SIMformer: Single-Layer Vanilla Transformer Can Learn Free-Space Trajectory Similarity

Chuang Yang
The University of Tokyo
chuang.yang@csis.u-tokyo.ac.jp

Renhe Jiang*
The University of Tokyo
jiangrh@csis.u-tokyo.ac.jp

Xiaohang Xu
The University of Tokyo
xhxu@g.ecc.u-tokyo.ac.jp

Chuan Xiao
Osaka University
chuanx@ist.osaka-u.ac.jp

Kaoru Sezaki
The University of Tokyo
sezaki@iis.u-tokyo.ac.jp

ABSTRACT

Free-space trajectory similarity calculation, e.g., DTW, Hausdorff, and Fréchet, often incur quadratic time complexity, thus learning-based methods have been proposed to accelerate the computation. The core idea is to train an encoder to transform trajectories into representation vectors and then compute vector similarity to approximate the ground truth. However, existing methods face dual challenges of effectiveness and efficiency: 1) they all utilize Euclidean distance to compute representation similarity, which leads to the severe curse of dimensionality issue – reducing the distinguishability among representations and significantly affecting the accuracy of subsequent similarity search tasks; 2) most of them are trained in triplets manner and often necessitate additional information which downgrades the efficiency; 3) previous studies, while emphasizing the scalability in terms of efficiency, overlooked the deterioration of effectiveness when the dataset size grows. To cope with these issues, we propose a simple, yet accurate, fast, scalable model that only uses a single-layer vanilla transformer encoder as the feature extractor and employs tailored representation similarity functions to approximate various ground truth similarity measures. Extensive experiments demonstrate our model significantly mitigates the curse of dimensionality issue and outperforms the state-of-the-arts in effectiveness, efficiency, and scalability.

PVLDB Reference Format:
Chuang Yang, Renhe Jiang, Xiaohang Xu, Chuan Xiao, and Kaoru Sezaki. SIMformer: Single-Layer Vanilla Transformer Can Learn Free-Space Trajectory Similarity. PVLDB, 18(2): 390 - 398, 2024.
doi:10.14778/3705829.3705853

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/SUSTC-ChuangYANG/SIMformer/.

1 INTRODUCTION

With the rapid development of positioning and sensing technologies, large-scale trajectory data is being collected from various sources like smartphone apps and navigation systems. This data holds immense value and plays crucial roles in fields such as intelligent transportation [12, 31], urban planning [44], and epidemic simulation [38]. In particular, *computing the trajectory similarity* is always a fundamental operation for various trajectory analysis tasks, such as clustering [1, 32], similarity search [20, 26, 37], and anomaly detection [5, 24]. To meet the requirements in different scenarios, many distance functions have been used to calculate the trajectory similarity in free space, such as Dynamic Time Warping (DTW) [26], Hausdorff distance [4], and Fréchet distance [28]. However, the time complexity of these distance measurements is typically $O(mn)$ (where m and n are the lengths of two trajectories), which limits their application in large-scale datasets [10, 15]. To this end, a series of approximate approaches to accelerate computation has been proposed, which can be divided into two categories:

- *Non-learning-based methods* [2, 8, 11, 25] focus on designing more efficient handcrafted approximate algorithms to speed up distance computation. Nevertheless, these methods are often designed for specific one or two distance functions and cannot be easily extended to other trajectory similarity measures.
- *Learning-based methods* [13, 14, 39–42, 46] concentrate on learning a neural network encoder to transform the original trajectory into a d-dimensional representation vector, and then calculate the Euclidean distance between vectors to approximate the target distance measures such as DTW or Hausdorff. This reduces the complexity of trajectory similarity computation to $O(d)$, achieving a shift from quadratic to linear complexity.

