Region-Point Joint Representation for Effective Trajectory Similarity Learning

Hao Long*, **Silin Zhou***, **Lisi Chen**, **Shuo Shang**†

University of Electronic Science and Technology of China
haolong@std.uestc.edu.cn, {zhousilinxy,jedi.shang}@gmail.com, lchen012@e.ntu.edu.sg

Abstract

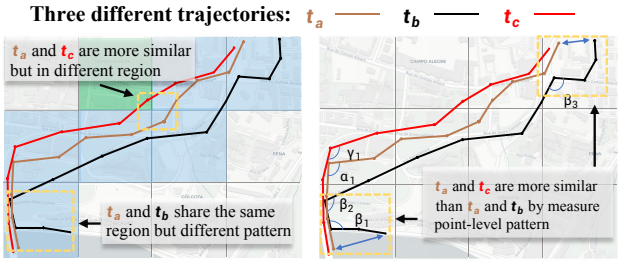
Recent learning-based methods have reduced the computational complexity of traditional trajectory similarity computation, but state-of-the-art (SOTA) methods still fail to leverage the comprehensive spectrum of trajectory information for similarity modeling. To tackle this problem, we propose **RePo**, a novel method that jointly encodes **Re**gion-wise and **Po**int-wise features to capture both spatial context and fine-grained moving patterns. For region-wise representation, the GPS trajectories are first mapped to grid sequences, and spatial context are captured by structural features and semantic context enriched by visual features. For point-wise representation, three lightweight expert networks extract local, correlation, and continuous movement patterns from dense GPS sequences. Then, a router network adaptively fuses the learned point-wise features, which are subsequently combined with region-wise features using cross-attention to produce the final trajectory embedding. To train RePo, we adopt a contrastive loss with hard negative samples to provide similarity ranking supervision. Experiment results show that RePo achieves an average accuracy improvement of 22.2% over SOTA baselines across all evaluation metrics.

Code — https://github.com/Alfred4c/RePo
Extended version — https://arxiv.org/abs/2511.13125

1 Introduction

The rapid proliferation of GPS-enabled devices and location-based services have led to the generation of massive amounts of trajectory data (2016; 2025a). Trajectory similarity computation (2018a; 2014) is a fundamental task in trajectory analysis, supporting a wide range of downstream applications, such as trajectory classification (2024), urban planning (2012), and route recommendation (2019).

Many heuristic distance measures, such as Discrete Fréchet (DFD) (2016) and Dynamic Time Warping (DTW) (1998), compute trajectory similarity via exhaustive pairwise comparisons, resulting in poor scalability to large datasets (2024a; 2018b). To address this issue, trajectory similarity learning methods (2018) aims to encode each trajectory into a vector representation, enabling efficient linear-time similarity computation via vector operations.



(a) Similarity on grid trajectory (b) Similarity on GPS trajectory

Figure 1: Comparison of trajectory similarity under grid (region) trajectory and GPS (point) trajectory.

Existing trajectory similarity learning methods first transform GPS trajectories into grid trajectories to reduce irregularity in free space, and then learn trajectory embeddings from the resulting grid sequences (2025b). These approaches can be categorized into three types: RNN-based models, e.g., t2vec (2018), CL-Tsim (2022), leverage recurrent neural networks or contrastive learning to capture sequential dependencies; Graph-based models, e.g., KGTS (2024b), TrajGAT (2022), construct graphs to model spatial relationships or global dependencies among trajectory points; Transformer-based models, e.g., SIMformer (2024), TrajCL (2023), employ self-attention mechanisms for efficient long-range dependency modeling.

While grid-based trajectories help mitigate GPS irregularities in free space and effectively preserve the main structure of trajectories, they inevitably compromise the descriptive richness of trajectory representations. As illustrated in Figure 1(a), mapping trajectories t_a, t_b, and t_c onto grid sequences smooths out local noise and highlights their global spatial structure; for instance, t_a and t_b become identical at the region level after grid mapping. This grid abstraction, however, introduces two key limitations: 1) Although t_a and t_b become identical after grid mapping, t_a is actually more similar to t_c, even though t_c follows a different grid sequence. 2) Intra-grid variations, such as local deviations and sharp turns, are lost, making it difficult to capture fine-

*These authors contributed equally.
†Shuo Shang is the corresponding author.

grained movement details. In essence, although grid representations are robust to irregularity and highlight the overall spatial structure, they may obscure subtle characteristics and fail to capture detailed motion dynamics.

Given these limitations, it is instructive to consider the intrinsic information captured at the point-level. As shown in Figure 1(b), although three trajectories fall within the same grid cell (highlighted in red), point-level information enables a clear distinction among them. Specifically, point-wise features encode three key aspects: (1) *variability*, capturing dynamic changes in inter-point distances or turning angles; (2) *locality*, preserving fine-grained geometric and spatial patterns; and (3) *continuity*, maintaining the smooth progression of movement. These motivate us to combine the robustness of grid-based representations with point-level information for more discriminative trajectory representation.

In this light, we propose **RePo**, an efficient multimodal trajectory similarity learning method that jointly learns trajectory embeddings from both region-wise and point-wise perspectives. For region-wise learning, we consider two modalities to introduce inter-grid information, i.e., structural context, and intra-grid information, i.e., visual context. Structural context embedding captures spatial dependencies and transition patterns between regions by constructing a grid-level transition graph and learning global inter-region correlations. Visual context embedding captures intra-grid semantics by extracting environmental features from the satellite image of each grid cell, enriching the grid representation with detailed local visual cues. For point-wise learning, we employ locality-aware, correlation-aware, and continuity-preserving encoding modules to comprehensively capture local details, dynamic correlations, and continuity at the point level. Region-wise and point-wise features are fused with cross-attention, and the model is trained by supervised contrastive loss.

