

# TrajDiff: Diffusion Bridge Network with Semantic Alignment for Trajectory Similarity Computation

Xiao Zhang<sup>1</sup>, Xingyu Zhao<sup>1</sup>, Hong Xia<sup>1</sup>, Yuan Cao<sup>1</sup>, Guiyuan Jiang<sup>1</sup>, Junyu Dong<sup>1</sup>, Yanwei Yu<sup>1✉</sup>

<sup>1</sup>Faculty of Information Science and Engineering, Ocean University of China, Qingdao, China  
 {zhangxiao4549, zhaoxingyu, xhong}@stu.ouc.edu.cn,  
 {cy8661, jiangguiyuan, dongjunyu, yuyanwei}@ouc.edu.cn

**Abstract**—With the proliferation of location-tracking technologies, massive volumes of trajectory data are continuously being collected. As a fundamental task in trajectory data mining, trajectory similarity computation plays a critical role in a wide range of real-world applications. However, existing learning-based methods face three challenges: First, they ignore the semantic gap between GPS and grid features in trajectories, making it difficult to obtain meaningful trajectory embeddings. Second, the noise inherent in the trajectories, as well as the noise introduced during grid discretization, obscures the true motion patterns of the trajectories. Third, existing methods focus solely on point-wise and pair-wise losses, without utilizing the global ranking information obtained by sorting all trajectories according to their similarity to a given trajectory. To address the aforementioned challenges, we propose a novel trajectory similarity computation framework, named TrajDiff. Specifically, the semantic alignment module relies on cross-attention and an attention score mask mechanism with adaptive fusion, effectively eliminating semantic discrepancies between data at two scales and generating a unified representation. Additionally, the DDBM-based Noise-robust Pre-Training introduces the transfer patterns between any two trajectories into the model training process, enhancing the model’s noise robustness. Finally, the overall ranking-aware regularization shifts the model’s focus from a local to a global perspective, enabling it to capture the holistic ordering information among trajectories. Extensive experiments on three publicly available datasets show that TrajDiff consistently outperforms state-of-the-art baselines. In particular, it achieves an average HR@1 gain of 33.38% across all three evaluation metrics and datasets. The code for our model is available at <https://github.com/zhx66741/TrajDiff>.

**Index Terms**—Trajectory similarity computation, denoising diffusion bridge, cross attention, attention score mask

## I. INTRODUCTION

With the rapid development of positioning and sensing technologies, vast amounts of trajectory data are continuously collected from diverse sources [1]–[4]. These spatio-temporal trajectories are essential in various applications, including intelligent transportation systems [5], [6], urban planning [7], and influenza monitoring [8]. Among the key tasks in trajectory data mining, trajectory similarity computation is a fundamental operation [9]–[11], forming the basis for a wide range of downstream tasks such as trajectory clustering, retrieval, and anomaly detection. Therefore, accurate and efficient similarity computation is crucial for informed decision-making and driving advancements in these fields [12]–[14].

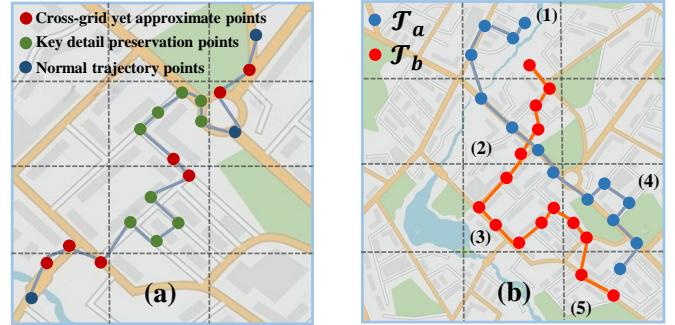


Fig. 1: Many-to-one mapping issues caused by feature misalignment and noise interference. (a) All green points are projected onto the same grid, ignoring fine-grained information; the red points are geographically adjacent, but after discretization, they fall into different grids. (b) Two trajectories are projected onto the same grid sequence, yet exhibit differences in their activity regions and movement patterns.

Generally, trajectory similarity computation methods can be categorized into two classes: **heuristic measures** and **learnable measures** [15]. Heuristic measures are typically based on rules defined manually to identify the optimal point match between two trajectories [16]. For instance, the Symmetric Segment Path Distance (SSPD) is computed as the symmetric average of the minimum distances between corresponding segments of two trajectories. However, heuristic measures typically rely on dynamic programming or geometric computations, resulting in a time complexity that often reaches  $\mathcal{O}(n^2)$  or even higher [17]–[19], where  $n$  represents the number of points in the trajectory. When processing large-scale trajectory datasets, this computational complexity creates a significant performance bottleneck. Furthermore, for a dataset containing  $N$  trajectories, the number of required pairwise computations increases quadratically, scaling as  $\mathcal{O}(N^2)$ . For example, computing pairwise Fréchet distances for 1,000, 2,000, and 7,000 trajectories takes approximately 1.4 hours, 3.2 hours, and more than 13 hours, respectively.

In contrast, learnable measures employ neural networks to map trajectories into  $d$ -dimensional space, transforming pairwise similarity computations into highly efficient vec-

tor operations, typically with  $\mathcal{O}(1)$  time complexity. As an illustration, TrajCL [15] adopts an attention-based encoder to model trajectories, achieving speed improvements of 21 $\times$  and 25 $\times$  compared to Hausdorff and Fréchet distance computations, respectively. Motivated by this, we also propose a learning-based approach to enable fast and effective trajectory similarity computation.

Recent studies like KGTS [20], TrajCL [15], and TrajGAT [21] have advanced trajectory similarity computation using novel approaches. TrajGAT introduces a graph attention network framework that captures the structural dependencies among trajectory points by integrating both geographical semantics and topological connections, enabling more accurate trajectory similarity modeling. Building upon representation learning, TrajCL employs an unsupervised contrastive learning paradigm that generates augmented views of raw trajectories and optimizes a similarity-preserving objective to learn robust trajectory embeddings. Extending the modeling capacity with external knowledge, KGTS leverages knowledge graph embeddings and prompt-based learning, and jointly encodes trajectories and spatial grids through contrastive objectives to enhance trajectory similarity computation.

However, there are still three challenges in trajectory similarity computation. ***First, the primary challenge arises from the misalignment between the fine-grained GPS features and coarse-grained grid features of trajectories.*** These alignment errors result in many-to-one mapping issues. On one hand, multiple trajectory points may be projected onto the same grid and assigned identical values, thereby overlooking fine-grained movement features such as vehicle acceleration and intersection turns. On the other hand, different trajectories might be mapped to the same grid sequence, appearing similar at the grid level but exhibiting significant differences in their finer details. As illustrated in Figure 1(a), the green point sequences capture fine-grained movement patterns. However, at the grid level, they are confined to the same grid cell, causing these detailed behaviors to be lost. Meanwhile, Fig. 1(b) illustrates that trajectory  $\mathcal{T}_a$  is primarily active in regions (1) and (4), whereas trajectory  $\mathcal{T}_b$  predominantly operates in regions (3) and (5). Despite both trajectories being mapped to the same grid sequence, they exhibit notable differences in their active regions and movement patterns within those regions.

***Second, the presence of noise in the data masks the intrinsic trajectory patterns, hindering the model's ability to learn precise and reliable representations.*** Trajectory data is inherently noisy, arising from factors such as the limitations of data collection devices, environmental interference, and variations in user behavior. Moreover, additional noise emerges during the transformation of continuous GPS features into discrete grid features. In such cases, when a trajectory point lies near the boundary of a grid cell, even a minor physical displacement or device error can be significantly amplified by the grid quantization mechanism. As depicted in Figure 1(a), the red point sequences exhibit a high degree of clustering within the fine-grained GPS coordinate space. However, following grid discretization, the spatial distance between

their corresponding grid-based representations is substantially magnified.

***Third, existing methods often neglect the global ranking structure of trajectories, which is essential for capturing holistic similarities.*** Trajectory similarity computation and retrieval tasks fundamentally share the same core principles. The retrieval task involves identifying the most similar or top- $k$  samples from a dataset based on a given query item, which is conceptually aligned with a similar trajectory query, as both aim to achieve the same objective. However, mainstream approaches that use point-wise losses (e.g., MSE) or pairwise losses (e.g., triplet loss) have significant limitations: the former treats individual samples in isolation, while the latter focuses solely on pairwise relationships, both of which fail to capture the overall ranking information across the dataset. This limitation is especially critical in practical applications. For instance, in the trajectory similarity query task, after applying the proposed overall ranking-aware regularization, we achieved average improvements of 15.47%, 23.89%, and 52.85% in SSPD distance under HR@1 metric across three datasets, compared to using MSE loss alone.

To address the aforementioned challenges, we propose a novel trajectory similarity computation framework, named **TrajDiff**. Specifically, we propose the semantic alignment module to effectively bridge the semantic gap between features at two distinct scales, integrating them into a unified and expressive embedding. To further enhance its ability to model latent semantic transitions, we pre-train this module using Denoising Diffusion Bridge Models (DDBM), by learning to progressively denoise intermediate states conditioned on both endpoints, the module captures the underlying dynamics while preserving semantic consistency and improving robustness to input noise. Finally, we design the overall ranking-aware regularization to incorporate global ranking information into the model training, enabling it to capture the overall ranking patterns effectively. Experiments on three public datasets show the superiority of our model. We achieve the maximum performance gain of 92.99% on the T-Drive dataset under the SSPD evaluation metric.

We summarize the key contributions of this work as follows:

- We propose a novel semantic alignment module that explicitly models and integrates information from two different scales, effectively eliminating the semantic gap.
- We propose a novel pretraining method based on DDBM, which leverages both the forward and backward diffusion processes to learn robust and semantically consistent representations.
- We propose a novel overall ranking-aware regularization to optimize the embedding space from a global ranking perspective, improving query performance.
- We conduct extensive experiments on three real-world datasets to demonstrate the superiority of our TrajDiff over state-of-the-art baselines, achieving an average 53.58% improvement under SSPD metrics across three datasets on HR@1.

## II. RELATED WORK

In this section, we review important related work on three critical aspects that significantly influence the problem addressed in this paper: grid embedding pretraining, trajectory feature aggregation, and self-supervised trajectory similarity computation.