Compared to non-learning-based methods, learning-based methods can efficiently perform large-scale similarity calculations on the learned representations. It has been demonstrated that an LSTM-based model can bring a speedup of 50x-1000x over brute-force methods and 3x-500x over non-learning-based methods across three distance measures with better top-k hit and recall rates [41]. Besides, learning-based methods can easily approximate different distance measures by simply changing the objective function, showing stronger expandability over non-learning-based ones. Though remarkable progress has been made, similarity learning models under the free-space setting [39–42, 46] still exhibit the following issues.

(i) Curse of dimensionality and limited effectiveness of learned representations. Existing studies [39–42, 46] have employed *Euclidean distance-based* representation similarity function to approximate the target measures. In fact, this will lead to severe "*curse of dimensionality*" [3] issue, a phenomenon in high-dimensional space such that pairs of objects are not easy to be

*Corresponding Author
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 18, No. 2 ISSN 2150-8097.
doi:10.14778/3705829.3705853

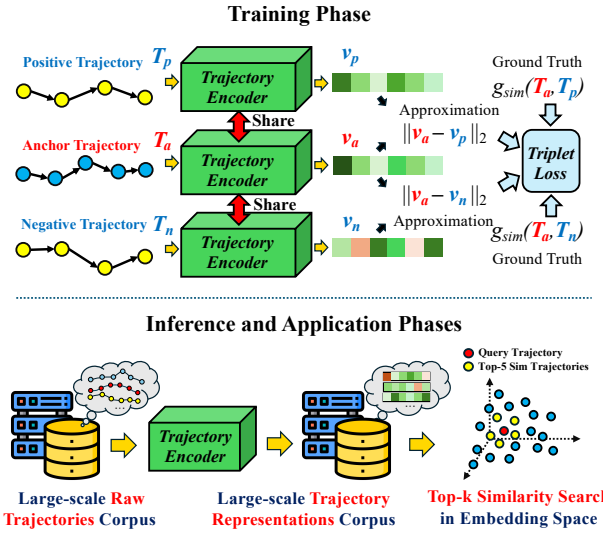


Figure 1: Pipeline of existing learning-based solutions.

distinguished, rendering distances, particularly Euclidean distance, to be less effective. Such indistinguishability of the latent representations essentially results in poor performance in subsequent tasks, such as the k-nearest neighbor (kNN) search. Besides, existing works only evaluated the model effectiveness by checking the top-k hit&recall rate for similarity search. However, *ranking quality* is also a critical aspect. Overlooking it may lead to high hit ratios but poorly disordered rankings, which inevitably requires re-ranking [23]. **(ii) Over-complicated models and limited efficiency.** Following [41], researchers have adopted the triplet loss that constructs positive&negative sample pairs for similarity learning as a common technique [39, 40, 42, 46], as illustrated in Figure 1. However, this choice greatly increases the training cost. Meanwhile, researchers have suggested employing auxiliary information to improve the performance, like mapping trajectory points onto grids for additional structural insights [40–42] or pre-computing sub-trajectory distances as auxiliary supervision [39, 46]. All these make the state-of-the-art (SOTA) models over-complicated and less efficient. Simplifying the existing framework in a minimalism fashion while making it more efficient and effective becomes the ultimate goal of this study. **(iii) Scalability of effectiveness.** Though scalability is often associated with efficiency, we note significant drops in the effectiveness of current methods as the number of trajectories grows. This is because these methods are typically trained and tested – also with ground truth labeled – on a limited portion of the dataset, rendering them less effective against challenging scenarios, such as noise, which predominantly arise in larger datasets. However, this issue has been largely overlooked in prior research.

To cope with these limitations, we propose ***SIMformer***, a *single-layer vanilla transformer encoder* under a *simple pair-wise mean squared error (MSE) loss* framework, without using the typical triplet loss framework or any auxiliary information. In particular, we find that using *tailored representation similarity functions* (e.g., Chebyshev distance for Fréchet and cosine for DTW) instead of *relying solely on Euclidean distance* can greatly alleviate the impact of the curse of dimensionality. This small adjustment allows a solution – only with a 1-layer vanilla transformer and a Siamese network

Table 1: Comparison of model architectures and features.