We conduct extensive experiments on three real-world trajectory datasets and compare with seven SOTA baselines under three widely used trajectory similarity measures, i.e., DTW, DFD, and EDwP (2015). Experiment results show that RePo consistently outperforms all baselines across different datasets and distance measures, with an average accuracy improvement of 22.2%. Ablation studies confirm the effectiveness of RePo's designs. Generalization and visualized results show that RePo can effectively capture true trajectory similarity and generalize well to large-scale datasets.

Our contributions are summarized as follows:

- We observe that existing SOTA methods, which rely solely on grid-based representations, fail to capture fine-grained variations in trajectory details for similarity.
- We propose RePo, a novel framework that jointly learns region-wise and point-wise representations to construct unified trajectory embeddings with a strong capacity to capture fine-grained and discriminative spatial patterns for similarity measurement.
- We evaluate RePo by comparing it against seven state-of-the-art methods across multiple popular similarity measures, conducting comprehensive ablation studies, and evaluating its generalization performance.

2 Related Work

Heuristic-based Similarity Measures. Traditional trajectory similarity measures can be classified into two categories. *Point-based methods*, such as Euclidean Distance, Dynamic Time Warping (DTW) (1998), LCSS (2002), and EDR (2005), compare individual points along the trajectories to compute similarity. *Shape-based methods*, like Hausdorff Distance (2010) and Fréchet Distance (1995), focus on the overall geometric shape, measuring maximum deviation between curves. While these methods provide interpretable similarity scores, they generally require explicit pairwise matching, resulting in $\mathcal{O}(l^2)$ computational complexity, where l is the trajectory length, thus limiting scalability to large datasets.

Learning-based Similarity Measures. Recent learning approaches address the limitations of heuristic methods by learning trajectory-level embeddings for efficient similarity computation. RNN-based models, such as t2vec (2018), NeuTraj (2019), and CL-Tsim (2022), employ recurrent networks to capture sequential and spatial-temporal dependencies. Graph-based models (e.g., TrajGAT (2022), GTS (2021), KGTS (2024b)) leverage graph structures and attention mechanisms to model spatial relationships among trajectory points. Transformer-based models, such as SIMformer (2024) and TrajCL (2023), utilize self-attention for long-range dependency modeling and fine-grained feature extraction. Most existing methods fail to fully exploit the rich information present in trajectories. In contrast, our approach integrates region-wise and point-wise in a unified framework for comprehensive trajectory similarity learning.

3 Preliminaries

3.1 Definition

Definition 1 (GPS Trajectory)**.** *A GPS trajectory T_{gps} is defined as a sequence of points collected at a fixed sampling time interval, where each point $p_i = (lon_i, lat_i)$ of T_{gps} represents the i-th sampled location, consisting of a longitude lon_i and a latitude lat_i.*

Definition 2 (Grid Trajectory)**.** *A grid trajectory T_{grid} is a sequence of discrete grid cells over the city map. Each element $g_i \in T_{grid} = \langle g_1, g_2, ..., g_{n_1} \rangle$ represents a grid cell ID, obtained by mapping GPS points of T_{gps} to the grid space.*

3.2 Problem Statement

Trajectory Similarity Learning. Given a similarity measure $g(\cdot, \cdot)$ (e.g., DTW, DFD), trajectory similarity learning aims to learn a function $f(\cdot, \cdot)$ that approximates $g(\cdot, \cdot)$ for any pair of trajectories (T_i, T_j) in a d-dimensional embedding space, i.e., $f(T_i, T_j) \approx g(T_i, T_j)$. The function $f(\cdot, \cdot)$ is trained to preserve the relative similarity ranking of trajectories. That is, for any three different trajectories T_i, T_j, T_k, if $g(T_i, T_j) < g(T_i, T_k)$, then it should follow that $f(T_i, T_j) < f(T_i, T_k)$.

4 The RePo Method

Figure 2 shows the overall architecture of RePo. The model consists of three key components: 1) *Region-wise Encoder*:

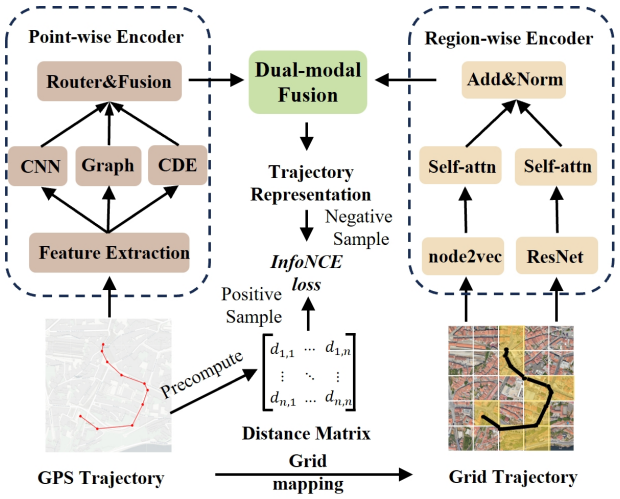


Figure 2: An overview of our RePo Method

it learns regional embeddings from the grid-based trajectory by incorporating both spatial structural features and visual contextual information; 2) *Point-wise Encoder*: it captures point embeddings from the GPS trajectory by modeling three types of encodings: locality, correlation, and continuity; 3) *Dual-modal Fusion*: it integrates point and grid embedding to generate the trajectory embedding. We train our model with a supervised contrastive loss, using positive and hard negative samples based on trajectory distances to learn similarity rankings. We use `[CLS]` token to learn trajectory embedding. For simplicity, multi-head attention and the `[CLS]` token are omitted in the formulas.

4.1 Region-wise Encoder

The region-wise encoder captures global trajectory semantics at the grid level through two complementary representations: structural context embedding, which models inter-region spatial correlations, and visual context embedding, which provides visual features for each region.

