MovSemCL: Movement-Semantics Contrastive Learning for Trajectory Similarity

Zhichen Lai[1], Hua Lu[2*], Huan Li[3,4], Jialiang Li[1], Christian S. Jensen[2]

[1]Department of People and Technology, Roskilde University, Roskilde, Denmark
[2]Department of Computer Science, Aalborg University, Aalborg/Copenhagen, Denmark
[3]The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China
[4]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, Hangzhou, China

zhichenl@ruc.dk, luhua@cs.aau.dk, lihuancs@zju.edu.cn, jiali@ruc.dk, csj@cs.aau.dk

Abstract

Trajectory similarity computation is fundamental functionality that is used for, e.g., clustering, prediction, and anomaly detection. However, existing learning-based methods exhibit three key limitations: (1) insufficient modeling of trajectory semantics and hierarchy, lacking both movement dynamics extraction and multi-scale structural representation; (2) high computational costs due to point-wise encoding; and (3) use of physically implausible augmentations that distort trajectory semantics. To address these issues, we propose MovSemCL, a movement-semantics contrastive learning framework for trajectory similarity computation. MovSemCL first transforms raw GPS trajectories into movement-semantics features and then segments them into patches. Next, MovSemCL employs intra- and inter-patch attentions to encode local as well as global trajectory patterns, enabling efficient hierarchical representation and reducing computational costs. Moreover, MovSemCL includes a curvature-guided augmentation strategy that preserves informative segments (e.g., turns and intersections) and masks redundant ones, generating physically plausible augmented views. Experiments on real-world datasets show that MovSemCL is capable of outperforming state-of-the-art methods, achieving mean ranks close to the ideal value of 1 at similarity search tasks and improvements by up to 20.3% at heuristic approximation, while reducing inference latency by up to 43.4%.

Code — https://github.com/ryanlaics/MovSemCL

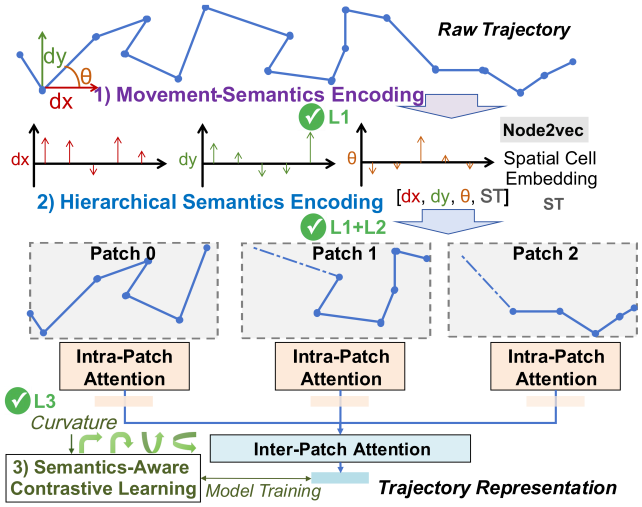


Figure 1: Pipeline of MovSemCL. The framework addresses three limitations: Movement-Semantics Encoding extracts movement dynamics (**L1**), Hierarchical Semantics Encoding captures multi-scale patterns with reduced complexity (**L1, L2**), and Semantics-Aware Contrastive Learning uses curvature-guided augmentation (**L3**).

Introduction

The proliferation of GPS-enabled devices enabled massive collections of vehicle trajectory data, enabling applications such as ride-sharing, logistics, and urban analytics (Zheng 2015). A fundamental operation underlying these applications is *trajectory similarity computation*, which quantifies the similarity between trajectories. This functionality is key to enabling a variety of trajectory applications, like similarity search (Su et al. 2020), route recommendation (Chen et al. 2020), and mobility prediction (Feng et al. 2018).

Using rigid geometric similarity measures (Hausdorff 1914; Fréchet 1906), traditional trajectory similarity computation methods (Cormode and Muthukrishnan 2007; Keogh and Ratanamahatana 2005; Vlachos, Kollios, and Gunopulos 2002) are computationally expensive and ignore underlying semantics. Recent learning-based methods embed trajectories into latent vector spaces using neural architectures such as RNNs (Liu et al. 2016; Li, Liu, and Lu 2024; Deng et al. 2022), CNNs (Yao et al. 2018; Chang et al. 2024), and Transformers (Xu et al. 2020; Chang et al. 2023), enabling efficient similarity computation. However, these methods still face three key limitations:

Limitation 1 (L1): Insufficient modeling of trajectory semantics and hierarchy. Trajectories contain movement-semantics and are hierarchical in nature—points form maneuvers, maneuvers compose travels—requiring both semantic feature extraction and hierarchical modeling of fine-grained local patterns and coarse-grained global patterns. Existing methods (Liu et al. 2016; Li, Liu, and Lu 2024; Deng et al. 2022; Chang et al. 2023; Xu et al. 2020) treat trajectories as flat sequences of raw coordinates, failing to

*Corresponding author.

capture both movement dynamics and multi-scale structure.

Limitation 2 (L2): Computational inefficiency for trajectories with many locations. Real-world trajectories often contain hundreds of points. RNN-based methods (Liu et al. 2016; Li, Liu, and Lu 2024; Deng et al. 2022) lack parallelization, while Transformer-based methods (Xu et al. 2020; Chang et al. 2023) scale quadratically with sequence length, forcing lossy downsampling that compromises movement fidelity.

Limitation 3 (L3): Semantically-unaware augmentations in contrastive learning. Existing contrastive methods (Chang et al. 2023; Li et al. 2022) apply generic augmentations that create physically impossible trajectories—randomly masking points causes spatial jumps, while uniform sampling destroys critical movement patterns like turns and intersections, corrupting signals that are important for learning.