Model	Core Architecture	Repr. Sim. Func.	Loss Func.	Need Auxiliary Info.	
				Sub-traj.	*Gridification*
NeuTraj [41]	Augmented LSTM	Euclidean	Triplet	No	Yes
Traj2simvec [46]	Vanilla LSTM	Euclidean	Triplet	Yes	No
T3S [40]	LSTM & Attention	Euclidean	Triplet	No	Yes
TMN [39]	LSTM & Attention	Euclidean	Triplet	Yes	No
SIMformer	**Vanilla Transformer**	**Tailored**	**MSE**	**No**	**No**

– to outperform SOTA in terms of accuracy, speed, and scalability. On four widely-used benchmarks, Porto [16], T-Drive [45], Geolife [48], and AIS [22], SIMformer achieves an average improvement of 27.59% top-k hit ratio on DTW, 34.42% on Hausdorff, and 12.80% on Fréchet over the best-performing baseline, and improves ranking quality by reducing 20.08% inversions. Meanwhile, SIMformer is 30% faster in inference and saves 10% memory usage. A scalability test on Porto and its augmented version with random noise shows that SIMformer still retains good accuracy when baselines start to report poor results. Comprehensive theoretical and experimental analysis were conducted to reveal how our method alleviates the curse of dimensionality. Furthermore, SIMformer's applicability in different data and measurement scenarios was also explored.

Our contributions are summarized as follows: 1) We developed a single-layer vanilla transformer with a simple Siamese architecture to achieve the SOTA performance in both effectiveness and efficiency. 2) To the best of our knowledge, we are the first to study the curse of dimensionality in trajectory similarity learning and propose the idea of *tailoring representation similarity functions* to significantly alleviate this issue. 3) We conducted extensive experiments on four widely-used trajectory benchmarks with three distance measures. We also evaluated the *ranking quality* and *scalability* to further demonstrate the superiority of the proposed solution.

2 RELATED WORK

Existing trajectory similarity learning methods can be divided into two categories based on whether the measure considers the topological structure, which can be built from the proximity relationship between trajectory points [14] or the underlying road network [13]. GTS [14] is a graph-based approach proposed for approximate TP distance (an extension of Hausdorff distance on spatial networks). ST2Vec [13] is built for spatial-temporal trajectory similarity learning in road networks, supporting network-based distance measures like LCRS [43] and NetERP [18]. In comparison, *free-space* trajectory similarity learning, which is not constrained by topological structure, has broader applications and attracts more research attention: NeuTraj [41] is an attention-augmented LSTM model that incorporates spatial context by mapping trajectories into grids and introduces a weighted ranking loss (a triplet loss variant) to improve the performance; Traj2simvec [46] uses a sampling strategy with a k-d tree and kNN to accelerate training and sub-trajectory distances for auxiliary supervision; T3S [40] employs self-attention mechanism with LSTM and further utilizes grid information by incorporating the grid sequence of trajectories as supplementary structural data; TMN [39] uses LSTM and cross-attention for explicitly modeling matching information among trajectories to improve accuracy. We outline the model structures and characteristics of these methods in Table 1, along with our model. It is evident that SIMformer is more concise in design, as it does not require any auxiliary information or the construction of triplets for training.

3 PRELIMINARIES

Definition 1. (**Trajectory**): A trajectory T is defined as a time-ordered sequence of locations $\{l_1, l_2, \ldots, l_n\}$, where n is the number of points in the trajectory, and $l_i = (x_i, y_i)$ denotes the i-th point in the trajectory. Following existing studies [39–41, 46], we focus on the shape of trajectories without considering time.

Definition 2. (**Free-space Trajectory Distance Measure**): Given two trajectories T_i and T_j, and a free-space trajectory distance measure M, $d_M(T_i, T_j)$ quantifies the distance between trajectories T_i and T_j without considering the physical constraints in the environment such as the road network [15].