Structural Context Embedding. Trajectory movement is inherently constrained by the environment. For example, a vehicle can only move through grid cells that overlap with road segments. To capture the spatial dependencies and transition patterns between different regions, we construct a transition graph $\mathcal{G} \in \mathbb{R}^{M \times M}$ from the training trajectory dataset, where M denotes the total number of grid cells. An edge $\mathcal{G}[i][j] = 1$ is established if there exists a trajectory that moves from grid cell g_i to g_j. We do not incorporate transition frequencies into the graph, as they are often imbalanced and less indicative of the underlying road structure context. Then we use Node2vec (Grover and Leskovec 2016) to learn global spatial correlation grid embeddings as follows:

$$\mathbf{E}^s = \text{Node2Vec}(\mathcal{G}), \tag{1}$$

where $\mathbf{E}^s \in \mathbb{R}^{M \times d}$. Then we can transform a grid trajectory sequence $T_{grid} = \langle g_1, g_2, ..., g_{n_1} \rangle$ into a structural context embedding sequence $\mathbf{E}^s = \langle \mathbf{e}^s_1, \mathbf{e}^s_2, ..., \mathbf{e}^s_{n_1} \rangle$ by looking up the embedding table $\mathcal{E}^s$. Finally, we add position encoding (Vaswani et al. 2017) and use a self-attention to learn sequence-level correlations by following:

$$\mathbf{H}^s = \text{SelfAttn}(\mathbf{E}^s) \in \mathbb{R}^{n_1 \times d}. \tag{2}$$

Visual Context Embedding. In addition to spatial correlations, the visual semantics of grid cells provide rich contextual information. For example, trajectories tend to avoid areas dominated by buildings and are more likely to traverse regions with roads. Easily accessible satellite imagery corresponding to grid cells (e.g., from Google Maps) can reveal such features. To capture these visual semantics, we use ResNet (He et al. 2016) to process the satellite images corresponding to each grid cell and extract visual representations. Given the city map $\mathcal{I} \in \mathbb{R}^{H \times W}$, where $H \times W = M$ and each element in $\mathcal{I}$ corresponds to the satellite image of a grid cell, we extract visual context embeddings as follows:

$$\mathbf{E}^v = \text{ResNet}(\mathcal{I}), \tag{3}$$

where $\mathbf{E}^v \in \mathbb{R}^{M \times d}$. Similar to structural context embedding, we apply position encoding and apply self-attention to learn sequence-level correlations:

$$\mathbf{H}^v = \text{SelfAttn}(\mathbf{E}^v) \in \mathbb{R}^{n_1 \times d}, \tag{4}$$

Therefore, the final trajectory embedding on region context information can be obtained as follows:

$$\mathbf{H}^r = \mathbf{H}^s + \mathbf{H}^v. \tag{5}$$

4.2 Point-wise Encoder

The point-wise encoder learns trajectory representations through three expert modules: locality-aware encoding for local details, correlation-aware encoding for variability, and continuity-preserving encoding for smooth transitions.

Feature Extraction. Since GPS data contains only longitude and latitude, directly inputting it provides limited spatial context. To enhance trajectory representation, we first extract spatial features for each point by converting geographic coordinates to Mercator projection coordinates (x_i, y_i) (Chang et al. 2023), thus mitigating geographic scale effects. Then we compute the distance and bearing angle between p_i and its adjacent points p_{i-1} and p_{i+1}. The resulting feature for each trajectory point is a vector defined as:

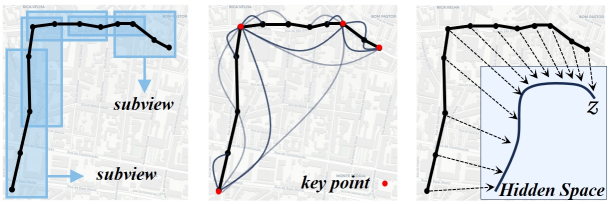
$$\mathbf{s}_i = (x_i, y_i, d_{i-1,i}, \theta_{i-1,i}, d_{i,i+1}, \theta_{i,i+1}), \tag{6}$$

where (x_i, y_i) denote the normalized Mercator coordinates, $d_{i-1,i}$ and $d_{i,i+1}$ represent the distances between p_i and its previous and next points, respectively, and $\theta_{i-1,i}$, $\theta_{i,i+1}$ are the corresponding bearing angles. Then we use a Linear layer to learn point embedding as follows:

$$\mathbf{e}^p_i = \text{Linear}(\mathbf{s}_i) \in \mathbb{R}^d, \tag{7}$$

therefore, a n_2-length GPS trajectory T_{gps} can be transformed into a feature embedding $\mathbf{E}^p \in \mathbb{R}^{n_2 \times d}$.

Locality-aware Encoding. Locality refers to the strong short-range dependency between adjacent points in a trajectory, reflecting the fine-grained geometric and dynamic patterns over short segments. In a trajectory, each point is



(a) Locality (b) Correlation (c) Continuity

Figure 3: Three types of encodings in Point-wise encoder.

highly correlated with its immediate neighbors, capturing local continuity and motion trends. As shown in Figure 3(a), we extract overlapping local sub-segments along the trajectory to focus on these local details, enabling the model to effectively capture and encode fine-grained movement patterns. To effectively capture such local dependencies, we employ a multi-layer 1D Convolutional Neural Network (CNN) (Krizhevsky, Sutskever, and Hinton 2012), which is well-suited for extracting fine-grained locality information by modeling short-range spatial patterns along the trajectory. This ensures that local dynamics are adequately preserved. The locality-aware encoding is learned as follows:

$$\mathbf{H}_{(l)}^{loc} = \text{LeakyReLU}(\text{GN}(\text{Conv}(\mathbf{H}_{(l-1)}^{loc}))) \in \mathbb{R}^{n_2 \times d}, \quad (8)$$

where l is the layer number and $\mathbf{H}_{(0)}^{loc} = \mathbf{E}^p$ is the initial embedding of the trajectory. The final output $\mathbf{E}^{loc} = \mathbf{H}_{(l)}^{loc}$ encodes the local movement patterns within a trajectory.