### A. Grid Embedding Pretraining

In trajectory similarity computation, the geographical area is typically partitioned into non-overlapping grids of fixed size to obtain a coarse-grained representation of the overall trajectory. However, using only grid IDs as features is insufficient, prompting the adoption of various pretraining methods to learn high-dimensional grid representations. TrajCL [15] constructs the grids as a graph structure and pre-trains the grid embedding using the Node2Vec [22]. TrajGAT also employs Node2Vec to obtain grid embeddings. Unlike TrajCL, it partitions the entire geographical area into a PR quadtree [23] based on the spatial distribution of trajectory points. KGTS [20] pre-trains spatially-aware grid embeddings using a modified RotatE [24] based on complex-space rotation mechanisms. GRLSTM [25] pre-trains grid embedding using the TransH [26] model on a knowledge graph that integrates trajectory and road network information.

However, existing pretraining approaches face two major limitations: first, the pretrained grid embeddings remain fixed and are not updated during model training; second, these methods are generally graph-based, where the grid embeddings are updated by considering the connectivity of neighboring nodes, but they fail to account for skipping connections in real trajectories. To overcome these limitations, we abandon the pretraining paradigm and instead employ a projection layer that maps grid coordinates directly into a high-dimensional embedding space.

### B. GPS and Grid Feature Aggregation

Effective fusion of GPS features and grid features is crucial for trajectory similarity computation. T3S [27] processes the GPS features and the grid features using LSTM [28], [29] and a self-attention-based network. The final trajectory embedding is obtained by summing these two components. TrajCL leverages a dual-feature self-attention mechanism to fuse structural and spatial features of trajectories by separately computing and adaptively integrating their attention weights, enabling the learning of more comprehensive trajectory representations. In addition, some methods utilize only one type of feature. Traj2SimVec [30] introduces an efficient and robust model through innovative designs, including trajectory simplification, efficient indexing, sub-trajectory distance loss, and point matching loss. TMN [12] solely relies on the GPS features, and it proposes an attention-based matching mechanism for point-wise matching between different trajectories. The matching mechanism not only takes into account the sequential information of individual trajectories but also incorporates the interaction information between different trajectories for similarity, thereby improving the accuracy of similarity calculation. To

learn more realistic grid embeddings, HHL-Traj [31] employs a hypergraph [32] to model the grid space and a GRU with residual connections to obtain more discriminative trajectory embeddings. Additionally, a deep hash network is used to transform the trajectory into binary hash codes, enabling fast similarity computation.

### C. Self-supervised Trajectory Similarity Computation

In recent years, trajectory similarity computation has increasingly embraced the self-supervised learning paradigm [33], [34], which enables the acquisition of more discriminative trajectory representations without relying on manual annotations. Existing self-supervised methods generally follow two main approaches: reconstruction-based and contrastive-based learning. t2vec [35] employs a seq2seq [36], [37] architecture to encode low sampling rate trajectories into continuous vectors in an unsupervised setting, and achieves similarity modeling by reconstructing high sampling rate trajectories. In contrast, contrastive learning methods directly optimize representation similarity. CL-TSim [38] adopts the SimCLR framework [39] for pre-training via contrastive learning, where positive and negative sample pairs are generated through trajectory downsampling and perturbation. Similarly, TrajCL is built upon the MoCo framework [40], enhancing the stability of contrastive learning through a momentum encoder and a negative sample queue, and then fine-tunes the model to obtain robust and discriminative trajectory representations. Furthermore, KGST [20] proposes grid-aware data augmentation strategies and introduces Prompt Trajectory Embedding, which guides the encoder to capture structured semantic patterns in grid-based trajectories better.

## III. PRELIMINARIES

In this section, we introduce the foundational concepts and formally define the problem of trajectory similarity computation.

**Definition 1** (Trajectory). A trajectory is defined as an ordered sequence of spatio-temporal points:  $\mathcal{T} = [p_1, p_2, \dots, p_n]$ , where  $p_i$  denotes the  $i$ -th location point, typically represented as  $p_i = (\text{lon}_i, \text{lat}_i)$ , and  $n$  represents the length of the trajectory.

**Definition 2** (Grid Partitioning). Given a geographical region bounded by latitude  $[\text{lat}_{\min}, \text{lat}_{\max}]$  and longitude  $[\text{lon}_{\min}, \text{lon}_{\max}]$ , and the predefined cell size, the region is divided into a uniform grid of size  $M \times U$ , where  $M$  and  $U$  represent the number of divisions along the latitude and longitude axes, respectively. Each grid cell  $g_{i,j}$  corresponds to a fixed rectangular subregion within the geographical area. Formally, the set of all grid cells in this spatial partition is defined as:

$$\mathcal{G} = \{g_{i,j} \mid 1 \leq i \leq M, 1 \leq j \leq U\},$$

where  $\mathcal{G}$  denotes the complete set of grid cells in the  $M \times U$  partition.

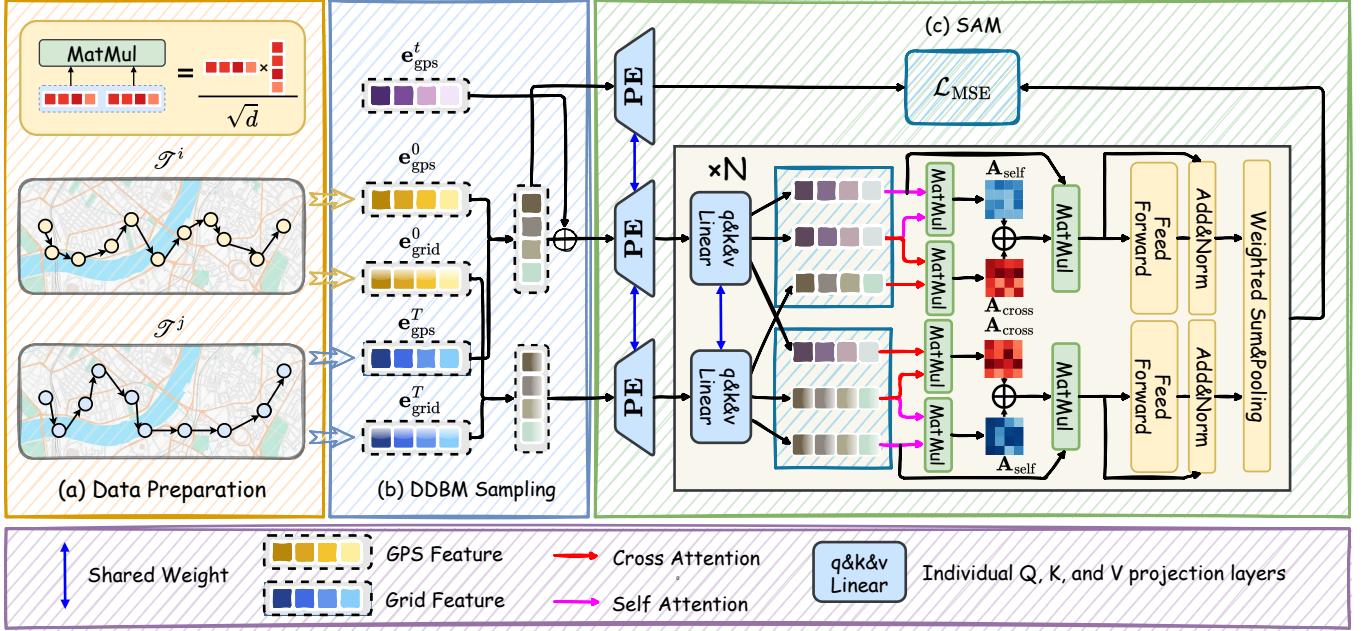


Fig. 2: DDBM-based Noise-robust Pre-Training.

**Definition 3** (Trajectory Grid Sequence). A trajectory grid sequence, denoted as  $\mathcal{T}_g$ , is the sequence of grid cells obtained by mapping each point in a trajectory  $\mathcal{T} = [p_1, p_2, \dots, p_n]$  to its corresponding grid cell in the grid set  $\mathcal{G}$ :

$$\mathcal{T}_g = [g_{i_1, j_1}, g_{i_2, j_2}, \dots, g_{i_n, j_n}] \in \mathbb{R}^{n \times 2},$$

where  $g_{i_n, j_n}$  is the grid cell containing  $p_n$ .

**Definition 4** (Heuristic Similarity). Heuristic measures quantify the similarity between two trajectories using predefined distance metrics, such as Hausdorff or Discrete Fréchet distance. The computed similarity values are stored in a square matrix  $\mathbf{H} \in \mathbb{R}^{N \times N}$ , where  $N$  represents the number of trajectories and each entry  $\mathbf{H}_{ij}$  represents the distance between trajectories  $\mathcal{T}_i$  and  $\mathcal{T}_j$  under the chosen metric.

**Problem** (Trajectory Similarity Computation). Given a heuristic metric  $d(\cdot, \cdot)$  and two arbitrary trajectories  $\mathcal{T}_i$  and  $\mathcal{T}_j$ , the objective is to learn a parameterized trajectory similarity function  $f_\theta(\cdot, \cdot)$  that minimizes the discrepancy between  $f_\theta(\mathcal{T}_i, \mathcal{T}_j)$  and  $d(\mathcal{T}_i, \mathcal{T}_j)$  as defined by the following equation:

$$\operatorname{argmin}_\theta |f_\theta(\mathcal{T}_i, \mathcal{T}_j) - d(\mathcal{T}_i, \mathcal{T}_j)|. \quad (1)$$

In this work, we use DDBM to learn the reasonable evolution between any two trajectories, enhancing the model's robustness to noise. Next, we briefly review some basic concepts needed to understand diffusion models [41].

**Diffusion process.** To model the data distribution  $q_{\text{data}}(\mathbf{x})$  for  $\mathbf{x} \in \mathbb{R}^d$ , a stochastic diffusion process is introduced, defined by a sequence of time-indexed variables  $\{\mathbf{x}_t\}_{t=0}^T$ . This process begins with  $\mathbf{x}_0 \sim p_0(\mathbf{x}) := q_{\text{data}}(\mathbf{x})$  and evolves such that the final state satisfies  $\mathbf{x}_T \sim p_T(\mathbf{x}) := p_{\text{prior}}(\mathbf{x})$ , where  $p_{\text{prior}}(\mathbf{x})$  denotes a fixed prior distribution, such as the Gaussian

distribution. The forward dynamics of the system are governed by the following stochastic differential equation (SDE):

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t) dt + g(t) d\mathbf{w}_t, \quad (2)$$

where  $\mathbf{f} : \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^d$  represents the drift function,  $g(t)$  is a scalar diffusion coefficient, and  $\mathbf{w}_t$  denotes a standard Wiener process. Under this formulation, the variable  $\mathbf{x}_T$  follows the prior distribution. The time-reversed process that recovers  $\mathbf{x}_0$  from  $\mathbf{x}_T$  is characterized by:

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) dt + g(t) d\mathbf{w}_t, \quad (3)$$

where  $p(\mathbf{x}_t)$  is the marginal density of  $\mathbf{x}_t$  at time  $t$ . Additionally, this stochastic process admits an equivalent deterministic formulation known as the probability flow ODE [42], which shares the same marginal distributions:

$$d\mathbf{x}_t = \left[ f(\mathbf{x}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \right] dt. \quad (4)$$

This formulation allows sampling from  $q_{\text{data}}$  by first drawing  $\mathbf{x}_T \sim q_{\text{data}}(y)$  and then solving the reverse-time SDE or ODE to obtain the corresponding  $\mathbf{x}_0$ .