Definition 3. (**Ground Truth Trajectory Similarity**): Given two trajectories T_i and T_j, the ground truth trajectory similarity $g_{sim}(T_i, T_j)$ is the result of applying negative exponential normalization to trajectory distance $d_M(T_i, T_j)$:

$$g_{sim}(T_i, T_j) = \exp(-\alpha * d_M(T_i, T_j)) \in [0, 1], \tag{1}$$

where α is an adjustable parameter, controlling the distribution of similarity values. This transformation was first introduced in [41] and has been widely adopted in later studies It can be treated as a smoothing operation that maps an indefinite range of distance distributions to the range [0, 1], making it easier for models to learn.

Definition 4. (**Approximate Similarity Function**): Given two trajectories T_i and T_j, the approximate similarity function $f(T_i, T_j)$ approximates the true trajectory similarity $g_{sim}(T_i, T_j)$. It comprises a trainable neural network encoder f_{enc} that maps an input trajectory T to a d-dimensional vector v, called the trajectory representation:

$$f_{enc} : T \mapsto v \in \mathbb{R}^d, \tag{2}$$

and a *representation similarity function* $f_{sim}(\cdot, \cdot)$ that calculates the similarity between two trajectory representations with a time complexity of (d). Formally, the approximate similarity function is

$$\begin{aligned} f(T_i, T_j) &= f_{sim}(f_{enc}(T_i), f_{enc}(T_j)), \\ &= f_{sim}(v_i, v_j). \end{aligned} \tag{3}$$

Definition 5. (**Free-space Trajectory Similarity Learning**): Given a free-space trajectory distance measure M, and a training dataset $\mathcal{T}$, the objective is to learn a trajectory encoder f_{enc} such that the discrepancy between the ground truth similarity $g_{sim}(T_i, T_j)$ and the approximated similarity $f(T_i, T_j)$ for trajectory pairs (T_i, T_j) over the training set is minimized, denoted as:

$$\min_{\theta} \sum_{(T_i, T_j) \in \mathcal{T}} \left| g_{sim}(T_i, T_j) - f(T_i, T_j; \theta) \right|, \tag{4}$$

where θ is the parameters of the encoder f_{enc}. In this study, we select three representative free-space distance measures for ground truth similarity: DTW distance, Hausdorff distance, and Fréchet distance. For Fréchet, the discrete version of it is utilized to accommodate the discrete nature of trajectories, inline with previous studies [39–41].

4 METHODOLOGY

Figure 2 shows the overall framework of the proposed solution, which utilizes a Siamese network architecture. Given a pair of trajectories T_i, T_j, the shared trajectory encoder transforms them into d-dimensional representations, and then computes their similarity with the similarity function $f_{sim}(v_i, v_j)$ to approximate the ground truth $g_{sim}(v_i, v_j)$. The Mean Squared Error (MSE) is employed as the training loss. We utilize a single-layer vanilla transformer to extract the features, following a pooling & activation layer to summarize the sequence output, making it length-independent. As for the similarity function, we adopt a tailored approach – customizing the most appropriate one for different target measures. Because of its **sim**plicity and the use of the trans**former** encoder for **sim**ilarity learning, our model is named **SIMformer**.

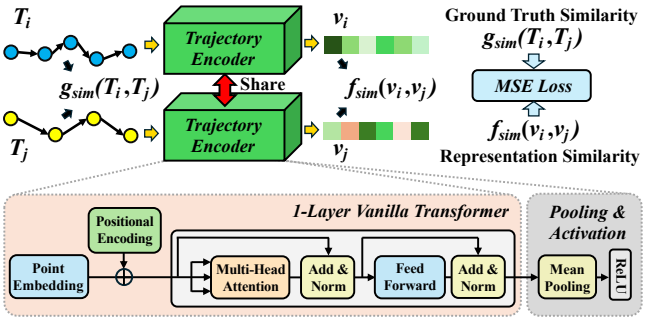


Figure 2: The overall framework of SIMformer.