Correlation-aware Encoding. Global correlation refers to the semantic relationships and dependencies between non-adjacent points in a trajectory, which are essential for capturing overall trajectory variability and complex movement patterns. We represent each trajectory as an adaptive graph, where edges are learned based on the semantic similarity between points. This enables the model to capture both strong and weak dependencies across the entire sequence. As illustrated in Figure 3(b), the connections between key points visualize how global correlations of varying strength are modeled within the trajectory.

Given the trajectory feature matrix $\mathbf{E}^p \in \mathbb{R}^{n_2 \times d}$, node embeddings are obtained via a Linear projection:

$$\mathbf{E} = \text{Linear}(\mathbf{E}^p) \in \mathbb{R}^{n_2 \times d}, \quad (9)$$

Next, an adjacency matrix $\mathcal{A} \in \mathbb{R}^{n_2 \times n_2}$ for a trajectory is constructed based on node similarity:

$$\mathcal{A} = \text{Softmax}\left(\text{ReLU}(\mathbf{E}\mathbf{E}^\top)\right). \quad (10)$$

Structural information is propagated and normalized as:

$$\mathbf{E}^{cor} = \text{LayerNorm}(\mathcal{A}\mathbf{E}W), \quad (11)$$

where $W \in \mathbb{R}^{d \times d}$ is a learnable matrix.

Continuity-preserving Encoding. Continuity at the representation space is essential for faithfully preserving the smooth transitions in trajectories, ensuring that the learned embeddings change gradually along the path. To achieve this, we use a neural controlled differential equation (CDE) encoder (Kidger et al. 2020), which interpolates discrete feature sequences into a continuous path in the representation space. As illustrated in Figure 3(c), this approach effectively enables the model to capture consistent and smooth transitions in the hidden space, thereby providing a natural and coherent encoding of trajectory continuity.

Given $\mathbf{E}^p \in \mathbb{R}^{n_2 \times d}$, we interpolate the feature sequence using Hermite cubic splines (Press et al. 2007) to obtain a continuous path. The initial hidden state $\mathbf{z}_0$ is computed from the first point in the trajectory, where e denotes the feature vector of the first point. Then, the initial hidden state is computed as:

$$\mathbf{z}_0 = \text{MLP}(e) = \phi_2(\phi_1(e)), \quad (12)$$

where ϕ_1 and ϕ_2 are linear layers with ReLU activation. The evolution of the hidden state is governed by the neural CDE:

$$\mathbf{z}(t_\ell) = \mathbf{z}_0 + \int_{t_0}^{t_\ell} f_\theta(\mathbf{z}(s))\, d\mathbf{X}(s), \quad (13)$$

where $f_\theta(\mathbf{z}(t))$ is a neural network function representing the rate of change of the hidden state. The final continuity-preserving embedding can be obtained by:

$$\mathbf{E}^{con} = \text{Linear}(\mathbf{z}) \in \mathbb{R}^{n_2 \times d}. \quad (14)$$

Mixture-of-Experts Dynamic Fusion. Since the three encoding modules capture different aspects of trajectory information, we design a dynamic fusion mechanism to adaptively integrate locality, correlation, and continuity cues. For each position in the trajectory, the importance of each expert is dynamically determined based on the local features at that position, allowing the model to adjust the fusion strategy according to the content of each part of the trajectory.

We adopt a mixture-of-experts (MoE) paradigm (Jacobs et al. 1991; Gan et al. 2026), where the outputs of the three trajectory encoding modules serve as distinct experts. A shared MLP network (router) predicts the importance weights of each expert, enabling adaptive, position-specific fusion. The final fused feature at each position is computed as a weighted sum:

$$\mathbf{w} = \text{Softmax}\big(\text{MLP}(\mathbf{E}^{loc}),\ \text{MLP}(\mathbf{E}^{cor}),\ \text{MLP}(\mathbf{E}^{con})\big), \quad (15)$$

where $\mathbf{w} \in \mathbb{R}^{n_2 \times 3}$ denotes the importance scores for all sequence positions. Let $w_j^{(i)}$ denote the i-th element of the weight vector at position $j (i = 1, 2, 3)$, corresponding to the locality, correlation, and continuity experts, respectively. The fused feature at each position is computed as:

$$\mathbf{H}_j^p = w_j^{(1)}\mathbf{E}_j^{loc} + w_j^{(2)}\mathbf{E}_j^{cor} + w_j^{(3)}\mathbf{E}_j^{con}. \quad (16)$$

This position-wise dynamic fusion enables the model to flexibly leverage the complementary strengths of different experts throughout the trajectory.

4.3 Dual-modal Fusion

The dual-modal fusion module integrates point and grid embeddings to generate the final trajectory representation. We use cross-attention with point embeddings as queries and grid embeddings as keys and values, because each trajectory point benefits from actively attending to global spatial

context. This setup enables each point to selectively aggregate relevant information from the entire grid, allowing the model to enrich local features with comprehensive global cues and produce a more expressive representation.

To seamlessly integrate region-wise and point-wise trajectory information, we design a cross-modal attention fusion module. Specifically, the grid representation $\mathbf{H}^r \in \mathbb{R}^{n_1 \times d}$ serves as the query $\mathbf{Q}$, while the point representation $\mathbf{H}^p \in \mathbb{R}^{n_2 \times d}$ serves as the key $\mathbf{K}$ and value $\mathbf{V}$. The cross-attention operation is formulated as:

$$\mathbf{H}^o = \mathrm{CrossAttn}(\mathrm{q} = \mathbf{H}^r, \mathrm{k} = \mathbf{H}^p, \mathrm{v} = \mathbf{H}^p), \quad (17)$$

where appropriate padding masks are applied for variable-length sequences.