**Denoising score matching.** The score function  $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$  can be estimated through a denoising score-matching objective, defined as:

$$\mathcal{L}(\theta) = \mathbb{E} \left[ \|s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t | \mathbf{x}_0)\|^2 \right], \quad (5)$$

by minimizing this objective yields a score estimator  $s_\theta^*(\mathbf{x}_t, t)$  that approximates the true conditional score. This formulation remains tractable due to the fact that the transition distribution  $p(\mathbf{x}_t | \mathbf{x}_0)$  is explicitly defined as the Gaussian:  $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \epsilon$ ,  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ , where  $\alpha_t$  and  $\sigma_t$  are time-dependent functions that control the scaling of the signal and

the noise, respectively. In practice, it is common to express the diffusion process in terms of the signal-to-noise ratio (SNR), defined as  $\alpha_t^2/\sigma_t^2$ .

TABLE I: Key notations and descriptions.

Notations	Descriptions
$\mathcal{T}$	Trajectory data
$\mathcal{T}^0, \mathcal{T}^T$	Initial and target states of the DDBM during pre-training
$e_{\text{gps}}, e_{\text{grid}}$	Trajectory GPS feature and grid features
$\mathcal{T}_g$	Trajectory grid sequence
$n$	The number of points in a trajectory
$N$	Number of trajectories in dataset
$d$	Model embedding dimension
$d_h$	Hidden layer dimension
$L$	Total layers of Semantic Alignment Module
$\mathcal{G}$	The set of all spatial grid cells
$\mathbf{H}$	Heuristic similarity matrix
$\mathbf{H}_p$	Model predicted similarity
$h$	Trajectory embedding
$d(\cdot, \cdot)$	Heuristic metric
$\mathbf{PE}$	Pre-encoding module
$\lambda_q, \lambda_k, \lambda_v$	Adaptive scaling factors
$\gamma_1, \gamma_2$	List-wise loss weighting coefficients
$\epsilon$	Feature fusion coefficient
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Attention matrix

#### IV. METHOD

Our proposed framework consists of three main components: (1) *Semantic Alignment Module (SAM)*, (2) *DDBM-based Noise-robust Pre-Training*, and (3) *Overall Ranking-aware Regularization*. First, SAM bridges coarse and fine grained inputs by aligning their semantic meanings and integrating them into a unified representation. Second, DDBM-based noise-robust pre-training enables the model to learn a stochastic transformation between paired trajectories and improved its robustness to noise. Third, overall ranking-aware regularization incorporates global trajectory ranking information into the model’s optimization process, enabling it to better capture the overall ranking structure and improve performance on ranking-sensitive metrics.

##### A. Semantic Alignment Module

We propose SAM as a foundational architecture for trajectory similarity computation. It is specifically designed to explicitly model and align representations across sequences with heterogeneous granularities. Specifically, SAM comprises  $L$  stacked Semantic Alignment Layer (SALayer), each layer consisting of two components: a dual semantic alignment attention module followed by a residual feedforward network. Specifically, given a trajectory  $\mathcal{T}$ , we first extract its fine-grained GPS feature  $e_{\text{gps}} \in \mathbb{R}^{n \times 2}$  and then project the points onto the corresponding grids to obtain the coarse-grained grid features  $e_{\text{grid}} \in \mathbb{R}^{n \times 2}$ . Subsequently, we encode the inputs using the pre-encoding module  $\mathbf{PE}$  to map them into a unified  $d$ -dimensional space. For SSPD and Hausdorff distance,  $\mathbf{PE}$  is implemented as a linear layer, while for discrete Fréchet distance, it consists of a one-layer LSTM. The computation process is as follows:

$$\mathbf{Z}_{\text{gps}}^{(0)} = \mathbf{PE}(e_{\text{gps}}), \quad \mathbf{Z}_{\text{grid}}^{(0)} = \mathbf{PE}(e_{\text{grid}}) \quad (6)$$

where  $\mathbf{Z}_{\text{gps}}^{(0)}, \mathbf{Z}_{\text{grid}}^{(0)} \in \mathbb{R}^{n \times d}$  and  $d$  denotes the embedding dimension of the model. The pre-encoded representations  $\mathbf{Z}_{\text{gps}}^{(0)}, \mathbf{Z}_{\text{grid}}^{(0)}$  are then progressively refined through the stacked SALayer. At each layer  $l \in [1, L]$ , the representations are updated as follows:

$$\mathbf{Z}_{\text{gps}}^{(l)}, \mathbf{Z}_{\text{grid}}^{(l)} = \text{SALayer}^l(\mathbf{Z}_{\text{gps}}^{(l-1)}, \mathbf{Z}_{\text{grid}}^{(l-1)}). \quad (7)$$

In the following sections, we first introduce the Semantic Alignment Attention (SAA), and then describe the Dual SAA.

1) *Semantic Alignment Attention*: The core design of SAA lies in combines cross-attention mechanism with an adaptively scaled attention score mask strategy to bridge the semantic gap between inputs of two different scales. Given two input representations  $\mathbf{Z}_{\text{gps}}^{(0)}, \mathbf{Z}_{\text{grid}}^{(0)} \in \mathbb{R}^{n \times d}$ , we treat  $\mathbf{Z}_{\text{gps}}^{(0)}$  as the query and  $\mathbf{Z}_{\text{grid}}^{(0)}$  as the source for both key and value. Linear projections are then applied to obtain the query, key, and value matrices:  $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$ . The cross-attention score  $\mathbf{A}_{\text{cross}}$  is computed via the scaled dot-product between  $\mathbf{Q}$  and  $\mathbf{K}$ , while the self-attention score  $\mathbf{A}_{\text{self}}$  is derived from the scaled dot-product between  $\mathbf{K}$  and  $\mathbf{V}$ . These are formally defined as:

$$\mathbf{Q} = \mathbf{Z}_{\text{gps}}^{(0)} \mathbf{W}^q, \mathbf{K} = \mathbf{Z}_{\text{grid}}^{(0)} \mathbf{W}^k, \mathbf{V} = \mathbf{Z}_{\text{grid}}^{(0)} \mathbf{W}^v, \quad (8)$$

$$\mathbf{A}_{\text{cross}} = \text{softmax} \left( \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} \right), \quad (9)$$

$$\mathbf{A}_{\text{self}} = \text{softmax} \left( \frac{\mathbf{K}\mathbf{V}^T}{\sqrt{d}} \right), \quad (10)$$

where  $\mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v \in \mathbb{R}^{d \times d}$  are learnable weight matrices.  $\mathbf{A}_{\text{self}}$  captures intra-scale dependencies, and  $\mathbf{A}_{\text{cross}}$  further facilitates cross-scale information exchange, progressively narrowing the semantic gap. To achieve more comprehensive alignment, additional mechanisms are required to fully bridge the representations across different granularities. To achieve this goal, we introduce an adaptive fusion mechanism that dynamically computes weighting coefficients for  $\mathbf{A}_{\text{self}}$  and  $\mathbf{A}_{\text{cross}}$  as follows:

$$\begin{aligned} \lambda_{\text{cross}} &= \exp \left( \sum_{i=1}^d \lambda_q^{(i)} \cdot \lambda_k^{(i)} \right), \\ \lambda_{\text{self}} &= \exp \left( \sum_{i=1}^d \lambda_k^{(i)} \cdot \lambda_v^{(i)} \right), \\ \mathbf{Z}_{\text{gps}}^{(l)} &= \text{SAA}(\mathbf{Z}_{\text{gps}}^{(l-1)}, \mathbf{Z}_{\text{grid}}^{(l-1)}) \\ &= (\lambda_{\text{self}} \cdot \mathbf{A}_{\text{self}} + \lambda_{\text{cross}} \cdot \mathbf{A}_{\text{cross}}) \mathbf{V}, \end{aligned} \quad (11)$$

where  $\lambda_q, \lambda_k$ , and  $\lambda_v \in \mathbb{R}^d$  are learnable parameters that are used to derive the scaling factors  $\lambda_{\text{self}}$  and  $\lambda_{\text{cross}}$  for adaptive attention fusion. The output  $\mathbf{Z}_{\text{gps}}^{(l)}$  is further updated through a feedforward network with residual connections, formulated as:

$$\mathbf{Z}_{\text{gps}}^{(l)} = \text{Norm} \left( \mathbf{Z}_{\text{gps}}^{(l)} + \text{FFN}(\mathbf{Z}_{\text{gps}}^{(l)}) \right), \quad (12)$$

where  $\text{Norm}(\cdot)$  denotes layer normalization.

2) *Dual Semantic Alignment Attention*: To further bridge the semantic gap between inputs of different granularities, we extend the SAA mechanism to a dual formulation. Specifically, after computing  $\mathbf{Z}_{\text{gps}}^{(l)} = \text{SAA}(\mathbf{Z}_{\text{gps}}^{(l-1)}, \mathbf{Z}_{\text{grid}}^{(l-1)})$ , we symmetrically reverse the roles of the two inputs by treating  $\mathbf{Z}_{\text{grid}}^{(l-1)}$  as the query and  $\mathbf{Z}_{\text{gps}}^{(l-1)}$  as the key-value source. Under this reversed configuration, we compute  $\mathbf{Z}_{\text{grid}}^{(l)}$  via a symmetric attention operation similar to Eqs. (10), (11) and (12).

After updating through  $L$  stacked SALayers, we obtain the final representations  $\mathbf{Z}_{\text{gps}}^{(L)}$  and  $\mathbf{Z}_{\text{grid}}^{(L)}$ . These two representations are then fused into a unified embedding via a weighted combination controlled by a hyperparameter  $\epsilon$ :

$$\mathbf{h} = \epsilon \cdot \mathbf{Z}_{\text{gps}}^{(L)} + (1 - \epsilon) \cdot \mathbf{Z}_{\text{grid}}^{(L)}, \quad (13)$$

where  $\mu \in (0, 1)$  serves as a balancing hyperparameter that controls the relative contribution of  $\mathbf{Z}_{\text{gps}}^{(L)}$  and  $\mathbf{Z}_{\text{grid}}^{(L)}$  to the final fused representation  $\mathbf{h}$ .