To address these limitations, we introduce `MovSemCL`, a movement-semantics contrastive learning framework for trajectory similarity computation. As shown in Figure 1, `MovSemCL` transforms raw GPS sequences into interpretable movement features (see Section), segments them into semantically coherent patches, and encodes both local and global patterns through dual-level attention (see Section). In addition, `MovSemCL` features a curvature-guided augmentation strategy that generates physically plausible trajectory views by masking redundant segments and preserving behaviorally salient ones (see Section).

Our main contributions are as follows:

- We propose `MovSemCL`, a movement-semantics contrastive learning framework for trajectory similarity computation that captures movement dynamics.
- We design three key components: movement-semantics encoding captures rich movement semantics (**L1**), hierarchical patch-based encoding reduces computational complexity from quadratic to near-linear (**L1, L2**), and curvature-guided augmentation (CGA) generates physically plausible, semantics-aware trajectory augmentations for robust learning (**L3**).
- Extensive experiments on real-world datasets demonstrate that `MovSemCL` achieves up to **72.6%** more accurate similarity search and **43.4%** faster inference compared to state-of-the-art baselines.

Problem Formulation

GPS Trajectory A GPS trajectory $\mathcal{T}$ is a sequence of timestamped locations:

$$\mathcal{T} = \langle (s_0, t_0), (s_1, t_1), \cdots, (s_{L-1}, t_{L-1}) \rangle, \tag{1}$$

where $s_i = (\text{lon}_i, \text{lat}_i)$ are spatial coordinates, t_i is a timestamp, and L is the trajectory length.

Trajectory Similarity Computation Given a collection $\mathcal{D} = \{\mathcal{T}_1, \cdots, \mathcal{T}_N\}$ of GPS trajectories with varying lengths, the objective is to learn a representation function $f : \mathcal{D} \to \mathbb{R}^d$ that maps a trajectory $\mathcal{T} \in \mathcal{D}$ to a fixed-dimensional embedding vector $\mathbf{z} = f(\mathcal{T})$, facilitating efficient similarity computation. The similarity between trajectories $\mathcal{T}_i$ and $\mathcal{T}_j$ is then defined as the Cosine similarity of their embeddings:

$$\text{sim}(\mathcal{T}_i, \mathcal{T}_j) = \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|} \tag{2}$$

Methodology

Overall Architecture

As shown in Figure 1, `MovSemCL` encompasses three stages: (1) *Movement-Semantics Encoding* transforms raw GPS data into movement-semantics features; (2) *Hierarchical Semantics Encoding* segments trajectories into patches for hierarchical modeling via dual-level attention; (3) *Semantics-Aware Contrastive Learning* leverages physically plausible augmentations and a contrastive loss to focus learning on behaviorally informative segments.

Movement-Semantics Encoding

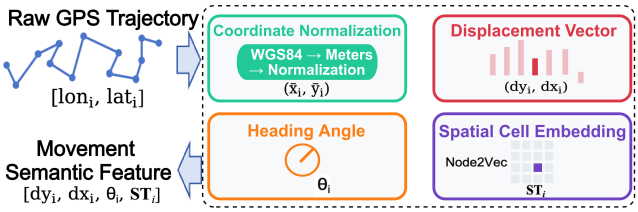


Figure 2: Overview of Movement-Semantics Encoding.

As illustrated in Figure 2, the proposed Movement-Semantics Encoding facilitates modeling of trajectory semantics by first normalizing GPS coordinates, extracting movement dynamics features (displacement vectors and heading angles), constructing trajectory-induced spatial graphs via Node2Vec (Grover and Leskovec 2016) to encode spatial context into spatial cell embeddings, and composing all features into movement-semantics representations (**L1**).

Coordinate Normalization Raw GPS coordinates in the WGS84 coordinate system (longitude, latitude) are not suitable for direct distance computation due to the Earth's curvature. Therefore, we project GPS coordinates to a planar coordinate system using the Mercator projection (Snyder 1987):

$$(mx_i, my_i) = \text{Mercator}(\text{lon}_i, \text{lat}_i) \tag{3}$$

The resulting coordinates are further normalized with respect to the map region of interest:

$$(\bar{x}_i, \bar{y}_i) = \left(\frac{mx_i - x_{\min}^{(m)}}{W_{\text{r}}}, \frac{my_i - y_{\min}^{(m)}}{H_{\text{r}}} \right), \tag{4}$$

where W_{r} and H_{r} are the width and height of the region.

Movement Dynamics Features We then compute the displacement vectors and heading angle for each point:

$$dx_i = \begin{cases} 0 & i = 0 \\ \bar{x}_i - \bar{x}_{i-1} & i > 0 \end{cases}, \quad dy_i = \begin{cases} 0 & i = 0 \\ \bar{y}_i - \bar{y}_{i-1} & i > 0 \end{cases}. \tag{5}$$

Using these displacement vectors, we calculate the heading angle to capture directional changes:

$$\theta_i = \begin{cases} \frac{\arctan 2(dy_i, dx_i)}{\pi} & \text{if } dx_i^2 + dy_i^2 > \epsilon \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $\epsilon = 10^{-6}$ prevents division by zero. These features capture directional flow and instantaneous changes, enabling distinction between smooth movements and abrupt maneuvers.

Trajectory-Induced Spatial Graph Construction To provide spatial context beyond coordinates, we partition the map region into a regular grid of $N_x \times N_y$ cells and assign each trajectory point to a cell according to its normalized coordinates $(\bar{x}_i, \bar{y}_i)$:

$$\text{cell}_i = \left\lfloor \frac{\bar{x}_i}{\Delta_x} \right\rfloor \times N_y + \left\lfloor \frac{\bar{y}_i}{\Delta_y} \right\rfloor, \tag{7}$$

where Δ_x, Δ_y define the grid resolution and N_y is the number of cells along the y-axis.