The cross-attended features $\mathbf{H}^o$ are combined with the query via residual connection and layer normalization, and then passed through a feed-forward network (FFN) to yield the final output of the fusion layer:

$$\mathbf{E} = \mathrm{FFN}(\mathrm{LayerNorm}(\mathbf{H}^o + \mathbf{H}^r)) + \mathbf{H}^o, \quad (18)$$

where $\mathbf{E} \in \mathbb{R}^{n_1 \times d}$. Finally, we use the `[CLS]` token of $\mathbf{E}$ as the final trajectory embedding.

4.4 Supervised Contrastive Loss

To train RePo, we employ a supervised InfoNCE-based (van den Oord, Li, and Vinyals 2018) loss that leverages precomputed trajectory distance matrices. Specifically, we compute a pairwise ground-truth distance matrix $\mathcal{D}$ using trajectory distance measures. For each anchor trajectory T_i, the positive sample j^+ is selected as its nearest neighbor in $\mathcal{D}$, while hard negatives are selected from the batch based on cosine similarity in the embedding space (excluding the anchor and positive). For each pair of trajectories T_i and T_j, the similarity $S_{i,j}$ is computed as the cosine similarity between their final embeddings. The loss is defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[\frac{S_{i,j^+}}{\tau} - \log \sum_{j \in \mathcal{N}_i^-} \exp\left(\frac{S_{i,j}}{\tau}\right)\right], \quad (19)$$

where τ is a temperature hyperparameter and $\mathcal{N}_i^-$ denotes the set of k hardest negatives for anchor i.

We select positives using ground-truth trajectory distances to ensure true semantic similarity, while negatives are mined in the representation space to encourage the model to distinguish hard cases under the current embedding. This design leverages both accurate distance supervision and adaptive discrimination, promoting a more robust and semantically meaningful embedding space.

4.5 Complexity Analysis

The overall computational complexity of RePo is dominated by the attention mechanisms within each encoder block. For a trajectory of length L and hidden dimension d, both the self-attention and cross-attention modules require $\mathcal{O}(L^2 d)$ operations due to pairwise sequence interactions. In addition, the convolutional, graph-based, and neural CDE modules, as well as the feed-forward sublayers, each require $\mathcal{O}(Ld^2)$ operations. Therefore, the total per-layer complexity is $\mathcal{O}(L^2 d + Ld^2)$, where the quadratic attention term $\mathcal{O}(L^2 d)$ dominates for longer sequences.

5 Experimental Evaluation

5.1 Experiment Settings

Datasets. We conduct experiments on three widely used real-world trajectory datasets: **Porto** (2016), **Geolife** (2010) and **Chengdu**. The points of trajectories in Porto, Chengdu, and Geolife are sampled every 15 seconds, 30 seconds, and 5 seconds. For all datasets, we retain trajectories with lengths between 10 and 300 points. Following previous studies (2019; 2022), we sampled 10,000 trajectories from each dataset and randomly split them into 20% training, 10% validation, and 70% testing sets.

Baselines. We compare RePo with 7 representative trajectory similarity learning methods as follows:

- **t2vec** (2018): it learns trajectory embeddings using RNN and a spatial proximity-aware loss.
- **NeuTraj** (2019): it is based on LSTM with spatial memory for supervised similarity learning.
- **CL-TSim** (2022): it employs contrastive learning and a Siamese network for trajectory similarity search.
- **TrajGAT** (2022): it models each trajectory as a hierarchical graph and applies graph attention networks.
- **TrajCL** (2023): it adopts self-supervised contrastive learning with dual-feature self-attention.
- **KGTS** (2024b): it incorporates spatial and semantic knowledge into trajectory representation.
- **SIMformer** (2024): it leverages a transformer encoder for efficient similarity approximation.

Implementation Details. All experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU. The embedding dimension is set to 512. The spatial grid for mapping GPS points is constructed at zoom level 18, with each grid cell covering roughly 153 m $\times$ 153 m at the equator. The self-attention layer of the Region-wise encoder is set to 1. The CNN layer of the Point-wise encoder is set to 3. We use a temperature hyperparameter $\tau = 0.2$ for the contrastive loss. The Adam optimizer is used with a learning rate of 2×10^{-5}.

Evaluation Metrics. Following prior studies (Yao et al. 2022; Chang et al. 2023), we use top-k Hit Ratio (HR@k) and top-k Partial Hit Ratio (Rm@k) for trajectory similarity retrieval and ranking evaluation. Besides, we further introduce two widely used ranking metrics, i.e., Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG), to provide a comprehensive evaluation. Higher values indicate better performance.

5.2 Main Results

We report results on the Porto, Geolife, and Chengdu datasets under three trajectory similarity measures (DFD, DTW, EDwP) and four evaluation metrics (HR@1, R5@20, MRR, NDCG@50) in Table 1, more results are shown in the Appendix. RePo consistently achieves the best performance across all metrics and datasets, demonstrating strong robustness to different similarity measures and data distributions. Notably, RePo outperforms the best baselines by up to 72.7% on HR@1 (DTW, Porto), and delivers stable gains in both retrieval and ranking metrics.