4.1 1-Layer Transformer Trajectory Encoder

Point Embedding. Each point in the trajectory, denoted as l_i, is mapped to $\mathbb{R}^d$ with a linear layer first. The transformation is defined as: $z_i = W l_i + b$, where $W \in \mathbb{R}^{d \times 2}$ is the weight matrix, $b \in \mathbb{R}^d$ is the bias vector, and $z_i \in \mathbb{R}^d$ is the transformed point embedding.

Positional Encoding. Before feeding the input into the transformer encoder, a common practice is to add *positional encoding*. This ensures the order information of points is integrated, as the transformer encoder processes all input elements in parallel and cannot distinguish their order. Here, we adopt a learnable positional encoding $e_i \in \mathbb{R}^d$ for each position i, expressed as: $h_i = z_i + e_i$.

Transformer Encoder. Now, we obtained a preliminary trajectory representation: $H = [h_1, h_2, \ldots, h_n]^\top \in \mathbb{R}^{n \times d}$. To learn the complex patterns and dependencies within it, H is further fed into a 1-layer vanilla transformer encoder [29], where the multi-head self-attention extracts and fuses the features of key points, producing a sequence of d-dimensional vectors:

$$\begin{aligned} H' &= \text{Transformer Encoder}(H), \\ &= [h'_1, h'_2, \ldots, h'_n]^\top \in \mathbb{R}^{n \times d}. \end{aligned} \tag{5}$$

Pooling and Activation. Lastly, we utilize *mean pooling* to summarize the information within the embedding sequence H', resulting in a d-dimensional representation h'_{avg}: $h'_{avg} = \frac{1}{n} \sum_{i=1}^{n} h'_i$. Finally, an ReLU activation function is applied to h'_{avg} to confine the final representation within the first quadrant, ensuring that the cosine similarity of the output vectors ranges from 0 to 1: $v = \text{ReLU}(h'_{avg}) \in \mathbb{R}^d$.

4.2 Tailored Representation Similarity Function

Given two trajectory representation vectors $v_i, v_j \in \mathbb{R}^d$, existing learning-based methods [39–41, 46] use the conventional method to compute the representation similarity based on Euclidean distance:

$$f_{sim}(v_i, v_j) = \exp(-d_{\text{Euc}}(v_i, v_j)) = \exp\left(-\sqrt{\sum_{k=1}^{d} (v_{i,k} - v_{j,k})^2} \right). \tag{6}$$

Previous studies [3, 19] have indicated that Euclidean distance-based measure may lead to the severe curse of dimensionality issues. It is essential to explore *alternative similarity functions* that are less sensitive to high-dimensional spaces. Here, we consider the alternatives from the perspective of **feasible solution spaces**:

Assuming v_i is the representation vector of anchor trajectory, and v_j is the representation vector of the query trajectory, their ground truth distance is r. When adopting Euclidean distance for approximation, the deep learning model will force v_j to lie on the surface of a d-dimensional **hyperball** B^d with radius r centered at v_i. In other words, given v_i, the feasible solution space for v_j under Euclidean setting is the surface of B^d. However, if we replace Euclidean distance with Chebyshev distance:

$$d_{\text{Cheby}}(v_i, v_j) = \max_k |v_{i,k} - v_{j,k}|, \tag{7}$$

the feasible solution space becomes the surface of a d-dimensional **hypercube** C^d centered at v_i with side length $2r$, where B^d is inscribed within C^d. Similarly, when using cosine:

$$d_{\text{cosine}}(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\|_2 * \|v_j\|_2}, \tag{8}$$

the feasible solution space is the surface of a d-dimensional **hypercone** K^d with apex at v_i and a fixed opening angle. Figure 3 illustrates the shape of the feasible solution spaces for different similarity functions when $d = 3$. The red marker represents the anchor point. For Euclidean, all feasible solutions reside on a ball; for Chebyshev, they lie on a cube's surface with a side length of $2r$; under cosine similarity, all solutions reside on a conical surface where all points maintain a constant angle to the anchor point.