We construct a trajectory-induced directed graph $G = (V, E)$ where each cell is modeled as a node $v \in V$. Given the trajectory collection $\mathcal{D}$, we define edges $e_{ij} \in E$ between cells i and j based on consecutive transitions observed in trajectories:

$$e_{ij} = \begin{cases} 1 & \text{if } \exists \mathcal{T} \in \mathcal{D}, \exists \text{ timestamp } t: \\ & \quad \text{cell}_t = i \text{ and } \text{cell}_{t+1} = j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The edge weight w_{ij} represents the number of transitions from cell i to cell j:

$$w_{ij} = \sum_{\mathcal{T} \in \mathcal{D}} \sum_{t=0}^{L-2} \mathbf{1}[\text{cell}_t = i \text{ and } \text{cell}_{t+1} = j]. \tag{9}$$

This graph encodes mobility patterns where nodes representing frequently connected cells have large weights, reflecting common movements, while pairs of nodes with no edges represent cells with no immediate movement between them. We apply Node2Vec (Grover and Leskovec 2016) to learn a structural embedding for each cell:

$$\mathbf{ST}_i = \text{Node2Vec}(G, \text{cell}_i) \in \mathbb{R}^{d_{\text{se}}} \tag{10}$$

Feature Composition We concatenate all features at each point to form the final movement-semantics representation:

$$\mathbf{f}_i = [dx_i, dy_i, \theta_i, \mathbf{ST}_i] \in \mathbb{R}^{d_{\text{in}}}, \tag{11}$$

where $d_{\text{in}} = 3 + d_{\text{se}}$. Here, the first three elements encode local movement, while $\mathbf{ST}_i$ captures spatial context.

Hierarchical Semantics Encoding

Conventional flat sequence models struggle to capture both local motion details and global intent (**L1**) and scale poorly to trajectories with many locations (**L2**). Our hierarchical encoding builds on the movement-semantics features to address **L1** and **L2**, enabling efficient modeling of trajectory semantics while reducing computational complexity.

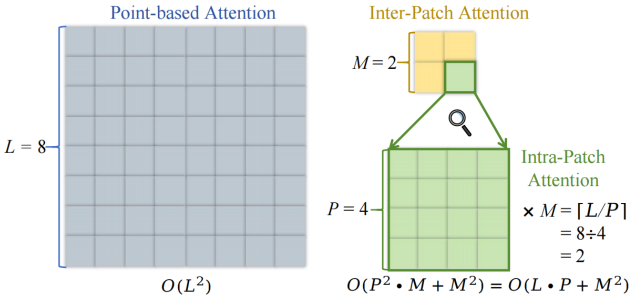


Figure 3: Comparison of attention mechanisms. (a) Traditional Self-Attention with $L = 8$. (b) Hierarchical Semantics Encoding with $L = 8$, $P = 4$, and $M = \lceil L/P \rceil = 2$.

Patch Construction Given an enriched feature sequence $\langle \mathbf{f}_i \rangle$ from the previous stage, we partition it into $M = \lceil L/P \rceil$ patches, where P is the patch length and M is the number of patches in the trajectory. We obtain the following representation, where a patch $\mathbf{P}_j \in \mathbb{R}^{P \times d_{\text{in}}}$ serves as a locally coherent movement unit:

$$\mathcal{P} = \langle \mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_M \rangle \tag{12}$$

As shown in Figure 3, this patch-based approach reduces computational complexity from $O(L^2)$ in conventional flat sequence models to $O(L \cdot P + M^2)$, where $M = \lceil L/P \rceil$ for typical trajectory lengths, and $M \ll L$ holds, effectively addressing the scalability issues caused by trajectories with many locations (**L2**). Notably, the dominant term in the complexity depends on the relationship between L and P: when $L < P^3$, the $O(L \cdot P)$ term dominates, resulting in near-linear scaling with L; otherwise, when $L \geq P^3$, the $O(M^2)$ term becomes dominant, leading to quadratic growth in complexity. Padding and binary masks are used to support variable-length trajectories, following a previous study (Chang et al. 2023).

Intra-Patch Attention Within each patch, we employ a self-attention layer to capture patterns unique to each local segment. The intra-patch attention outputs $\mathbf{H}_j^{\text{intra}} \in \mathbb{R}^{P \times d_h}$ are computed as follows.

$$\mathbf{H}_j^{\text{intra}} = \text{SelfAttn}_{\text{intra}}(\mathbf{P}_j + \mathbf{PE}_{\text{local}}), \tag{13}$$

where $\mathbf{PE}_{\text{local}}$ provides local positional encoding. To summarize each patch as a fixed-length embedding, we apply masked average pooling over valid (non-padded) positions:

$$\mathbf{h}_j = \frac{1}{\sum_{k=1}^{P}(1 - m_{j,k}^{\text{intra}})} \sum_{k=1}^{P} (1 - m_{j,k}^{\text{intra}}) \cdot \mathbf{H}_{j,k}^{\text{intra}}, \tag{14}$$

where $m_{j,k}^{\text{intra}}$ is a binary mask indicating padding.

Inter-Patch Attention The patch embedding $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_M]^T \in \mathbb{R}^{M \times d_h}$ is then processed by an inter-patch self-attention layer, capturing long-range dependencies and global trajectory intent:

$$\mathbf{H}^{\text{inter}} = \text{SelfAttn}_{\text{inter}}(\mathbf{H} + \mathbf{PE}_{\text{global}}), \tag{15}$$

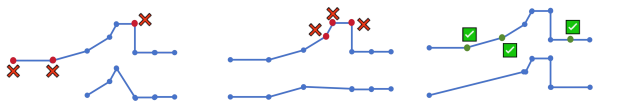
where $\mathbf{PE}_{\text{global}}$ encodes patch-level position information and padded patches are excluded from the computation.