Dataset	Method	DFD				DTW				EDwP			
		HR@1	R5@20	MRR	NDCG	HR@1	R5@20	MRR	NDCG	HR@1	R5@20	MRR	NDCG
Porto	t2vec	0.213	0.556	0.337	0.433	0.239	0.624	0.373	0.490	0.235	0.612	0.366	0.494
	CL-TSim	0.101	0.317	0.184	0.213	0.116	0.362	0.204	0.266	0.114	0.357	0.203	0.273
	NeuTraj	0.372	0.840	0.536	0.755	0.308	0.807	0.448	0.672	0.346	0.787	0.501	0.647
	TrajCL	0.326	0.803	0.483	0.687	0.217	0.603	0.343	0.556	0.197	0.586	0.328	0.493
	TrajGAT	0.346	0.748	0.497	0.475	0.296	0.683	0.437	0.450	0.161	0.413	0.261	0.298
	KGTS	0.215	0.392	0.321	0.333	0.253	0.444	0.369	0.411	0.234	0.420	0.345	0.388
	SIMformer	0.422	0.828	0.575	0.714	0.300	0.755	0.446	0.651	0.321	0.579	0.466	0.475
	RePo	**0.483**	**0.922**	**0.640**	**0.771**	**0.532**	**0.955**	**0.682**	**0.821**	**0.512**	**0.947**	**0.667**	**0.833**
	Improv.	**14.5%**	**9.76%**	**11.3%**	**2.11%**	**72.7%**	**18.3%**	**52.2%**	**22.1%**	**47.9%**	**20.3%**	**33.1%**	**28.7%**
Geolife	t2vec	0.180	0.401	0.260	0.334	0.188	0.430	0.273	0.368	0.171	0.429	0.256	0.368
	CL-TSim	0.166	0.506	0.270	0.430	0.181	0.571	0.295	0.497	0.173	0.556	0.285	0.500
	NeuTraj	0.289	0.713	0.403	0.630	0.202	0.532	0.287	0.455	0.250	0.657	0.363	0.581
	TrajCL	0.129	0.479	0.238	0.457	0.107	0.357	0.195	0.361	0.062	0.285	0.134	0.316
	TrajGAT	0.283	0.719	0.426	0.574	0.212	0.587	0.335	0.477	0.214	0.569	0.326	0.501
	KGTS	0.206	0.308	0.283	0.338	0.219	0.324	0.296	0.366	0.210	0.323	0.289	0.369
	SIMformer	0.316	0.730	0.452	0.677	0.194	0.525	0.288	0.474	0.218	0.570	0.342	0.560
	RePo	**0.325**	**0.745**	**0.455**	**0.678**	**0.342**	**0.776**	**0.475**	**0.685**	**0.347**	**0.787**	**0.483**	**0.728**
	Improv.	**2.84%**	**2.05%**	**0.66%**	**0.14%**	**56.5%**	**32.2%**	**41.8%**	**37.8%**	**38.8%**	**19.8%**	**33.1%**	**25.3%**
Chengdu	t2vec	0.306	0.737	0.462	0.533	0.330	0.792	0.493	0.602	0.324	0.768	0.487	0.596
	CL-TSim	0.094	0.400	0.190	0.310	0.114	0.444	0.215	0.367	0.103	0.431	0.204	0.366
	NeuTraj	0.316	0.830	0.478	0.777	0.249	0.722	0.385	0.609	0.263	0.740	0.409	0.657
	TrajCL	0.313	0.859	0.482	0.768	0.110	0.409	0.206	0.354	0.091	0.407	0.184	0.376
	TrajGAT	0.338	0.824	0.505	0.542	0.190	0.618	0.327	0.447	0.100	0.388	0.193	0.321
	KGTS	0.179	0.420	0.294	0.403	0.225	0.491	0.353	0.485	0.186	0.432	0.297	0.463
	SIMformer	0.387	0.862	0.546	0.763	0.235	0.706	0.369	0.608	0.279	0.755	0.429	0.667
	RePo	**0.392**	**0.895**	**0.552**	**0.796**	**0.428**	**0.929**	**0.592**	**0.823**	**0.360**	**0.904**	**0.526**	**0.830**
	Improv.	**1.29%**	**3.82%**	**1.09%**	**2.44%**	**29.6%**	**17.3%**	**20.1%**	**35.1%**	**11.1%**	**17.7%**	**8.00%**	**24.4%**

Table 1: Performance comparison on the three datasets under different trajectory similarity metrics. Best results are bolded, second-best are underlined. "Improv" shows our improvement over the strongest baseline.

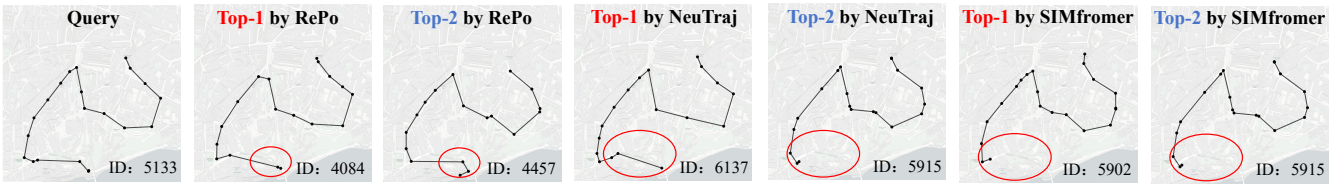


Figure 4: Trajectory visualization results on the Porto dataset. We compare the top-2 retrievals produced by RePo, NeuTraj, and SIMformer. Significant differences in visualization results are highlighted with red circles.

RePo's explicit fusion of region-wise and point-wise representations enables comprehensive trajectory modeling, resulting in higher retrieval accuracy and more robust ranking. While models such as SIMformer and NeuTraj achieve competitive results on certain metrics, their performance fluctuates across datasets and evaluation criteria.

Performance on Geolife is generally lower for all methods, including RePo, likely due to its sparser trajectories and higher urban complexity, which make discrimination more difficult. Notably, improvements under the DFD metric are less pronounced compared to DTW and EDwP. This is because DFD focuses primarily on the global geometric shape and overall alignment of trajectories, while our model emphasizes the capture of multi-scale spatial patterns. In contrast, DTW and EDwP are more sensitive to local similarities and flexible alignments, allowing our discriminative multi-modal representations to demonstrate larger gains.