In summary, the proposed Dual SAA significantly enhances the model's ability to capture nuanced cross-granular interactions, leading to more comprehensive and expressive representations in practice.

### B. DDBM-based Noise-robust Pre-Training

As discussed above, trajectory data is often affected by noise. To address this issue, we adopt DDBM as the pretraining strategy, which enables the model to learn interactions and transfer patterns between arbitrary pairs of trajectories, enhancing robustness and generalization. We first introduce DDBM and then explain how it is adapted for our task. Figure 2 presents an overview of the pretraining process.

1) *Denoising Diffusion Bridge Models*: The objective of DDBM is to learn bidirectional conditional generation directly from the initial state  $\mathbf{x}_0$  to the terminal state  $\mathbf{x}_T$  through the diffusion bridge. In the forward process, DDBM constructs a bridge diffusion process with fixed endpoints, transporting the initial state  $\mathbf{x}_0$  to the target state  $\mathbf{x}_T = y$ . This process is formulated by modifying the drift term using Doob's  $h$ -transform [43], [44]. Accordingly, Eq.(2) is rewritten as follows:

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)^2(\mathbf{h}_t)dt + g(t)d\mathbf{w}_t, \quad (14)$$

where  $\mathbf{h}_t = \mathbf{h}(x, t, y, T) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_T = y | \mathbf{x}_t = x)$  denotes the gradient of the log transition kernel, guaranteeing almost sure convergence to  $y$  at terminal time  $T$ . Using this equation, we can move from  $\mathbf{x}_0$  to  $\mathbf{x}_T$ .

In the reverse process, to sample from the terminal state  $\mathbf{x}_T$  back to  $\mathbf{x}_0$ , DDBM formulates the time-reversed dynamics of the conditional distribution  $q(\mathbf{x}_t | \mathbf{x}_T)$ :

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2(\mathbf{s}_t - \mathbf{h}_t)] dt + g(t) d\hat{\mathbf{w}}_t, \quad (15)$$

where  $\mathbf{s}_t = \mathbf{s}(x, t, y, T) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t = x | \mathbf{x}_T = y)$  is the gradient of the conditional probability distribution of the state  $x_t$  given the terminal state  $x_T$  which is learned by neural network, and  $\hat{\mathbf{w}}_t$  denotes the reverse-time Wiener process.

For loss computation, since the true bridge distribution score  $s(x, t, y, T)$  is not available in closed form, DDBM leverages the tractable Gaussian bridge marginal  $q(\mathbf{x}_t | \mathbf{x}_0, \mathbf{x}_T)$  to define the training objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_T, t, \mathbf{x}_t} \| s_\theta(\mathbf{x}_t, \mathbf{x}_T, t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0, \mathbf{x}_T) \|^2, \quad (16)$$

minimizing this loss drives the network output  $s_\theta$  to match the true conditional score  $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_T)$ . In practice, DDBM adopts the pred-x parameterization from EDM [45]. We also apply this technique, and redefine the loss calculation as the MSE loss between the true value  $\mathbf{x}$  and the model output. For more details, please refer to DDBM [46].

2) *DDBM Pre-training*: Given an arbitrary pair of distinct trajectories, one is designated as the initial state  $\mathcal{T}^0$ , while the other serves as the terminal state  $\mathcal{T}^T$  at the final time step. We extract their GPS and grid features at both the initial and final states,  $\mathbf{e}_{\text{gps}}^0, \mathbf{e}_{\text{grid}}^0$  and  $\mathbf{e}_{\text{gps}}^T, \mathbf{e}_{\text{grid}}^T$ , respectively. Subsequently, following the DDBM sampling process, we obtain the intermediate time data  $\mathbf{e}_{\text{gps}}^t$  at time  $t$  by sampling from  $\mathbf{e}_{\text{gps}}^0$  and  $\mathbf{e}_{\text{gps}}^T$ , the formula is:

$$q(\mathbf{x}_t | \mathbf{x}_0, \mathbf{x}_T) = \mathcal{N}(\hat{\mu}_t, \hat{\sigma}_t^2 I),$$

$$\begin{aligned} \hat{\mu}_t &= \frac{\text{SNR}_T}{\text{SNR}_t} \frac{\alpha_t}{\alpha_T} \mathbf{e}_{\text{gps}}^T + \alpha_t \mathbf{e}_{\text{gps}}^0 \left(1 - \frac{\text{SNR}_T}{\text{SNR}_t}\right), \\ \hat{\sigma}_t^2 &= \sigma_t^2 \left(1 - \frac{\text{SNR}_T}{\text{SNR}_t}\right), \\ \mathbf{e}_{\text{gps}}^t &= \hat{\mu}_t + \hat{\sigma}_t \times \mathbf{e}_{\text{Noise}}^t, \end{aligned} \quad (17)$$

where  $\hat{\mu}_t$  denotes the ground truth of the sample  $\mathbf{e}_{\text{gps}}^t$ ,  $\hat{\sigma}_t^2$  represents the corresponding noise variance and  $\mathbf{e}_{\text{Noise}}^t$  denotes randomly Gaussian noise. In addition,  $\alpha_t$  and  $\sigma_t$  are the pre-defined signal and noise schedules, respectively, and the signal-to-noise ratio (SNR) is computed as  $\text{SNR}_t = \alpha_t^2 / \sigma_t^2$ . In practice, we adopt a linear noise schedule defined as  $\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min})$ , which yields the signal coefficient:

$$\alpha_t = \exp \left( -\frac{1}{2} \int_0^t \beta(s) ds \right). \quad (18)$$

Using Eq. (17), we obtain the intermediate state  $\mathbf{e}_{\text{gps}}^t$ , which serves as one of the dual-scale inputs to the model. Since  $\mathbf{e}_{\text{gps}}^t$  contains only fine-grained and potentially noisy information, we further compute a noise-free, coarse-grained representation  $\mathbf{e}_{\text{grid}}^t$  from the grid view to complement the input. The formula is given as follows:

$$\mathbf{e}_{\text{grid}}^t = \frac{\text{SNR}_T}{\text{SNR}_t} \frac{\alpha_t}{\alpha_T} \mathbf{e}_{\text{grid}}^T + \alpha_t \mathbf{e}_{\text{grid}}^0 \left(1 - \frac{\text{SNR}_T}{\text{SNR}_t}\right), \quad (19)$$

this auxiliary feature  $\mathbf{e}_{\text{grid}}^t$  is incorporated into the SAM with  $\mathbf{e}_{\text{gps}}^t$  to provide stable structural guidance and help the model better capture the high-level semantic patterns of the trajectory.

Then, we can obtain the trajectory embedding using the following formula:

$$\begin{aligned} \mathbf{Z}_{\text{gps}}^t &= \mathbf{PE}(\mathbf{e}_{\text{gps}}^t), \quad \mathbf{Z}_{\text{grid}}^t = \mathbf{PE}(\mathbf{e}_{\text{grid}}^t), \\ \mathbf{h} &= \text{SAM}(\mathbf{Z}_{\text{gps}}^t, \mathbf{Z}_{\text{grid}}^t), \end{aligned} \quad (20)$$

where  $\mathbf{h} \in \mathbb{R}^{n \times d}$  denotes the reconstruction of  $\hat{\mu}$  in the embedding space. We follow the pred-x parameterization for loss computation, while computing the loss in the embedding space rather than the original data space. The objective is formulated as:

$$\begin{aligned}\mathbf{h}_{\hat{\mu}} &= \mathbf{PE}(\hat{\mu}), \\ \mathcal{L}_{\text{MSE}} &= \|\mathbf{h} - \mathbf{h}_{\hat{\mu}}\|^2.\end{aligned}\quad (21)$$

By computing the loss in the high-dimensional space, we can better optimize the model's performance in the feature space, improving the accuracy and stability of trajectory similarity computation, while also mitigating the influence of local noise in the original data space to enable the model to capture the global similarity between trajectories more accurately.

### C. Overall Ranking-aware Regularization

To explore the global trajectory-level ranking information that has been overlooked in previous studies, we introduce the overall ranking-aware regularization, which combines two list-wise ranking objectives: **ListNet** [47] and **Rank-Decay ListNet**. We first provide a detailed explanation of how ListNet and Rank-Decay ListNet are utilized, along with the overall loss function employed during the fine-tuning stage.

1) *ListNet*: Let  $s = [s_1, s_2, \dots, s_k]$  denote the predicted similarity scores for  $k$  candidate trajectories with respect to a query trajectory, and let  $r = [r_1, r_2, \dots, r_k]$  denote the corresponding heuristic similarity labels. We adopt the list-wise learning-to-rank paradigm, which treats the entire set of candidates as a single learning instance, rather than focusing on individual items or item pairs.

Specifically, we adopt the top-one probability formulation proposed in ListNet, which models the probability of each candidate being ranked at the top based on its similarity score. Given a list of predicted similarity scores  $s$  and the corresponding ground-truth scores  $r$ , the top-one probability distributions are computed using the softmax function:

$$P(i | s) = \frac{\exp(s_i)}{\sum_{j=1}^k \exp(s_j)}, \quad (22)$$

$$P(i | r) = \frac{\exp(r_i)}{\sum_{j=1}^k \exp(r_j)}, \quad (23)$$

where  $P(i | s)$  denotes the model-predicted probability of the  $i$ -th candidate being ranked at the top, and  $P(i | r)$  denotes the target top-one probability for the  $i$ -th candidate, computed from the heuristic similarity labels  $r$ . The ListNet loss is then defined as the cross-entropy between the target and predicted top-one probability distributions:

$$\mathcal{L}_{\text{ListNet}} = - \sum_{i=1}^k P(i | r) \log P(i | s). \quad (24)$$

Since the ground truth distribution  $P(\cdot | r)$  is fixed, its entropy does not affect the optimization objective. Therefore, minimizing the cross-entropy is equivalent to making the model-predicted ranking distribution  $P(\cdot | s)$  closer to the reference distribution  $P(\cdot | r)$ , enabling the model to better learn the correct ranking relationships.

2) *Rank-Decay ListNet*: While ListNet leverages top-one probability distributions derived from similarity scores to learn list-wise ranking preferences, it implicitly treats all candidates equally, regardless of their importance in the final ranking. However, in practical retrieval scenarios, ranking errors among the top candidates are often more critical than those among lower-ranked ones. To better reflect this intuition during training, we introduce Rank-Decay ListNet (RD-ListNet), which incorporates rank-aware weights into the ListNet loss.