Trajectory Embedding Finally, we aggregate the outputs of the inter-patch attention to obtain a compact, expressive trajectory embedding:

$$\mathbf{z} = \frac{1}{\sum_{j=1}^{M}(1 - m_j^{\text{inter}})} \sum_{j=1}^{M}(1 - m_j^{\text{inter}}) \cdot \mathbf{H}_j^{\text{inter}}, \quad (16)$$

where m_j^{inter} masks out padded patches. The resulting embedding $\mathbf{z} \in \mathbb{R}^{d_h}$ robustly preserves both local and global movement semantics of a trajectory.

Semantics-Aware Contrastive Learning



(a) Random Point Mask (b) Random Block Mask (c) Curvature Guided Mask

Figure 4: Comparison of trajectory masking strategies.

While the movement-semantics and hierarchical encodings enable rich trajectory modeling, it is crucial that the model learning focuses on behaviorally informative segments—such as sharp turns or rare maneuvers—rather than overfitting to redundant, straight-line fragments (**L3**).

To achieve this, we propose a semantics-aware contrastive learning framework. For each trajectory, we generate *two different augmented views* using *Curvature-Guided Augmentation (CGA)* to create a positive pair, while embeddings of other trajectories in a batch serve as negative samples. This design ensures physically plausible and semantically meaningful augmentations, enhancing the learning by considering behaviorally relevant patterns.

Curvature-Guided Augmentation Effective contrastive learning hinges on generating semantically consistent and physically plausible augmented views. `MovSemCL` generates two different augmented views of each trajectory using *Curvature-Guided Augmentation (CGA)*, which form positive pairs for contrastive learning. This augmentation masks trajectory points with probabilities inversely proportional to their local curvature: high-curvature regions (e.g., turns or intersections) are preferentially retained, while straight-line segments are preferentially dropped. Compared to naive random or block masking (Figures 4a–b), CGA produces views that preserve behaviorally informative patterns better, while avoiding spatial discontinuities (Figure 4c). The CGA procedure is detailed in the appendix (Lai et al. 2025).

Contrastive Objective To learn a model, we adopt the MoCo contrastive learning framework (He et al. 2020), which maintains a dynamic queue of negative embeddings. Given a query embedding $\mathbf{z}_q$, its corresponding positive key $\mathbf{z}_k$, a set of negatives $\{\mathbf{z}_n^-\}_{n=1}^N$, and temperature τ, the contrastive loss is formulated as:

$$\mathcal{L} = -\frac{\text{sim}(\mathbf{z}_q, \mathbf{z}_k)}{\tau} + \log\left(e^{\text{sim}(\mathbf{z}_q, \mathbf{z}_k)/\tau} + \sum_{n=1}^{N} e^{\text{sim}(\mathbf{z}_q, \mathbf{z}_n^-)/\tau}\right), \quad (17)$$

where $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$. Only the query encoder is updated via backpropagation; the key encoder is updated using an exponential moving average for stability.

Experiments

To assess the capabilities of `MovSemCL`, we design experiments to answer the following research questions:

- **RQ1 (Effectiveness)**: How does `MovSemCL` perform at trajectory similarity computation?
- **RQ2 (Robustness)**: How robust is `MovSemCL` under real-world conditions with noisy and degraded trajectories?
- **RQ3 (Versatility)**: Can `MovSemCL` approximate time-consuming heuristic similarity measures with fine-tuning?
- **RQ4 (Efficiency)**: What are the computational advantages of `MovSemCL` for large-scale trajectory processing?
- **RQ5 (Component Analysis)**: How do the components of `MovSemCL` contribute to its performance?
- **RQ6 (Hyperparameter Sensitivity)**: How do key hyperparameters affect `MovSemCL` and what are the best settings?

Experimental Setup

Datasets We use two complementary real-world datasets. **Porto**: dense taxi trajectories from Porto, Portugal (July 2013–June 2014), containing 48 points and being 6.37 km long on average, representing short-distance urban mobility. **Germany**: sparse long-distance trajectories across Germany (2006–2013), containing 72 points and being 252.49 km long on average, capturing diverse inter-city travel patterns. Following established evaluation protocols (Chang et al. 2023; Li et al. 2018; Liu et al. 2022), we eliminate trajectories with less than 20 or more than 200 points, and we exclude those outside valid geographic regions. For fair comparison with prior work (Chang et al. 2023), we use a subset of 200 thousand trajectories from the Porto dataset and the full Germany dataset. Each dataset is then split into training/validation/test sets at 70%/10%/20%. From the test set, we reserve 10 thousand samples for the heuristic similarity approximation, partitioned using the same ratio.

Implementation Details Based on preliminary experiments (see Figure 6), we set the patch size $P = 4$, the embedding dimension $d = 256$, the hidden dimension $d_h = 256$, and we train for 20 epochs with early stopping. We use the Adam optimizer with a learning rate $1e^{-4}$, batch size 128, and temperature $\tau = 0.05$ for contrastive learning. For the spatial discretization, we employ grids with cell sizes adapted to the spatial scale of each dataset: 100 meters for Porto and 1000 meters for Germany. All experiments are conducted on NVIDIA RTX A6000 GPUs. Further details on computational complexity, baselines, and datasets are in the appendix (Lai et al. 2025).

Trajectory Retrieval Performance (RQ1)

We first investigate `MovSemCL`'s effectiveness on the core task of trajectory similarity computation.

