECML/PKDD 15: Taxi Trajectory Prediction (I). https://kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i. Kaggle.

Si, J.; Yuan, H.; Li, X.; Jiang, N.; Ma, X.; Li, G.; and Wang, S. 2025. Having It Both Ways: Single Trajectory Embedding for Similarity Computation with Pairwise Learning. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 474–486. IEEE Computer Society.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, S.; Bao, Z.; Culpepper, J. S.; Sellis, T.; and Qin, X. 2019. Fast large-scale trajectory clustering. *Proceedings of the VLDB Endowment*, 13(1): 29–42.

Wu, N.; Zhao, X. W.; Wang, J.; and Pan, D. 2020. Learning effective road network representation with hierarchical graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 6–14.

Xia, H.; Zhang, X.; Cao, Y.; Cao, L.; Yu, Y.; and Dong, J. 2025. Self-supervised trajectory representation learning with multi-scale spatio-temporal feature exploration. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 779–792. IEEE.

Yan, Z.; Zhao, X.; Ma, H.; Chen, W.; Qi, J.; Yu, Y.; and Dong, J. 2025. Correlation-attention masked temporal transformer for user identity linkage using heterogeneous mobility data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, 12999–13007.

Yang, C.; Jiang, R.; Xu, X.; Xiao, C.; and Sezaki, K. 2024. SIMformer: Single-Layer Vanilla Transformer Can Learn Free-Space Trajectory Similarity. *arXiv preprint arXiv:2410.14629*.

Yang, P.; Wang, H.; Lian, D.; Zhang, Y.; Qin, L.; and Zhang, W. 2022. TMN: trajectory matching networks for predicting similarity. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 1700–1713. IEEE.

Yang, P.; Wang, H.; Zhang, Y.; Qin, L.; Zhang, W.; and Lin, X. 2021. T3s: Effective representation learning for trajectory similarity computation. In *2021 IEEE 37th international conference on data engineering (ICDE)*, 2183–2188. IEEE.

Yao, D.; Cong, G.; Zhang, C.; and Bi, J. 2019. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *2019 IEEE 35th international conference on data engineering (ICDE)*, 1358–1369. IEEE.

Yao, D.; Hu, H.; Du, L.; Cong, G.; Han, S.; and Bi, J. 2022. Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2275–2285.

Zhang, H.; Chen, W.; Zhao, X.; Qi, J.; Jiang, G.; and Yu, Y. 2025. Scalable trajectory-user linking with dual-stream representation networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, 13224–13232.

Zhang, H.; Zhang, X.; Jiang, Q.; Zheng, B.; Sun, Z.; Sun, W.; and Wang, C. 2020. Trajectory similarity learning with auxiliary supervision and optimal matching.

Zhang, Q.; Wang, Z.; Long, C.; Huang, C.; Yiu, S.-M.; Liu, Y.; Cong, G.; and Shi, J. 2023. Online anomalous subtrajectory detection on road networks with deep reinforcement learning. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 246–258. IEEE.

Zheng, Y. 2011. T-Drive trajectory data sample. T-Drive sample dataset.

Zheng, Y.; Xie, X.; Ma, W.-Y.; et al. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2): 32–39.

Zhou, S.; Li, J.; Wang, H.; Shang, S.; and Han, P. 2023. GRLSTM: trajectory similarity computation with graph-based residual LSTM. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 4972–4980.