Specifically, we first obtain a relevance-based permutation  $\mathbf{r}' = [r'_1, r'_2, \dots, r'_k]$  by sorting the ground-truth similarity scores  $\mathbf{r}$  in descending order. The corresponding rank-aware decay weights are then computed using a logarithmic function:

$$w_i = \frac{1}{\log_2(i + 1)}, \quad (25)$$

where  $i$  denotes the position index in the sorted label sequence, with smaller  $i$  indicating higher relevance. These decay weights are used to scale the cross-entropy loss between the predicted and ground-truth top-one distributions:

$$\mathcal{L}_{\text{RD-ListNet}} = - \sum_{i=1}^k w_i \cdot P(i | \mathbf{r}) \log P(i | \mathbf{s}), \quad (26)$$

where  $P(i | \mathbf{s})$  and  $P(i | \mathbf{r})$  are the softmax-normalized similarity scores and heuristic labels, respectively. By assigning higher weights to top-ranked items, RD-ListNet encourages the model to focus more on accurately ranking the most relevant trajectories.

3) *Total Loss*: To jointly leverage both point-wise and list-wise supervision signals, we combine the mean squared error loss with overall ranking-aware regularization. The overall training loss during fine-tuning is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \gamma_1 \mathcal{L}_{\text{ListNet}} + \gamma_2 \mathcal{L}_{\text{RD-ListNet}}, \quad (27)$$

where  $\gamma_1$  and  $\gamma_2$  are weighting hyperparameters that balance the contributions of the two list-wise components during training. We adopt this joint loss formulation during the fine-tuning stage, enabling the model to refine its similarity predictions with respect to both absolute magnitude and relative order. In our experiments, we find that this hybrid optimization strategy leads to improved generalization and more stable query performance, particularly in noisy or ambiguous scenarios where point-wise supervision alone may be inadequate.

### D. Model Fine-Tuning

We define the fine-tuning pseudo-code as Algorithm 1, where dual-scale features are iteratively aligned and fused through  $L$  stacked SALayers, followed by supervised optimization with total loss. Specifically, given two inputs from different scales  $\mathbf{e}_{\text{gps}}$  and  $\mathbf{e}_{\text{grid}}$ , we first project them into a unified embedding space using the pre-encoding module  $\mathbf{PE}$ , as shown in Line 2, obtaining initial representations  $\mathbf{Z}_{\text{gps}}^{(0)}$  and  $\mathbf{Z}_{\text{grid}}^{(0)}$ . These embeddings are then passed through  $L$  layers of SALayer, where Dual SAA is performed iteratively to produce the final representations  $\mathbf{Z}_{\text{gps}}^{(L)}$  and  $\mathbf{Z}_{\text{grid}}^{(L)}$ , as illustrated

in Lines 3–5. Subsequently, these two outputs are fused using a hyperparameter  $\epsilon$  and aggregated via mean pooling to obtain the final embedding  $\mathbf{h}$ . Then  $\mathbf{h}$  is used to compute the predicted similarity matrix  $\mathbf{H}_p$ , which is then compared against the ground truth  $\mathbf{H}$  to calculate the total loss, as defined in Line 9. Finally, the model is updated via backpropagation.

#### Algorithm 1 The Fine-Tuning Process of TrajDiff

**Input** Different scale representations  $\mathbf{e}_{\text{gps}}$  and  $\mathbf{e}_{\text{grid}}$ , per-encoding module **PE**, ground truth  $\mathbf{H}$ , pretrained SAM  
**Output** Embedding results  $\mathbf{h}$

- 1: **Aggregate two scale inputs using the SAM;**
- 2: Project the initial input
- $\mathbf{Z}_{\text{gps}}^{(0)} \leftarrow \text{pe}(\mathbf{z}_{\text{gps}})$ ,  $\mathbf{Z}_{\text{grid}}^{(0)} \leftarrow \text{pe}(\mathbf{z}_{\text{grid}})$ ;
- 3: **for**  $l \in [1, L]$  **do**
- 4:     Duplicate  $\mathbf{Z}_{\text{gps}}^{(l-1)}$  and  $\mathbf{Z}_{\text{grid}}^{(l-1)}$   
 $\mathbf{Z}_{\text{gps}}^{(l-1)'} \leftarrow \mathbf{Z}_{\text{gps}}^{(l-1)}$ ,  $\mathbf{Z}_{\text{grid}}^{(l-1)'} \leftarrow \mathbf{Z}_{\text{grid}}^{(l-1)}$ ;
- 5:     Execute the Dual Aggregation Attention  
 $\mathbf{Z}_{\text{gps}}^{(l)} \leftarrow \text{SAA}(\mathbf{Z}_{\text{gps}}^{(l-1)}, \mathbf{Z}_{\text{grid}}^{(l-1)})$   
 $\mathbf{Z}_{\text{grid}}^{(l)} \leftarrow \text{SAA}(\mathbf{Z}_{\text{grid}}^{(l-1)}, \mathbf{Z}_{\text{gps}}^{(l-1)'})$ ;
- 6: **end for**
- 7: Fuse two embeddings  
 $\mathbf{h} \leftarrow \epsilon \mathbf{Z}_{\text{gps}}^{(L)} + (1 - \epsilon) \mathbf{Z}_{\text{grid}}^{(L)}$   
 $\mathbf{h} \leftarrow \text{mean\_pooling}(\mathbf{h})$ ;
- 8: Calculate embedding distance  $\mathbf{H}_p$  from  $\mathbf{h}$ ;
- 9: Calculate total loss using  $\mathbf{H}_p$  and  $\mathbf{H}$   
 $\mathcal{L} \leftarrow \mathcal{L}_{\text{MSE}} + \gamma_1 \mathcal{L}_{\text{ListNet}} + \gamma_2 \mathcal{L}_{\text{RD-ListNet}}$ ;
- 10: Backpropagation and update parameters in TrajDiff;
- 11: **return**  $\mathbf{h}$

## V. EXPERIMENTS

In this section, we first present the experimental setup, including the datasets, baseline methods, parameter configurations, evaluation metrics, and implementation environment. We then conduct extensive experiments to evaluate the effectiveness of our proposed model, TrajDiff, aiming to address the following research questions:

- **RQ1:** Can TrajDiff achieves superior performance in trajectory similarity computation compared to several state-of-the-art baselines?
- **RQ2:** How does each module of TrajDiff contribute to its overall performance?
- **RQ3:** What is the impact of the major hyperparameters of TrajDiff on the similarity computation?
- **RQ4:** How does TrajDiff perform in terms of efficiency?

### A. Experimental Settings

1) **Datasets:** We evaluate our method on three publicly available datasets:

(1) **Proto**<sup>1</sup> contains 1.7 million taxi trajectories from Porto, Portugal (July 2013 – June 2014);

(2) **Geolife**<sup>2</sup> [48] includes trajectories of 182 users in

Beijing (April 2007 – August 2012);

(3) **T-Drive**<sup>3</sup> [49] provides GPS trajectories of 10,357 taxis in Beijing from February 2 to 8, 2008.

In line with prior work [15], we remove trajectories outside the geographic boundaries and those that do not meet the required length criteria. Since both Geolife and T-Drive are based in Beijing, we apply the same geographic boundary for consistency.

For T-Drive, since each trajectory contains data from many days, we segment multi-day taxi trajectories into sub-trajectories by identifying stay points, which are locations where a vehicle remains within a 100-meter radius for over 5 minutes. Additional details of the datasets are summarized in Tab. II.

For the Porto dataset, we select the first 200,000 trajectories for pre-training and additionally sample 10,000 trajectories for similarity computation. For both the T-Drive and Geolife datasets, we randomly sample 10,000 trajectories to compute pairwise similarities. Due to the smaller size of the Geolife and T-Drive datasets, we fine-tune the model pre-trained on the Porto dataset separately on these two datasets. The trajectories used for similarity computation are split into training, evaluation, and test sets with a ratio of 7:1:2.

TABLE II: Detailed statistics of the dataset.

Dataset	Porto	Geolife	T-Driven
Trajectories ( $n$ )	1,372,725	10,940	28,843
min length	20	20	20
max length	200	300	200
mean length	48	106	38
longitude	[-8.519, -8.005]	[116.25, 116.5]	[116.25, 116.5]
latitude	[41.1001, 41.2086]	[39.8, 40.1]	[39.8, 40.1]

2) **Baseline:** We focus primarily on trajectory similarity computation with the goal of more accurately approximating heuristic measures. To validate the effectiveness of our model, we compare TrajDiff with eight relevant methods.

- **t2vec** [35]: It learns trajectory embeddings via seq2seq modeling, enabling fast and noise-robust similarity computation.
- **NeuTraj** [50]: It is a neural metric learning model that approximates trajectory similarity using spatial attention and seed-guided supervision in linear time.
- **Traj2SimVec** [30]: It learns trajectory similarity through deep representation learning with auxiliary supervision, leveraging sub-trajectory distance loss and point matching loss for improved robustness and scalability.
- **T3S** [27]: It learns trajectory embeddings by combining LSTM and self-attention to capture spatial and structural features for similarity computation.
- **TMN** [12]: It approximates trajectory similarity via attention-based point matching between trajectory pairs.
- **TrajGAT** [21]: It models trajectory similarity using a graph-based Transformer with hierarchical spatial encoding for long-range dependency capture.

<sup>1</sup><https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data>

<sup>2</sup><https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/>

<sup>3</sup><https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/>

- **TrajCL [15]**: It employs contrastive learning with dual-feature self-attention and trajectory augmentations to efficiently learn unsupervised trajectory similarity representations with strong generalization.
- **KGTS [20]**: It proposes a knowledge-guided spatiotemporal graph neural network for traffic forecasting, which enhances prediction accuracy by incorporating external knowledge to guide the construction and modeling of graph structures, thereby capturing complex spatiotemporal dependencies more effectively.

3) *Parameter Settings*: We set the batch size to 128, the learning rate to 0.001, and the number of attention heads to 16. The experimental results were obtained using random seeds 18, 66, and 108, with each experiment conducted three times and the average value taken. During fine-tuning, we set  $\gamma_1 = 0.1$  and  $\gamma_2 = 0.001$ , and the number of SAM layers is set to 1. We pretrain for 20 epochs with early stopping patience of 5 epochs, and fine-tune for 30 epochs with early stopping patience of 10 epochs.