Experimental Protocol Following baseline methods (Chang et al. 2023), we evaluate trajectory similarity using a query set $\mathcal{Q}$ and database $\mathcal{D}$ created from 100K test trajectories. We randomly sample 1K trajectories and then split each trajectory $\mathcal{T}^q$ into odd points $\mathcal{T}^q_a = [p_1, p_3, p_5, \cdots]$ and even points $\mathcal{T}^q_b = [p_2, p_4, p_6, \cdots]$. $\mathcal{T}^q_a$ serves as query and $\mathcal{T}^q_b$ as ground-truth in database $\mathcal{D}$, which is augmented with random trajectories to form databases of varying sizes. This splitting creates reasonable similar pairs representing the same movement sequence with different sampling offsets. We report the mean rank of the ground-truth $\mathcal{T}^q_b$ when retrieving similar trajectories for each query $\mathcal{T}^q_a$, with 1 being the ideal rank and smaller ranks being better.

Results and Analysis Table 1 presents the mean rank results across different database sizes. MovSemCL consistently achieves the best performance, with mean ranks very close to 1 across all database sizes.

The performance gap reveals fundamental limitations in existing approaches. Traditional geometric methods (EDR, CSTRM, Hausdorff) measure spatial alignment without movement dynamics, causing severe degradation as geometric similarity becomes ambiguous in large databases. RNN-based methods (t2vec, TrjSR, E2DTC) struggle with long-range dependencies and lack parallelization, while the Transformer-based TrajCL treats trajectories as flat sequences, missing hierarchical movement structure. Additionally, TrajCL's use of random augmentation creates physically implausible trajectories, weakening its contrastive learning. MovSemCL's near-optimal performance across both dense urban (Porto) and sparse long-distance (Germany) datasets indicates that movement semantics—rather than geometric or temporal patterns alone—provide robust discriminative features that scale effectively. The stability across database sizes reflects a key insight: trajectory similarity is fundamentally about behavioral similarity, which existing methods do not capture comprehensively.

Robustness Evaluation (RQ2)

We proceed to examine MovSemCL's robustness to data degradation that occurs commonly in real-world settings.

Down-sampling Robustness GPS trajectories often suffer from missing data points due to signal loss, battery constraints, or privacy-preserving sampling. Following prior studies (Chang et al. 2023; Li, Liu, and Lu 2024), we down-sample trajectories in $\mathcal{Q}$ and $\mathcal{D}$ by randomly masking points in each trajectory with a probability $\rho_s \in [0.1, 0.5]$, while keeping $|\mathcal{D}| = 100,000$. Table 2 shows that MovSemCL achieves the best performance across all down-sampling rates. Among deep learning methods, TrajCL performs well at low rates but deteriorates significantly as degradation increases (36.352 at 0.5 on Porto). CLEAR, designed with augmentation strategies for robustness, maintains more stable performance on intra-city dataset Porto but degrades on inter-city trajectories like Germany. However, MovSemCL exhibits the best robustness across most settings, clearly demonstrating that movement-semantics modeling provides inherent robustness to data sparsity.

Dataset	Method	20K	40K	60K	80K	100K
Porto	EDR	8.318	14.398	17.983	22.902	28.753
	CSTRM	4.476	7.954	10.630	13.576	16.699
	EDwP	3.280	4.579	5.276	6.191	7.346
	Hausdorff	3.068	4.014	4.649	5.451	6.376
	Fréchet	3.560	4.959	5.968	7.192	8.631
	t2vec	1.523	2.051	2.257	2.612	3.068
	TrjSR	1.876	2.783	3.208	3.826	4.635
	E2DTC	1.560	2.111	2.349	2.731	3.213
	CLEAR	1.435	1.593	1.766	1.923	2.077
	TrajCL	<u>1.005</u>	<u>1.006</u>	<u>1.006</u>	<u>1.007</u>	<u>1.010</u>
	MovSemCL	**1.002**	**1.004**	**1.005**	**1.005**	**1.005**
Germany	EDR	279.385	558.288	834.208	1108.975	1370.004
	CSTRM	OOM	OOM	OOM	OOM	OOM
	EDwP	2.168	2.277	2.371	2.454	2.515
	Hausdorff	2.803	3.509	4.206	4.906	5.551
	Fréchet	2.581	3.108	3.633	4.113	4.589
	t2vec	1.571	1.982	2.387	2.718	3.053
	TrjSR	6.517	11.741	16.969	22.182	24.083
	E2DTC	3.136	5.156	7.248	9.207	10.956
	CLEAR	1.104	1.138	1.177	1.202	1.222
	TrajCL	<u>1.012</u>	<u>1.022</u>	<u>1.034</u>	<u>1.040</u>	<u>1.045</u>
	MovSemCL	**1.002**	**1.003**	**1.003**	**1.005**	**1.008**

Table 1: Mean rank vs. database size (RQ1). Best results are in **bold**, second-best are <u>underlined</u>. MovSemCL consistently achieves the best mean rank (ideal=1).

Distortion Robustness Real GPS data contains coordinate inaccuracies due to sensor noise and atmospheric interference. We follow prior studies (Chang et al. 2023; Li, Liu, and Lu 2024) and apply random coordinate shifts to a proportion $\rho_d \in [0.1, 0.5]$ of trajectory points. Table 3 shows that MovSemCL again exhibits the best robustness, maintaining mean ranks very close to 1 across all distortion rates.

Heuristic Similarity Approximation (RQ3)

Beyond similarity search, we evaluate whether or not MovSemCL's learned representations can effectively approximate traditional distance measures, thereby assessing the versatility of its movement-semantics embeddings.