An interesting and counterintuitive fact is that as d increases, the relative surface area of the hyperball compared to that of the hypercone or hypercube approaches zero. This is quite intuitive for hypercone, because its surface expands infinitely, and so have $A_{\text{hypercone},d}(\phi) = \infty$. For the hyperball and hypercube, the proof is as follows: the surface area of the hyperball [21] with radius r in dimension d is given by:

$$A_{\text{hyperball},d}(r) = \frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1}, \tag{9}$$

where Γ is the gamma function [33], and the surface area of the hypercube [34] with side length $2r$ is:

$$A_{\text{hypercube},d}(2r) = 2d \cdot (2r)^{(d-1)} = d \cdot 2^d \cdot r^{d-1}. \tag{10}$$

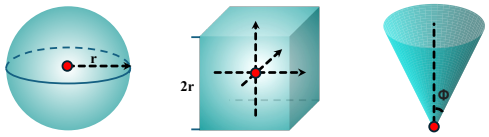
Then, the ratio R of the surface area of the hyperball to that of the hypercube is:

$$R = \frac{A_{\text{hyperball},d}(r)}{A_{\text{hypercube},d}(2r)} = \frac{\frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1}}{d \cdot 2^d \cdot r^{d-1}} = \frac{2\pi^{d/2}}{\Gamma(d/2) \cdot d \cdot 2^d}. \tag{11}$$

Using Stirling's approximation for the Gamma function:

$$R \approx \frac{2\pi^{d/2}}{\sqrt{2\pi \frac{d}{2}} \left(\frac{d/2}{e}\right)^{d/2} \cdot d \cdot 2^d} = \left(\frac{2}{\sqrt{\pi d} \cdot d}\right) \left(\frac{\pi e}{2d}\right)^{d/2}. \tag{12}$$

As $\lim_{d\to\infty} \left(\frac{\pi e}{2d}\right)^{d/2} = 0$, and $\lim_{d\to\infty} \frac{2}{\sqrt{\pi d} \cdot d} = 0$. Thus, $R \to 0$ when $d \to \infty$. Therefore, we conclude that replacing Euclidean distance with cosine similarity or Chebyshev distance results in a larger potential solution space as dimension increases. Relatively, the solution space for Euclidean contracts into a small area, which is the key source of the curse of dimensionality [30]. Experimental validation of this theoretical assertion is presented in Section 5.5.



(a) Euclidean (b) Chebyshev (c) Cosine

Figure 3: The feasible solution spaces for different similarity functions in three-dimensional space.

Despite these advantages, we find that there is **no one-fits-all solution**. It is crucial to carefully choose the appropriate similarity function for each distance measure. Specifically, all existing free-space distance measures require finding the optimal matching point for each point [15, 39]. However, the way these matching points are utilized differs significantly: DTW accumulates the distances of all matching point pairs along the optimal warping path, whereas Hausdorff and Fréchet distances emphasize the maximum distance among all matching pairs. Therefore, the similarity function for DTW should accentuate global features, while for Hausdorff and Fréchet, it should emphasize critical local differences. Based on this observation, we propose a **heuristic approach** for further selecting representation similarity functions: when the distance measure considers accumulated global differences (as in DTW), cosine similarity is more suitable because it accounts for global variations, while for measures focused on the best matching pair, such as Hausdorff and Fréchet, using Chebyshev that natively considered only the maximum component difference becomes a more elegant solution. Such observation is confirmed in our experiments (Table 3).