4) *Evaluation Metrics and Environment Implementation*: Following previous works [15], we adopt the top- $k$  hit ratio (HR@ $k$ ) and top- $k$  recall for top- $t$  ground truth (Recall- $t$ @ $k$ ) as our evaluation metrics. Both metrics assess the effectiveness of models in top- $k$  similarity search tasks. Specifically, HR@ $k$  measures the overlap between the top- $k$  trajectories retrieved using approximate similarity and those retrieved using ground-truth similarity. Recall- $t$ @ $k$ , on the other hand, evaluates the model's ability to retrieve the top- $t$  ground-truth items within the top- $k$  results. In this paper, we primarily report HR@1, HR@5, HR@20, and H5@20. Experiments are conducted on dual-socket AMD EPYC 7402 CPUs (2 × 24 cores, 96 threads) @ 2.80GHz and a GeForce RTX 4090 GPU.

### B. Performance Comparison (RQ1)

Table III presents a comprehensive performance comparison across various methods on three benchmark datasets. Our proposed model, TrajDiff, consistently achieves the best performance. Specifically, the average improvements in HR@1 across the three heuristic measure are 53.57%, 22.91%, and 23.71%, respectively, demonstrating the superiority of the TrajDiff in trajectory similarity computation.

First, existing studies (e.g., TrajCL, KGTS, TrajGAT, and T3S) primarily focus on single-scale trajectory representations, such as GPS-based features or grid-based features. In addition, these methods often rely on simplistic feature fusion strategies, including direct embedding summation or independent attention score masking. Such designs limit the ability of the model to fully exploit the complementary information inherent in dual-scale trajectory data, thereby resulting in suboptimal performance. In contrast, our SAM explicitly models the correlations between dual-scale features to achieve effective semantic alignment, leading to more expressive and discriminative trajectory representations. As shown in Table III, our method consistently outperforms the aforementioned baselines across all heuristic metric. To further verify the effectiveness of the SAM, we replace it with the fusion strategies employed

in TrajCL and T3S and evaluate the performance on the Porto dataset using three heuristic metric. We summarize the results in Table IV, indicating that substituting our SAM with the fusion strategies from TrajCL and T3S consistently leads to performance degradation across all three heuristic metrics. Specifically, the HR@1 scores exhibit average decreases of 50.32%, 40.52%, and 27.3% under three heuristic metrics. These findings demonstrate that the proposed SAM effectively mitigates semantic inconsistency and enhances the integration of dual-scale trajectory features.

Second, we pretrain our model using the denoising diffusion bridge, which enables it to learn a stochastic transformation between paired trajectories and enhances its robustness to noise. Importantly, instead of computing loss in the raw data space, we optimize the model in a high-dimensional embedding space. This allows the model to more effectively suppress local noise while preserving global semantic structure, resulting in more accurate and stable representations. As shown in Table III, our method achieves superior performance, particularly on high-noise datasets such as T-Driver and GeoLife. In contrast, on the Porto dataset where the trajectory data is relatively clean, the performance gain is moderate, reflecting the reduced necessity for denoising. This variation across datasets highlights the effectiveness of our denoising strategy in recovering informative representations from corrupted inputs. Overall, these findings confirm the advantage of incorporating a denoising diffusion bridge during pretraining and optimizing in the embedding space, equipping the model with improved resilience and generalization under noisy or ambiguous conditions.

Finally, most existing methods primarily rely on point-wise or list-wise losses as supervision signals, overlooking the global ranking structure across the entire trajectory dataset. In contrast, our approach explicitly integrates global ranking information via the proposed overall ranking-aware regularization, which contributes significantly to its superior performance over all baseline methods. To further verify the effectiveness of overall ranking-aware regularization, we incorporate it into the training processes of TrajCL and T3S by augmenting their loss functions, and evaluate its performance under the SSPD distance metric. As shown in Fig. 3, the overall ranking-aware regularization leads to consistent improvements in the HR@1 across all three datasets, with gains of 23.52%, 55.69%, and 21.24%, respectively. In addition to accuracy improvements, Fig. 7 also shows that the incorporation of overall ranking-aware regularization accelerates model convergence, reducing the number of training epochs required.

Overall, these findings underscore the general applicability and effectiveness of our proposed method. Furthermore, the consistent improvements observed when integrating our components into existing models provide strong evidence of their standalone utility.

### C. Ablation Study (RQ2)

To evaluate the effectiveness of the key components, we conduct a detailed ablation experiment on TrajDiff by remov-

TABLE III: Performance comparison of the proposed model and different baselines. Marker \* indicates the results are statistically significant (t-test with p-value < 0.01).

Dataset	Method	SSPD				Discrete Frechet				Hausdorff			
		HR@1	HR@5	HR@20	R5@20	HR@1	HR@5	HR@20	R5@20	HR@1	HR@5	HR@20	R5@20
Porto	t2vec	0.271	0.348	0.504	0.675	0.467	0.519	0.667	0.889	0.359	0.410	0.564	0.793
	Traj2SimVec	0.313	0.401	0.487	0.723	0.471	0.524	0.671	0.893	0.301	0.342	0.423	0.551
	T3S	0.288	0.412	0.534	0.763	0.461	0.581	0.723	0.946	0.490	0.610	0.691	0.941
	TMN	0.323	0.413	0.497	0.746	0.473	0.568	0.656	0.925	0.481	0.573	0.661	0.893
	TrajGAT	0.331	0.426	0.590	0.691	0.330	0.352	0.399	0.694	0.568	0.683	0.739	0.961
	TrajCL	<u>0.376</u>	<u>0.495</u>	<u>0.583</u>	<u>0.821</u>	<u>0.493</u>	<u>0.618</u>	<u>0.740</u>	<u>0.955</u>	0.516	0.645	0.721	0.955
	KGTS	0.301	0.369	0.537	0.680	0.436	0.531	0.658	0.876	0.431	0.528	0.607	0.817
	<b>Ours</b>	<b>0.485*</b>	<b>0.573*</b>	<b>0.657*</b>	<b>0.894*</b>	<b>0.541*</b>	<b>0.664*</b>	<b>0.762*</b>	<b>0.971*</b>	<b>0.583*</b>	<b>0.685*</b>	<b>0.747*</b>	<b>0.969*</b>
<i>Improvement</i>		28.98%	15.76%	12.69%	8.89%	9.74%	7.44%	2.97%	1.68%	2.64%	0.29%	1.08%	0.83%
Geolife	t2vec	0.240	0.378	0.513	0.707	0.402	0.451	0.601	0.782	0.302	0.360	0.521	0.736
	Traj2SimVec	0.264	0.398	0.542	0.749	0.432	0.482	0.592	0.772	0.254	0.302	0.383	0.511
	T3S	0.254	0.401	0.557	0.773	0.342	0.482	0.625	0.802	0.244	0.322	0.444	0.633
	TMN	0.268	0.399	0.546	0.769	0.385	0.501	0.624	0.794	0.373	0.523	0.681	0.845
	TrajGAT	0.281	0.413	0.554	<u>0.796</u>	0.392	0.531	0.712	0.901	<u>0.418</u>	0.584	0.734	0.904
	TrajCL	<u>0.301</u>	<u>0.443</u>	<u>0.573</u>	0.792	<u>0.400</u>	<u>0.526</u>	<u>0.704</u>	<u>0.891</u>	0.410	<u>0.587</u>	<u>0.736</u>	<u>0.906</u>
	KGTS	0.273	0.405	0.521	0.763	0.391	0.468	0.631	0.803	0.334	0.381	0.549	0.729
	<b>Ours</b>	<b>0.420*</b>	<b>0.541*</b>	<b>0.651*</b>	<b>0.847*</b>	<b>0.507*</b>	<b>0.646*</b>	<b>0.742*</b>	<b>0.929*</b>	<b>0.579*</b>	<b>0.693*</b>	<b>0.778*</b>	<b>0.953*</b>
<i>Improvement</i>		39.53%	22.12%	13.61%	6.40%	26.75%	22.81%	5.40%	4.26%	38.52%	18.06%	5.71%	5.19%
T-Driver	t2vec	0.137	0.278	0.414	0.607	0.193	0.381	0.401	0.702	0.214	0.303	0.485	0.709
	Traj2SimVec	0.139	0.284	0.426	0.611	0.244	0.432	0.525	0.747	0.254	0.302	0.383	0.511
	T3S	0.134	0.265	0.411	0.567	0.222	0.401	0.497	0.732	0.324	0.472	0.621	0.833
	TMN	0.148	0.281	0.426	0.609	0.276	0.441	0.547	0.794	0.325	0.489	0.631	0.845
	TrajGAT	0.161	0.296	0.431	0.616	0.296	0.451	0.579	0.829	0.330	0.519	0.649	0.902
	TrajCL	<u>0.167</u>	<u>0.302</u>	<u>0.449</u>	<u>0.629</u>	<u>0.307</u>	<u>0.464</u>	<u>0.581</u>	<u>0.826</u>	<u>0.337</u>	<u>0.524</u>	<u>0.651</u>	<u>0.906</u>
	KGTS	0.143	0.294	0.437	0.619	0.286	0.425	0.553	0.812	0.301	0.4571	0.609	0.889
	<b>Ours</b>	<b>0.321*</b>	<b>0.461*</b>	<b>0.577*</b>	<b>0.794*</b>	<b>0.406*</b>	<b>0.542*</b>	<b>0.660*</b>	<b>0.912*</b>	<b>0.438*</b>	<b>0.591*</b>	<b>0.699*</b>	<b>0.933*</b>
<i>Improvement</i>		92.22%	52.64%	28.51%	26.23%	32.25%	16.81%	13.60%	10.01%	29.97%	12.79%	7.37%	2.98%

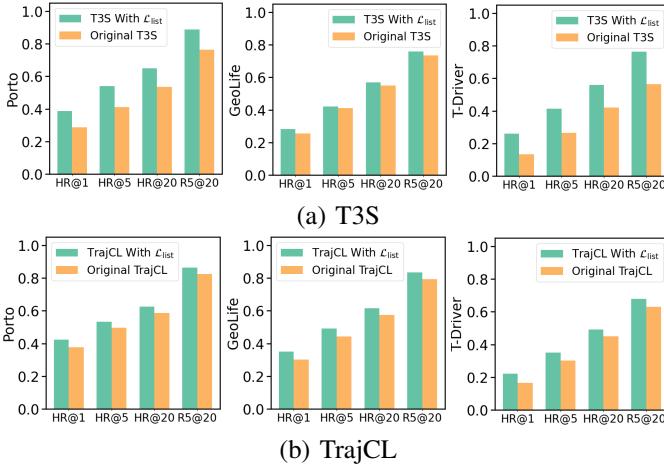


Fig. 3: Performance improvement T3S/TrajCL with our  $\mathcal{L}_{\text{list}}$ .