Experimental Protocol Following the prior work (Chang et al. 2023), we fine-tune the pre-trained MovSemCL encoder with a two-layer multilayer perceptron head to predict EDR, EDwP, Hausdorff, and Fréchet distances using the MSE loss. This setup tests whether the movement-semantics representations capture sufficient information to approximate these time-consuming heuristic measures. We compare against both fine-tuning models and supervised methods trained specifically for distance prediction, including TrajSimVec (Zhang et al. 2020), TrajGAT (Yao et al. 2022), and T3S (Yang et al. 2021). We report HR@k (Hit Ratio at k), the proportion of ground-truth top-k similar trajectories correctly identified in predicted top-k results, and R5@20 (Recall of top-5 in top-20), measuring recall of returning ground-truth top-5 trajectories in the top-20 results.

Results Table 4 shows that MovSemCL achieves an average rank of 1 across all measures. Notably, MovSemCL demonstrates improvements over TrajCL: 20.3% improvement on EDR, 1.7% on Hausdorff, and 1.8% on Fréchet for

Dataset	Method	0.1	0.2	0.3	0.4	0.5
Porto	EDR	57.173	203.993	806.033	2286.821	4872.231
	CSTRM	24.794	47.137	123.124	257.540	687.262
	EDwP	8.442	10.968	18.727	28.394	68.061
	Hausdorff	10.026	23.293	56.561	89.827	275.206
	Fréchet	10.668	18.516	29.740	93.851	181.271
	t2vec	4.786	8.461	19.689	35.219	115.364
	TrjSR	7.941	15.746	151.948	549.108	1341.883
	E2DTC	5.100	9.385	21.845	39.402	124.320
	CLEAR	1.518	1.914	2.792	3.650	**6.090**
	TrajCL	1.026	1.191	**1.513**	3.847	36.352
	MovSemCL	**1.018**	**1.098**	1.682	**1.961**	9.951
Germany	EDR	1368.829	1379.489	1375.261	1380.517	1389.433
	EDwP	2.173	2.509	2.176	2.191	2.209
	Hausdorff	2.514	2.742	4.353	4.448	5.627
	Fréchet	2.358	2.492	3.735	3.824	4.642
	CSTRM	OOM	OOM	OOM	OOM	OOM
	t2vec	4.453	6.736	9.087	9.470	9.775
	TrjSR	24.539	30.318	55.002	68.070	111.175
	E2DTC	11.595	13.478	15.843	18.532	19.134
	CLEAR	1.265	1.276	1.396	1.460	1.740
	TrajCL	1.048	1.050	**1.059**	1.418	2.045
	MovSemCL	**1.001**	**1.008**	1.080	**1.151**	**1.265**

Table 2: Mean rank vs. down-sampling rate (RQ2).

Dataset	Method	0.1	0.2	0.3	0.4	0.5
Porto	EDR	28.243	28.498	27.899	28.070	28.932
	EDwP	7.591	7.166	7.038	7.235	7.236
	Hausdorff	6.549	6.737	6.706	6.592	6.739
	Fréchet	8.689	8.854	8.755	8.636	9.083
	CSTRM	20.860	20.081	22.081	24.688	26.243
	t2vec	3.212	3.487	3.981	3.897	3.999
	TrjSR	4.781	5.087	35.144	6.194	7.201
	E2DTC	3.348	3.678	4.210	4.129	4.222
	CLEAR	1.345	1.313	1.330	1.309	1.356
	TrajCL	1.022	1.154	1.076	1.091	1.039
	MovSemCL	**1.004**	**1.012**	**1.005**	**1.006**	**1.004**
Germany	EDR	1373.985	1372.984	1373.981	1373.966	1373.944
	EDwP	2.488	2.489	2.492	2.489	2.489
	Hausdorff	5.587	5.576	5.573	5.566	5.568
	Fréchet	4.631	4.625	4.609	4.625	4.612
	CSTRM	OOM	OOM	OOM	OOM	OOM
	t2vec	3.863	3.976	4.903	3.580	3.625
	TrjSR	27.146	27.156	27.032	26.935	27.035
	E2DTC	10.946	11.161	10.940	11.275	10.693
	CLEAR	1.223	1.194	1.209	1.190	1.233
	TrajCL	1.049	1.051	1.049	1.062	1.054
	MovSemCL	**1.008**	**1.008**	**1.008**	**1.008**	**1.007**

Table 3: Mean rank vs. distortion rate (RQ2).

the HR@5 metrics. The consistently high R5@20 scores ($>$ 0.95) for Hausdorff and Fréchet indicate that the movement-semantics representations does very well at capturing the geometric properties these measures emphasize.

Efficiency Analysis (RQ4)

To assess deployment feasibility, we compare MovSemCL's computational efficiency with that of the SOTA method TrajCL, which also adopts self-attention-based encoder.

Inference Efficiency Table 5 compares MovSemCL with TrajCL under identical settings (embedding size = 256, batch size = 128, same hardware). MovSemCL achieves significant improvements across all metrics: 41.2% fewer FLOPs, 43.4% faster inference, and 76.6% higher throughput. These gains arise from replacing the global $O(L^2)$ attention with block-wise computations of $O(L \cdot P + M^2)$.

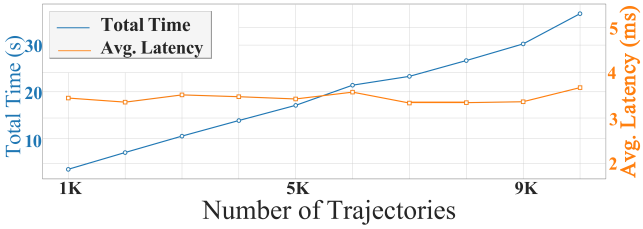


Figure 5: Scalability analysis (RQ4).

Scalability Analysis Figure 5 reports MovSemCL's scalability across varying workloads. The total processing time scales linearly with the number of trajectories, confirming performance without computational bottlenecks. Crucially, the per-sample latency remains remarkably stable (at $\sim$3.4ms) regardless of the dataset size, indicating consistent high efficiency from small batches to large-scale processing.