As such, we propose to use the "*tailored representation similarity function*", whose core idea of is to identify an appropriate similarity based on the specific target measure, rather than rigidly adhering to Euclidean distance. Equation 13 summarizes the cases for the three distance measures discussed in this paper.

$$f_{sim}(v_i, v_j) = \begin{cases} d_{\text{cosine}}(v_i, v_j) & \text{if } d_M = \text{DTW}, \\ \exp(-d_{\text{Cheby}}(v_i, v_j)) & \text{if } d_M = \text{Haus./Fréc.} \end{cases} \tag{13}$$

Finally, we employ the MSE loss $\mathcal{L}$ as the optimization objective:

$$\mathcal{L} = \frac{1}{N \cdot S} \sum_{i=1}^{N} \sum_{j=1}^{S} \left(f_{sim}(v_i, v_j) - g_{sim}(T_i, T_j)\right)^2, \tag{14}$$

where N is the number of trajectories in the training set, and S is the number of trajectories randomly sampled to form input pairs for each T_i.

5 EXPERIMENT

5.1 Experimental Setup

Datasets. We utilized four trajectory benchmarks, namely Porto [16], T-Drive [45], Geolife [48], and AIS [22], which encompass taxi trajectories in Porto and Beijing, and individuals' movement trajectories in Beijing, and vessel trajectories in U.S. waters, respectively. They have been used in previous studies [15, 39–41, 46]. For Porto and Geolife, we conducted the same data preprocessing as in [39, 41]:

trajectories that are too long (> 200 points), too short (< 10 points), or located too far from the central city area are removed. For T-Drive and AIS, which primarily consist of *long-term, continuously tracked trajectories* that have not been segmented into trips, we first segmented them according to stay points [47], Then, the segmented trajectories were processed following the same procedures as Geolife and Porto. Moreover, the spatial span of the AIS data nearly covers all U.S. waters. Hence, for this study, we only filtered out data for 2021 in Pacific Islands. Considering the high cost of calculating ground truth distance as reported in [40], we randomly sampled 10,000 trajectories from each dataset for experimentation (except for the scalability test), consistent with previous studies [39–41]. For scalability, we also considered an augmented version of Porto with **1M** trajectories by duplication and inclusion of random noise (Section 5.4). Table 2 summarizes the dataset statistics.

Baselines. We used the following methods as baselines: *NeuTraj* [41] incorporates a spatial attention memory-enhanced LSTM as encoder and distance-weighted triplet loss; *Traj2simvec* [46] uses the vanilla LSTM as encoder and sub-trajectory distances as extra supervision information; *T3S* [40] combines self-attention and LSTM in an encoder to extract both structured and spatial information; *TMN* [39] employs LSTM and cross-attention to cope with intra-and inter-trajectory information. Since its output representation is pair-specific and cannot be reused for large-scale computation [9], we used the **n**on-**m**atching version suggested by the authors, named TMN-NM. NeuTraj and TMN's source codes were available. Other baselines were implemented by ourselves. Additionally, we implemented three ablation variants of SIMformer using different similarity functions: Euclidean, cosine, and Chebyshev.

Parameter Settings. For each dataset, we divided it into training, validation, and test sets with a ratio of 2:1:7. The hidden dimension d of our model was set to 128, with a learning rate of 0.0005. We used a 1-layer transformer encoder with 16 heads for multi-head self-attention. The batch size for training was set to 20. Following the same sampling number used in most baselines (NeuTraj, T3S, and TMN), each trajectory was randomly paired with 20 other trajectories to construct the training samples, and the traj-dist library [6] was used to compute the ground truth. For α, we assigned a value of 16 for DTW, and 8 for both Hausdorff and Fréchet distances, consistent with previous studies [39–41]. Each experiment was run 3 times, and the average results were reported.

Environment. Experiments were conducted on Intel Xeon Silver 4210R CPU @ 2.40GHz and 4 NVidia GeForce RTX 3090 GPUs.