TABLE IV: Comparison of different feature fusion methods.  
Abbr.: AM: Attention Mask.

Metric	Method	HR@1	HR@5	HR@20	R5@20
Hausdorff	Sum	0.359	0.518	0.641	0.859
	AM	0.414	0.553	0.647	0.898
	ours	<b>0.583</b>	<b>0.685</b>	<b>0.747</b>	<b>0.969</b>
Discrete Fréchet	Sum	0.370	0.515	0.654	0.843
	AM	0.409	0.558	0.705	0.902
	ours	<b>0.546</b>	<b>0.661</b>	<b>0.759</b>	<b>0.968</b>
SSPD	Sum	0.388	0.539	0.649	0.887
	AM	0.375	0.502	0.602	0.838
	ours	<b>0.486</b>	<b>0.573</b>	<b>0.658</b>	<b>0.894</b>

ing different components to get different variants.

- **w/o Bridge:** remove the DDBM pre-train.
- **w/o SAM:** remove the semantic alignment module and use the vanilla Transformer instead.
- **w/o  $\mathcal{L}_{\text{list}}$ :** remove the overall ranking-aware regularization and use only the MSE loss.
- **w/o  $Z_{\text{gps}}$ :** only return  $Z_{\text{grid}}$  as the trajectory final embedding.
- **w/o  $Z_{\text{grid}}$ :** only return  $Z_{\text{gps}}$  as the trajectory final embedding.

The results of the ablation study are shown in Fig. 4. The performance of the model significantly declined after removing each module, demonstrating the effectiveness of each module.

First, **w/o SAM** shows the most pronounced performance degradation across all heuristic evaluation metrics and datasets. For example, on the Porto dataset, the HR@1 score under the SSPD distance drops by 11.8%, highlighting the pivotal role of SAM in trajectory similarity learning. This substantial decline demonstrates that removing SAM significantly hampers the model's ability to align representations across granularities. Notably, naive fusion strategies fail to capture the complex semantic correspondences between coarse-grained and fine-grained views. Without explicitly modeling cross-scale interactions, the model struggles to generate embeddings that faithfully reflect true trajectory similarity.

Second, the performance of **w/o DDBM** shows a more pronounced decline on the GeoLife and T-Drive datasets, both of which contain higher levels of noise, compared to the Porto dataset. This highlights the crucial role of DDBM-based pre-

training in enhancing model robustness under noisy conditions. By simulating a stochastic bridge process between any pair of trajectories, DDBM enables the model to learn noise-resilient pairwise representations. More importantly, our pretraining loss is computed in the high-dimensional embedding space, rather than in the original data space. This design allows the model to bypass local distortions or noise in raw trajectories and focus instead on optimizing meaningful representations in the learned feature space.

In addition, **w/o  $\mathcal{L}_{\text{List}}$**  significantly compromises the effectiveness of the SSPD evaluation metric compared to Hausdorff and Discrete Fréchet distances, resulting in an average decrease of 18.87% in HR@1 across the three datasets. This is because SSPD does not require precise point-to-point alignment but instead focuses on whether trajectories exhibit similar global path patterns. In comparison, our overall ranking-aware constraint is supervised with ranking as the optimization objective, which is highly aligned with the global trajectory ranking measured by SSPD. Finally, both **w/o  $\mathbf{Z}_{\text{grid}}$**  and  **$\mathbf{Z}_{\text{gps}}$**  lead to a moderate decline in performance. This is because, compared to the dual semantic alignment attention, a single semantic alignment attention can only achieve limited alignment. It fails to fully explore the bidirectional interaction across different granularities and overlooks the potential impact of reversing their roles.

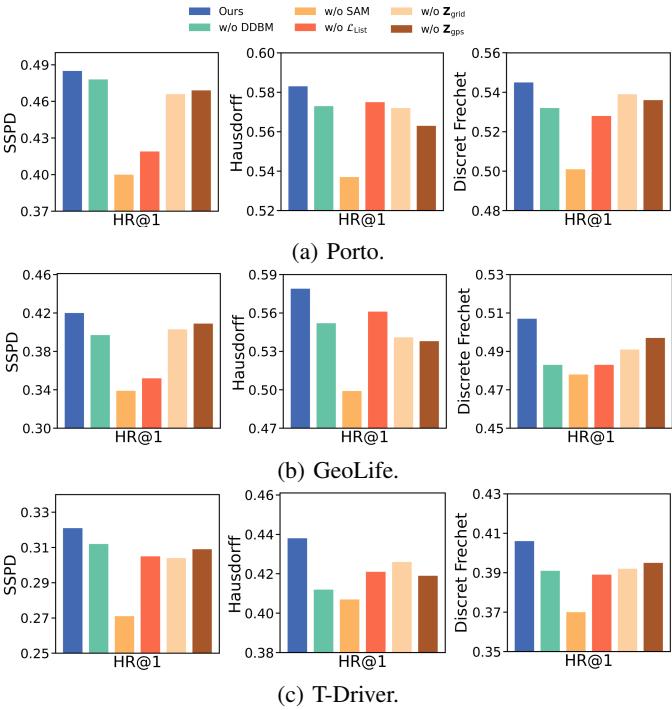


Fig. 4: Ablation study by removing different components.

#### D. Impact of Hyper-parameters (RQ3)

As shown in Fig. 5, we further conduct a sensitivity analysis of critical hyperparameters on three evaluation metrics of the Porto dataset and report the results on the HR@1 metric.

Initially, larger values of  $\gamma_1$  and  $\gamma_2$  cause the model to overlook non-global information. As these parameters decrease, a balance between local ranking information and global information is achieved, resulting in improved performance. However, when  $\gamma_1$  and  $\gamma_2$  become too small, their influence is neglected by the model, leading to a decline in performance.

The performance is optimal when  $\mu = 0.5$ . This is because the two embeddings represent complementary information, and balancing them maximizes their complementary nature, allowing the model to comprehensively understand different aspects of the data. Over-relying on one aspect of the information may cause the model to lose other important features.

When the initial cell size is small, for example, less than 100, changing the grid size does not significantly affect the model's performance. This is because smaller grids contain less overall trajectory structure information. However, excessively large grids can lead to the loss of structural information. Therefore, the model performs optimally when the cell size is set to 100.

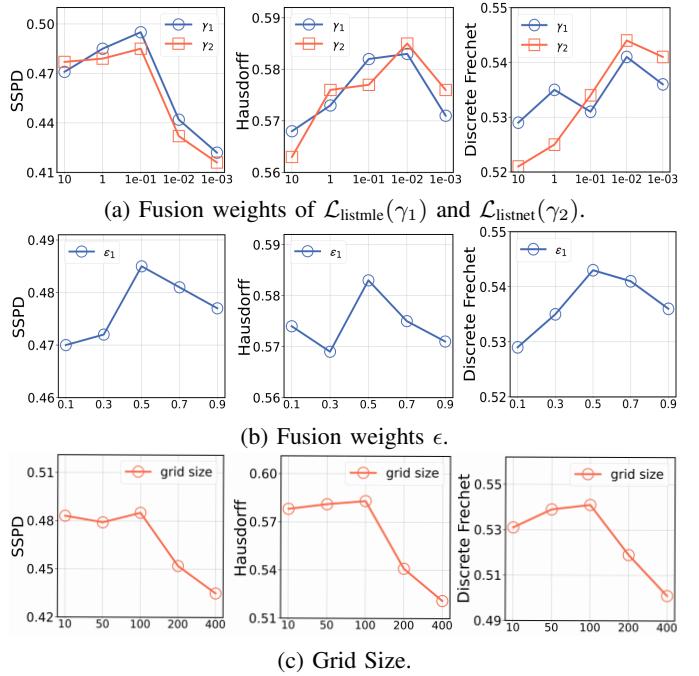


Fig. 5: The impact of hyperparameters on the performance of TrajDiff.

#### E. Efficiency Comparison (RQ5)

1) *Time Complexity Analysis*: To support various distance metrics, our framework employs different pre-encoding strategies: a simple linear projection for SSPD and Hausdorff distances, and an LSTM encoder for the discrete Fréchet distance. For a unified and conservative time complexity analysis, we focus on the LSTM-based pre-encoding, as it represents the most computationally demanding scenario.

The LSTM encoder processes sequences of length  $n$  with an embedding dimension  $d$ , resulting in a time complexity of  $\mathcal{O}(n \cdot d^2)$ . This reflects the recurrent nature of LSTMs, where each time step involves matrix multiplications of size  $d \times d$ .

Following pre-encoding, SAM is applied, which comprises several key components: **(1) Linear projections:** for query, key, and value, each with complexity  $\mathcal{O}(n \cdot d^2)$ . **(2) Attention score computations:** including both self- and cross-attention, with complexity  $\mathcal{O}(n^2 \cdot d)$  due to pairwise interactions between sequence elements. **(3) Feed-forward networks:** applied to each sequence element, with complexity  $\mathcal{O}(n \cdot d \cdot d_{\text{hid}})$ , where  $d_{\text{hid}}$  denotes the hidden dimension. Aggregating the complexities of all components, the overall time complexity for a single pass through the module is  $\mathcal{O}(n \cdot d^2 + n^2 \cdot d + n \cdot d \cdot d_{\text{hid}})$ . Here,  $n$  is the trajectory length,  $d$  is the embedding dimension, and  $d_{\text{hid}}$  is the hidden size of the feed-forward layers. This analysis demonstrates that the most computationally intensive part arises from the attention mechanism, especially for longer sequences, while the feed-forward and projection layers contribute linearly for  $n$ .

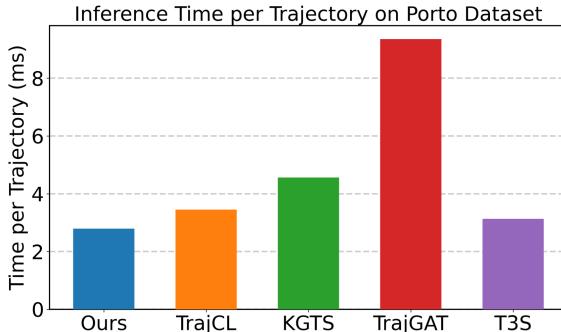


Fig. 6: Comparison of inference speed across diverse models.