Ablation Study (RQ5)

To evaluate the contribution of each system component, we remove the movement-semantics encoding (MSE, using only cell embedding), the hierarchical semantics encoding (HSE, using flat sequence processing), and the curvature-guided augmentation (CGA, using random point mask).

Results Table 6 reveals a clear hierarchy: MSE has the most critical impact, with its removal causing severe degradation, offering evidence that movement semantics are essential. CGA provides moderate but consistent improvements, validating the semantics-aware masking. HSE contributes the least but still provides meaningful improvements, confirming that hierarchical encoding is beneficial.

Hyperparameter Study (RQ6)

Finally, we examine MovSemCL's sensitivity to key hyperparameters on Porto to provide implementation guidance.

Training Epoch Figure 6(a) shows that MovSemCL converges rapidly within 10 epochs, with stable performance extending to 20 epochs without overfitting. This rapid convergence reduces training costs while ensuring reliability.

Training Trajectory Size Figure 6(b) shows that performance gains plateau around 20K trajectories for standard conditions, with degraded conditions requiring additional data for optimal performance. This provides practical guidance for minimum training data requirements.

Embedding Dimension Figure 6(c) reveals optimal performance at dimensionalities in the range 256–512, with substantial improvement from smaller dimensions to 256, followed by stabilization. This indicates that the hierarchical encoding efficiently utilizes the embedding space without excessive parameters.

Method	EDR			EDwP			Hausdorff			Fréchet			Average rank
	HR@5	HR@20	R5@20	HR@5	HR@20	R5@20	HR@5	HR@20	R5@20	HR@5	HR@20	R5@20	
t2vec	0.125	0.164	0.286	0.399	0.518	0.751	0.405	0.549	0.770	0.504	0.651	0.883	6
TrjSR	0.137	0.147	0.273	0.271	0.346	0.535	0.541	0.638	0.880	0.271	0.356	0.523	9
E2DTC	0.122	0.157	0.272	0.390	0.514	0.742	0.391	0.537	0.753	0.498	0.648	0.879	7
CSTRM	0.138	0.191	0.321	0.415	0.536	0.753	0.459	0.584	0.813	0.421	0.557	0.768	4
CLEAR	0.158	0.207	0.351	0.487	0.594	0.831	0.596	0.687	0.936	0.583	0.709	0.937	3
TrajCL	0.172	0.222	0.376	**0.546**	**0.646**	**0.881**	0.643	0.721	0.954	0.618	0.740	0.955	2
MovSemCL	**0.207**	**0.308**	**0.487**	0.536	0.642	0.873	0.654	**0.742**	**0.970**	**0.629**	**0.741**	**0.957**	**1**
TrajSimVec	0.119	0.163	0.285	0.172	0.253	0.390	0.339	0.429	0.543	0.529	0.664	0.894	10
TrajGAT	0.090	0.102	0.184	0.201	0.274	0.469	**0.686**	0.740	0.969	0.362	0.403	0.704	8
T3S	0.140	0.192	0.325	0.377	0.498	0.702	0.329	0.482	0.668	0.595	0.728	0.946	5

Table 4: HR@5, HR@20, and R5@20 of self-supervised and supervised methods to approximate heuristic measures on Porto (RQ3).

Metric	TrajCL	MovSemCL	Improv.
FLOPs (M)	158.69	**93.34**	41.2%
Latency (ms)	6.08	**3.44**	43.4%
Throughput (samples/s)	164.46	**290.41**	76.6%

Table 5: Efficiency comparison (RQ4). MovSemCL achieves substantial efficiency improvements over TrajCL.

Method	Porto		Germany	
	20K	100K	20K	100K
MovSemCL-MSE	1.521	3.045	1.595	4.122
MovSemCL-HSE	1.005	1.012	1.010	1.039
MovSemCL-CGA	1.033	1.098	1.180	1.234
MovSemCL	**1.002**	**1.005**	**1.002**	**1.008**

Table 6: Ablation study (RQ5). MSE is the most impactful module, and all modules are effective.

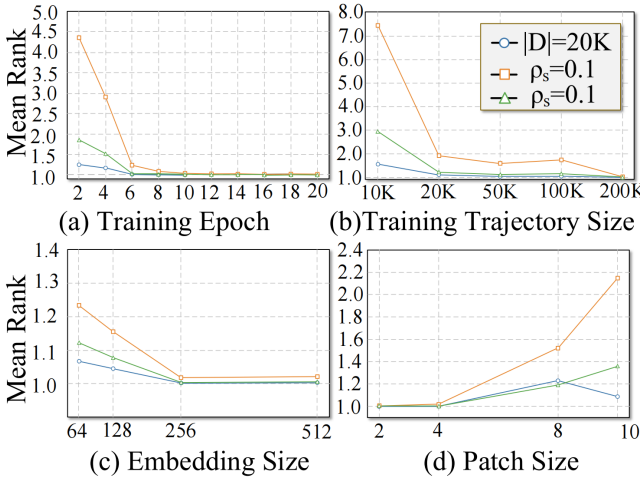


Figure 6: Hyperparameter study (RQ6).

Patch Size Figure 6(d) shows that patch size 4 achieves optimal performance. Smaller patches lack sufficient context, while larger patches dilute movement-semantics.

Related Work

Traditional Trajectory Similarity Computation Traditional methods rely on geometric and statistical principles. Alignment-based approaches adapt string matching algorithms, such as EDR (Chen, Özsu, and Oria 2005) which extends edit distance with spatial thresholds. Geometric methods include Hausdorff distance (Hausdorff 1914), measuring maximum point-to-nearest-neighbor distance, Fréchet distance (Fréchet 1906), considering spatio-temporal ordering, and EDwP (Ranu et al. 2015) projecting trajectories onto a road network. However, these methods are computationally expensive and ignore movement semantics.