Trajectory Retrieval Visualization. To qualitatively assess retrieval performance, we visualize the top-2 results from RePo, SIMformer, and NeuTraj for a random query (Figure 4). RePo retrieves trajectories that better match both global and local patterns, highlighting its superior ability to capture comprehensive trajectory similarity.

5.3 Micro Results

Ablation Study. We create several model variants by removing individual modules as follows:

- w/o ResNet: removing the ResNet network.
- w/o node2vec: removing the node2vec network.
- w/o CNN: removing the locality-aware encoding.
- w/o AG: removing the correlation-aware encoding.

Method	DFD		DTW	
	Porto	Chengdu	Porto	Chengdu
w/o ResNet	0.432	0.332	0.491	0.369
w/o node2vec	0.434	0.273	0.468	0.307
w/o CNN	0.374	0.284	0.478	0.344
w/o AG	0.434	0.310	0.467	0.392
w/o CDE	0.425	0.372	0.462	0.403
RePo	**0.483**	**0.392**	**0.532**	**0.428**

Table 2: Ablation study on Porto and Chengdu datasets.

Datasize	Method	HR@1	R5@20	MRR	NDCG@50
100k	NeuTraj	0.143	0.438	0.236	0.429
	SIMformer	0.123	0.427	0.213	0.433
	RePo	**0.300**	**0.788**	**0.448**	**0.725**
200k	NeuTraj	0.111	0.350	0.188	0.351
	SIMformer	0.071	0.200	0.113	0.224
	RePo	**0.192**	**0.555**	**0.302**	**0.508**

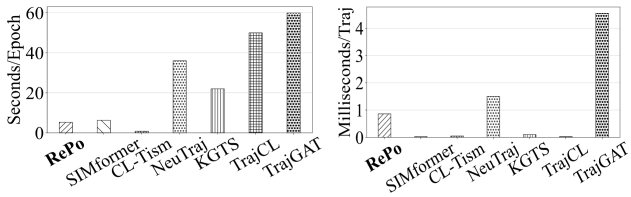
Table 3: Performance comparison on the 100k and 200k Porto dataset. The best results are highlighted in bold.

- w/o CDE: removing the continuity-preserving encoding.

In each variant, the remaining architecture stays unchanged. Table 2 shows that the full RePo model achieves the best HR@1 performance on all metrics and datasets, validating the effectiveness of multi-branch fusion and multi-scale feature extraction. Removing the CNN branch causes the largest drop, highlighting the importance of local pattern extraction. Omitting the ResNet or node2vec modules reduces the model's ability to capture spatial and structural grid information, while removing the graph or CDE branches weakens correlation and continuity modeling. These results confirm the complementary strengths of each module and the benefit of joint representation.

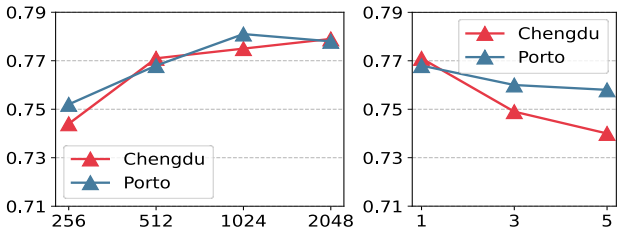
Scalable Experiment. To evaluate scalability, we directly test the model trained on 10,000 Porto trajectories on larger samples of 100,000 and 200,000 trajectories (Table 3). As the test set size increases, the retrieval task becomes more challenging for all methods, leading to an overall decline in performance. This is primarily because the model is trained on a smaller dataset, and the candidate pool becomes significantly larger. Despite this, RePo consistently outperforms NeuTraj and SIMformer across all metrics. This strong performance is attributed to RePo's multi-modal, discriminative representations, which generalize better to large and diverse candidate sets, confirming its robustness and scalability for real-world trajectory retrieval.

Efficiency Study. We compare the training and inference efficiency of different methods on the Porto dataset. The results of Figure 5 show that RePo achieves significantly faster training than TrajCL and TrajGAT, and maintains moderate inference speed, while consistently delivering the best retrieval accuracy. This demonstrates that RePo offers an effective trade-off between efficiency and performance, making it practical for real-world applications.



(a) Training time (b) Inference time

Figure 5: Model training time for one epoch under batch size 128 and model inference time of one trajectory.



(a) Dimensions (b) Sample nums

Figure 6: Hyperparameter sensitivity analysis.

Embedding Dimension. We experiment on how embedding dimension affects RePo's performance. As shown in Figure 6(a), increasing the embedding dimension from 256 to 512 leads to a substantial improvement in HR@50, with performance saturating beyond 512. Further increasing the dimension to 1024 or 2048 yields negligible or inconsistent gains, while incurring higher computational and memory costs. We thus select 512 as the default embedding dimension to balance accuracy and efficiency.

Number of Hard Negative Samples. Figure 6(b) shows the impact of the number of hard negative samples used in the contrastive loss (measured by HR@50). The best performance is obtained with a single hard negative. Using more negatives can introduce noise, especially in the early training stage when representations are not yet stable, which may lead to suboptimal learning. Furthermore, since our goal is to learn correct ranking and the loss primarily enforces pairwise discrimination, a single hard negative is sufficient to drive effective supervision, while additional negatives may dilute the learning signal and even mislead optimization.

6 Conclusion

We proposed RePo, a novel multimodal trajectory similarity learning method that integrates region-wise and point-wise information through a combination of structural, visual, and trajectory features. By adopting a mixture-of-experts paradigm and a hierarchical attention-based fusion mechanism, RePo effectively captures the complementary strengths of diverse feature modalities and models fine-grained trajectory characteristics. Extensive experiments on multiple real-world datasets show that RePo consistently achieves better performance compared to SOTA baselines.