**2) Inference Speed Comparison:** We compare the inference time of our method with several baseline approaches on the Porto dataset, as shown in Fig. 6. Our model achieves a speed improvement of 10.86% compared to the second-fastest method, demonstrating its superior efficiency. Compared to TrajCL, which employs two separate Transformer encoders to process spatial and structural information independently and sets the model depth to 2, our model is significantly faster. This efficiency gain stems from our unified architecture that avoids redundant computations across modalities, and from the use of a single-layer SAM, which further reduces computational overhead without sacrificing performance. Although T3S shares a similar architecture with our model, its LSTM input dimension is equal to the hidden dimension of our LSTM. This leads to increased computational overhead for T3S and contributes to its slower inference speed. KGTS and TrajGAT face significant computational challenges due to the large geographic area involved. As the geographical region expands, it leads to the creation of numerous sub-grids. This results in the graph neural network computations involving a vast number of edges and nodes, which substantially increases the time consumption. These results confirm that our model strikes a favorable balance between representational capacity and computational efficiency, making it well-suited for real-world deployment scenarios where both accuracy and speed are critical.

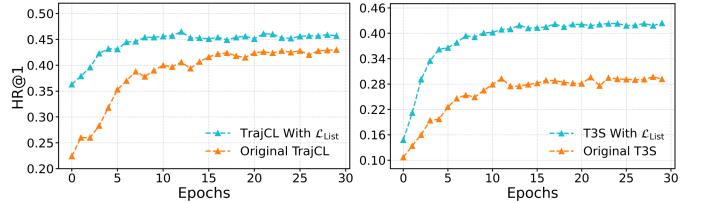


Fig. 7: Convergence speed of T3S/TrajCL with our  $\mathcal{L}_{\text{List}}$ .

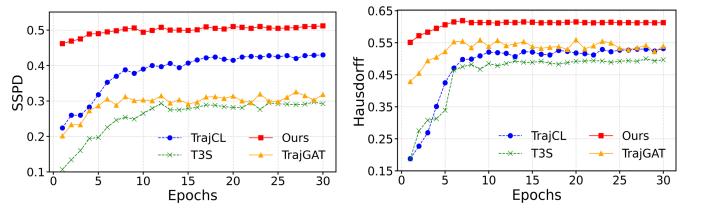


Fig. 8: Convergence speed of different methods.

**3) Convergence Speed Comparison:** We evaluate and compare the convergence speeds of various representative trajectory learning methods on the Porto dataset, utilizing three widely adopted evaluation metrics to ensure a comprehensive and fair assessment, as shown in Fig. 8. The experimental results clearly demonstrate that our proposed method not only converges significantly faster but also achieves a consistently higher final performance when compared to state-of-the-art baselines, including TrajCL, T3S, and TrajGAT. In particular, our method exhibits a rapid performance gain within the initial few training epochs, reaching a stable state with minimal fluctuations and superior metric scores. This contrasts sharply with the baseline methods, which tend to require a substantially larger number of epochs to achieve convergence. The accelerated convergence observed in our method can be primarily attributed to the effectiveness of our pretraining strategy, which enables the model to capture semantically rich and structurally meaningful trajectory representations from the outset. By initializing the model with well-informed parameters, our approach reduces the burden of learning from scratch and thereby shortens the training cycle.

## VI. CONCLUSION

In this paper, we propose a novel trajectory similarity computation framework, TrajDiff. The model incorporates a semantic alignment module to bridge the semantic gap between features at two distinct scales, integrating them into a unified and robust embedding. Additionally, a denoising diffusion bridge model is employed to pretrain the model in the embedding space, enabling it to learn the interactions and transfer patterns between arbitrary pairs of trajectories, thereby enhancing its robustness and generalization capability. Furthermore, we introduce a global ranking-aware constraint to improve performance on ranking-sensitive evaluation metrics. Experimental results on three public datasets demonstrate the effectiveness and superiority of our method in trajectory similarity computation.

## REFERENCES

- [1] P. K. Agarwal, K. Fox, K. Munagala, A. Nath, J. Pan, and E. Taylor, “Subtrajectory clustering: Models and algorithms,” in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, 2018, pp. 75–87.
- [2] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, and X. Qin, “Fast large-scale trajectory clustering,” *Proceedings of the VLDB Endowment*, vol. 13, no. 1, pp. 29–42, 2019.
- [3] C.-C. Hung, W.-C. Peng, and W.-C. Lee, “Clustering and aggregating clues of trajectories for mining trajectory patterns and routes,” *The VLDB Journal*, vol. 24, pp. 169–192, 2015.
- [4] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, “A survey of trajectory distance measures and performance evaluation,” *The VLDB Journal*, vol. 29, pp. 3–32, 2020.
- [5] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, “When will you arrive? estimating travel time based on deep neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [6] J. Kim and H. S. Mahmassani, “Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories,” *Transportation Research Procedia*, vol. 9, pp. 164–184, 2015.
- [7] J. Yuan, Y. Zheng, and X. Xie, “Discovering regions of different functions in a city using human mobility and pois,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 186–194.
- [8] C. Yang, Z. Zhang, Z. Fan, R. Jiang, Q. Chen, X. Song, and R. Shibasaki, “Epimob: Interactive visual analytics of citywide human mobility restrictions for epidemic control,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 8, pp. 3586–3601, 2022.
- [9] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, “Trajectory clustering via deep representation learning,” in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 3880–3887.
- [10] D. Yao, C. Zhang, Z. Zhu, Q. Hu, Z. Wang, J. Huang, and J. Bi, “Learning deep representation for trajectory clustering,” *Expert Systems*, vol. 35, no. 2, p. e12252, 2018.
- [11] S. Wang, Z. Bao, J. S. Culpepper, and G. Cong, “A survey on trajectory data management, analytics, and learning,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–36, 2021.
- [12] P. Yang, H. Wang, D. Lian, Y. Zhang, L. Qin, and W. Zhang, “Tmn: trajectory matching networks for predicting similarity,” in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1700–1713.
- [13] A. Belhadi, Y. Djennouri, J. C.-W. Lin, and A. Cano, “Trajectory outlier detection: Algorithms, taxonomies, evaluation, and open challenges,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 11, no. 3, pp. 1–29, 2020.
- [14] F. Meng, G. Yuan, S. Lv, Z. Wang, and S. Xia, “An overview on trajectory outlier detection,” *Artificial Intelligence Review*, vol. 52, pp. 2437–2456, 2019.
- [15] Y. Chang, J. Qi, Y. Liang, and E. Tanin, “Contrastive trajectory similarity learning with dual-feature attention,” in *2023 IEEE 39th International conference on data engineering (ICDE)*. IEEE, 2023, pp. 2933–2945.
- [16] S. Zhou, J. Li, H. Wang, S. Shang, and P. Han, “Grlstm: trajectory similarity computation with graph-based residual lstm,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4972–4980.
- [17] T. Eiter and H. Mannila, “Computing discrete fréchet distance,” 1994.
- [18] Y. Tao, A. Both, R. I. Silveira, K. Buchin, S. Sijben, R. S. Purves, P. Laube, D. Peng, K. Toohey, and M. Duckham, “A comparative analysis of trajectory similarity measures,” *GIScience & Remote Sensing*, vol. 58, no. 5, pp. 643–669, 2021.
- [19] H. Yuan and G. Li, “A survey of traffic prediction: from spatio-temporal data to intelligent transportation,” *Data Science and Engineering*, vol. 6, no. 1, pp. 63–85, 2021.
- [20] Z. Chen, D. Zhang, S. Feng, K. Chen, L. Chen, P. Han, and S. Shang, “Kgts: contrastive trajectory similarity learning over prompt knowledge graph embedding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, 2024, pp. 8311–8319.
- [21] D. Yao, H. Hu, L. Du, G. Cong, S. Han, and J. Bi, “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*, 2022, pp. 2275–2285.
- [22] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
- [23] H. Samet, “An overview of quadtrees, octrees, and related hierarchical data structures,” *Theoretical Foundations of Computer Graphics and CAD*, pp. 51–68, 1988.
- [24] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” *arXiv preprint arXiv:1902.10197*, 2019.
- [25] S. Zhou, J. Li, H. Wang, S. Shang, and P. Han, “Grlstm: trajectory similarity computation with graph-based residual lstm,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4972–4980.
- [26] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, no. 1, 2014.
- [27] P. Yang, H. Wang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, “T3s: Effective representation learning for trajectory similarity computation,” in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2183–2188.
- [28] S. Hochreiter, “Long short-term memory,” *Neural Computation MIT-Press*, 1997.
- [29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [30] H. Zhang, X. Zhang, Q. Jiang, B. Zheng, Z. Sun, W. Sun, and C. Wang, “Trajectory similarity learning with auxiliary supervision and optimal matching,” 2020.
- [31] Y. Cao, L. Li, X. Chen, X. Xu, Z. Huang, and Y. Yu, “Hypergraph hash learning for efficient trajectory similarity computation,” in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 175–186.
- [32] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, “Hypergraph neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3558–3565.
- [33] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, “Bootstrap your own latent-a new approach to self-supervised learning,” *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.
- [34] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *International conference on machine learning*. PMLR, 2016, pp. 1747–1756.
- [35] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, “Deep representation learning for trajectory similarity computation,” in *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 2018, pp. 617–628.
- [36] I. Sutskever, “Sequence to sequence learning with neural networks,” *arXiv preprint arXiv:1409.3215*, 2014.
- [37] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [38] L. Deng, Y. Zhao, Z. Fu, H. Sun, S. Liu, and K. Zheng, “Efficient trajectory similarity computation with contrastive learning,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 365–374.
- [39] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [40] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.
- [41] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [42] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in *9th International Conference on Learning Representations, ICLR 2021*.
- [43] J. L. Doob and J. Doob, *Classical potential theory and its probabilistic counterpart*. Springer, 1984, vol. 262.

- [44] L. C. G. Rogers and D. Williams, *Diffusions, Markov processes, and martingales: Itô calculus*. Cambridge university press, 2000, vol. 2.
- [45] T. Karras, M. Aittala, T. Aila, and S. Laine, “Elucidating the design space of diffusion-based generative models,” *Advances in neural information processing systems*, vol. 35, pp. 26\,565–26\,577, 2022.
- [46] L. Zhou, A. Lou, S. Khanna, and S. Ermon, “Denoising diffusion bridge models,” *arXiv preprint arXiv:2309.16948*, 2023.
- [47] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, “Learning to rank: from pairwise approach to listwise approach,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 129–136.
- [48] Y. Zheng, X. Xie, W.-Y. Ma *et al.*, “Geolife: A collaborative social networking service among user, location and trajectory.” *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.
- [49] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, “T-drive: driving directions based on taxi trajectories,” in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99–108.
- [50] D. Yao, G. Cong, C. Zhang, and J. Bi, “Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach,” in *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 2019, pp. 1358–1369.