Learning-Based Trajectory Representation Deep learning enables vector representations of trajectories that capture complex spatial-temporal patterns. Early approaches like t2vec (Li et al. 2018) use sequence-to-sequence autoencoders, while TrjSR (Cao et al. 2021) and E2DTC (Fang et al. 2021) employ recurrent architectures with attention mechanisms. Recent contrastive learning methods include TrajCL (Chang et al. 2023) with trajectory-specific augmentations and dual-feature attention, and CLEAR (Li, Liu, and Lu 2024) with multi-positive contrastive learning. However, existing methods struggle with hierarchical movement modeling, computational efficiency for trajectories with many locations, and semantically-aware augmentations—limitations that MovSemCL addresses.

Conclusion

We present MovSemCL, a movement-semantics contrastive learning framework that enriches GPS trajectories with movement-semantics features and uses hierarchical patch-based encoding with curvature-guided augmentation. Experiments show that MovSemCL is capable of outperforming state-of-the-art methods, reporting mean ranks close to 1 in similarity search, up to 20.3% improvements in heuristic approximation, and up to 43.4% faster inference, while maintaining the best robustness to data degradation.

Acknowledgments

This work was supported by Independent Research Fund Denmark (No. 1032-00481B).

References

Cao, H.; Tang, H.; Wu, Y.; Wang, F.; and Xu, Y. 2021. On accurate computation of trajectory similarity via single image super-resolution. In *IJCNN*, 1–9.

Chang, Y.; Tanin, E.; Tang, X.; Qi, J.; and Xia, Y. 2023. Contrastive trajectory similarity learning with dual-feature attention. In *ICDE*, 2933–2945.

Chang, Z.; Yu, L.; Li, H.; Wu, S.; Chen, G.; and Zhang, D. 2024. Revisiting CNNs for trajectory similarity learning. *Proc. VLDB Endow.*, 18(4): 1013–1021.

Chen, L.; Özsu, M. T.; and Oria, V. 2005. Robust and fast similarity search for moving object trajectories. *SIGMOD Rec.*, 34(2): 491–502.

Chen, X.; Xu, J.; Zhou, R.; Chen, W.; Fang, J.; and Liu, C. 2020. Learning deep representation for trajectory clustering. *Expert Syst. Appl.*, 144: 113111.

Cormode, G.; and Muthukrishnan, S. 2007. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1): 1–19.

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.

Fang, Z.; Du, Y.; Chen, L.; Hu, Y.; Gao, Y.; and Chen, G. 2021. E2DTC: An end-to-end deep trajectory clustering framework via self-training. In *ICDE*, 696–707.

Feng, J.; Li, Y.; Zhang, C.; Sun, F.; Meng, F.; Guo, A.; and Jin, D. 2018. DeepMove: Predicting human mobility with attentional recurrent networks. In *WWW*, 1459–1468.

Fréchet, M. 1906. Sur quelques points du calcul fonctionnel. *Rend. Circ. Mat. Palermo*, 22(1): 1–74.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*, 855–864.

Hausdorff, F. 1914. *Grundzüge der Mengenlehre*. Leipzig: Veit & Comp.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 9729–9738.

Keogh, E.; and Ratanamahatana, C. A. 2005. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3): 358–386.

Lai, Z.; Lu, H.; Li, H.; Li, J.; and Jensen, C. S. 2025. MovSemCL: Movement-Semantics Contrastive Learning for Trajectory Similarity (Extension). arXiv:2511.12061.

Li, H.; Jin, R.; Dou, S.; Chen, J.; and Xu, J. 2022. CLTSim: A contrastive learning framework for trajectory similarity computation. *IEEE Trans. Knowl. Data Eng.*, 35(8): 8130–8143.

Li, J.; Liu, T.; and Lu, H. 2024. CLEAR: Ranked multi-positive contrastive representation learning for robust trajectory similarity computation. In *MDM*, 21–30.

Li, X.; Zhao, K.; Cong, G.; Jensen, C. S.; and Wei, W. 2018. Deep representation learning for trajectory similarity computation. In *ICDE*, 617–628.

Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, volume 30.

Liu, X.; Tan, X.; Guo, Y.; Chen, Y.; and Zhang, Z. 2022. CSTRM: Contrastive self-supervised trajectory representation model for trajectory similarity computation. *Comput. Commun.*, 185: 159–167.

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

Snyder, J. P. 1987. *Map projections: A working manual*. US Government Printing Office.

Su, H.; Zheng, K.; Huang, J.; Wang, H.; and Zhou, X. 2020. Making sense of trajectory data: A partition-and-summarize approach. In *ICDE*, 973–984.

Vlachos, M.; Kollios, G.; and Gunopulos, D. 2002. Discovering similar multidimensional trajectories. In *ICDE*, 673–684.

Xu, K.; Qin, Z.; Wang, G.; Huang, K.; Ye, S.; and Zhang, H. 2020. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. *arXiv:2005.02545*.

Yang, P.; Wang, H.; Zhang, Y.; Qin, L.; Zhang, W.; and Lin, X. 2021. T3S: Effective representation learning for trajectory similarity computation. In *ICDE*, 2183–2188.

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.

Yao, D.; Zhang, C.; Zhu, Z.; Huang, J.; and Bi, J. 2018. Trajectory clustering via deep representation learning. In *IJCNN*, 3880–3887.

Zhang, H.; Zhang, X.; Jiang, Q.; Zheng, B.; Sun, Z.; Sun, W.; and Wang, C. 2020. Trajectory similarity learning with auxiliary supervision and optimal matching. In *IJCAI*, 11–17.

Zheng, Y. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3): 1–41.