Acknowledgments

This work was supported by the National Key R&D Program of China (2024YFE0111800) and the National Natural Science Foundation of China (U22B2037 and U21B2046).

References

Alt, H.; and Godau, M. 1995. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5: 75–91.

Atev, S.; Miller, G.; and Papanikolopoulos, N. P. 2010. Clustering of Vehicle Trajectories. *IEEE Trans. Intell. Transp. Syst.*, 11(3): 647–657.

Bringmann, K.; and Mulzer, W. 2016. Approximability of the discrete Fréchet distance. *J. Comput. Geom.*, 7(2): 46–76.

Chang, Y.; Qi, J.; Liang, Y.; and Tanin, E. 2023. Contrastive Trajectory Similarity Learning with Dual-Feature Attention. In *ICDE*, 2933–2945.

Chen, L.; Özsu, M. T.; and Oria, V. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *SIGMOD*, 491–502.

Chen, L.; Shang, S.; Jensen, C. S.; Yao, B.; Zhang, Z.; and Shao, L. 2019. Effective and Efficient Reuse of Past Travel Behavior for Route Recommendation. In *KDD*, 488–498. ACM.

Chen, W.; Liang, Y.; Zhu, Y.; Chang, Y.; Luo, K.; Wen, H.; Li, L.; Yu, Y.; Wen, Q.; Chen, C.; Zheng, K.; Gao, Y.; Zhou, X.; and Zheng, Y. 2024a. Deep Learning for Trajectory Data Management and Mining: A Survey and Beyond. *arXiv preprint arXiv:2403.14151*.

Chen, Z.; Zhang, D.; Feng, S.; Chen, K.; Chen, L.; Han, P.; and Shang, S. 2024b. KGTS: Contrastive Trajectory Similarity Learning over Prompt Knowledge Graph Embedding. In *AAAI*, 8311–8319.

Deng, L.; Zhao, Y.; Fu, Z.; Sun, H.; Liu, S.; and Zheng, K. 2022. Efficient Trajectory Similarity Computation with Contrastive Learning. In *CIKM*, 365–374.

Feng, Z.; and Zhu, Y. 2016. A Survey on Trajectory Data Mining: Techniques and Applications. *IEEE Access*, 4: 2056–2067.

Gan, W.; Ning, Z.; Qi, Z.; and Yu, P. S. 2026. Mixture of experts (MoE): A big data perspective. *Inf. Fusion*, 127: 103664.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*, 855–864.

Han, P.; Wang, J.; Yao, D.; Shang, S.; and Zhang, X. 2021. A Graph-based Approach for Trajectory Similarity Computation in Spatial Networks. In *KDD*, 556–564.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778. IEEE Computer Society.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive Mixtures of Local Experts. *Neural Comput.*, 3(1): 79–87.

Kidger, P.; Morrill, J.; Foster, J.; and Lyons, T. J. 2020. Neural Controlled Differential Equations for Irregular Time Series. In *NeurIPS*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Bartlett, P. L.; Pereira, F. C. N.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *NIPS*, 1106–1114.

Li, X.; Zhao, K.; Cong, G.; Jensen, C. S.; and Wei, W. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *ICDE*, 617–628.

Moreira-Matias, L.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; and Damas, L. 2016. Time-evolving OD matrix estimation using high-speed GPS data streams. *Expert systems with Applications*, 44.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 2007. *Numerical recipes: the art of scientific computing, 3rd Edition*. Cambridge University Press.

Ranu, S.; P, D.; Telang, A. D.; Deshpande, P.; and Raghavan, S. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *ICDE*, 999–1010.

Shang, S.; Chen, L.; Wei, Z.; Jensen, C. S.; Zheng, K.; and Kalnis, P. 2018a. Parallel trajectory similarity joins in spatial networks. *VLDB J.*, 27(3): 395–420.

Shang, S.; Chen, L.; Wei, Z.; Jensen, C. S.; Zheng, K.; and Kalnis, P. 2018b. Parallel trajectory similarity joins in spatial networks. *VLDB J.*, 27(3): 395–420.

Shang, S.; Ding, R.; Zheng, K.; Jensen, C. S.; Kalnis, P.; and Zhou, X. 2014. Personalized trajectory matching in spatial networks. *VLDB J.*, 23(3): 449–468.

van den Oord, A.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*, 5998–6008.

Vlachos, M.; Gunopulos, D.; and Kollios, G. 2002. Discovering Similar Multidimensional Trajectories. In *ICDE*, 673–684.

Yang, C.; Jiang, R.; Xu, X.; Xiao, C.; and Sezaki, K. 2024. SIMformer: Single-Layer Vanilla Transformer Can Learn Free-Space Trajectory Similarity. *Proc. VLDB Endow.*, 18(2): 390–398.

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 *ICDE*, 1358–1369.

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 *KDD*, 2275–2285.

Yi, B.; Jagadish, H. V.; and Faloutsos, C. 1998. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *ICDE*, 201–208.

Yuan, J.; Zheng, Y.; and Xie, X. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *KDD*, 186–194.

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).

Zhou, S.; Chen, Y.; Shang, S.; Chen, L.; He, B.; and Shibasaki, R. 2025a. Blurred Encoding for Trajectory Representation Learning. In *KDD*, 4132–4143.

Zhou, S.; Shang, S.; Chen, L.; Han, P.; and Jensen, C. S. 2025b. Grid and Road Expressions Are Complementary for Trajectory Representation Learning. In *KDD*, 2135–2146.

Zhou, S.; Shang, S.; Chen, L.; Jensen, C. S.; and Kalnis, P. 2024. RED: Effective Trajectory Representation Learning with Comprehensive Information. *Proc. VLDB Endow.*, 18(2): 80–92.