Evaluation Metrics. Following existing works [39–41, 46], we utilize the *top-k hit ratio* (HR@k) and *top-k recall for top-t ground truth* (Recall-t@k) as the evaluation metrics. Both are used to evaluate the effectiveness of models in top-k similarity search: HR@k measures the overlap between the top-k trajectories retrieved using the approximate similarity and those retrieved using the ground truth similarity. Recall-t@k evaluates the model's ability to recall the top-t ground truth items within the retrieved top-k results. Moreover, there are several methods for assessing the *ranking quality*, such as Spearman's/Kendall rank correlation coefficient [36] and the number of inversions [35]. We opted the latter one, as it directly reflects the re-ranking cost within the retrieved top-k list. We compared the order of approximate similarity with the order of ground truth similarity to calculate the inversions. Smaller values are preferred.

Table 2: Statistical information of the datasets.

Dataset	*Porto*	*T-Drive*	*Geolife*	*AIS*
Data source	Taxi	Taxi	Human	Vessel
#Moving objects	442	10,357	182	82,030
Length range	[10, 200]	[10, 200]	[10, 200]	[10, 200]
Longitude range	[-9.0, -7.9]	[115.9, 117.0]	[115.9, 117.0]	[-158.51, -157.41]
Latitude range	[40.7, 41.8]	[39.6, 40.7]	[39.6, 40.7]	[20.62, 21.72]
#Trajs.	599,632	15,314	11,169	10,700
#Average length	49.86	69.99	76.73	80.45

5.2 Effectiveness Evaluation

Top-k Similarity Search Performance. The performance of different models on the top-k similarity search task is displayed in Table 3. The best and second-best results are highlighted in **bold** and grey, respectively. SIMformer with tailored simlarity function significantly outperforms other models across both datasets and all three distance measures. Particularly, on the Porto dataset with DTW as the approximation target, its HR@1 surpasses SOTA by 41.68%. Additionally, on Porto, It achieves 94.90%, 98.11%, and 98.21% R10@50 across the three distance measures, demonstrating its potential for practical applications. Furthermore, SIMformer *w/ Euc.* achieves the second-best results for the Hausdorff across all datasets and on parts of the Fréchet. This underscores the superiority of transformers in capturing global key features compared to LSTM, which is used by all baseline models. Nevertheless, comparing the variants of SIMformer reveals that the primary performance enhancement still comes from selecting the most suitable similarity function, as it notably reduces the concentration effect (elaborated in Section 5.5), making the learned representations more discriminative in high-dimensional space. Another point worth noting is, all models, including SIMformer, exhibit decreased performance on T-Drive, Geolife and AIS compared to Porto. This is because that the average lengths in these datasets are much longer, while longer trajectories often correspond to more intricate shapes, making them more challenging to model. However, for Geolife and AIS, which are both collected in open-space environments - despite AIS having a longer average length, the model performs better on AIS. We attribute this to that Geolife encompasses multiple transportation modes (walking, cycling, driving, etc.), leading to higher complexity in spatiotemporal features [15], making it more difficult to learn.

Ranking Quality. Table 4 shows the ranking quality on Porto dataset. SIMformer achieves best performance across all settings. Notably, as k increases, the advantage of our model becomes more pronounced, reaching up to 28.26% (Porto, Fréchet, $k = 100$). This displays the high order consistency between the similarity predicted by SIMformer and the ground truth. Additionally, in most cases, all variants of SIMformer demonstrate better ranking quality than baselines. This is because, unlike baseline models that construct triplets (positive & negative samples) for training, SIMformer randomly samples trajectory pairs and treats them equally with the MSE loss. This strategy allows our models to focus more on learning the general law of the similarity distribution rather than just the extremes, leading to better ranking performance. Moreover, by comparing the three variants of SIMformer, we can find that the tailored similarity function is another key factor in improving ranking performance. It alleviates the curse of dimensionality, thereby reducing the concentration effect, enabling the model to better capture the real patterns in the data. We validated these conclusions

Table 3: Effectiveness in top-k similarity search performance (↑)

Dataset	Model	Top-k Similarity Search@DTW				Top-k Similarity Search@Hausdorff				Top-k Similarity Search@Fréchet			
		HR@1	*HR@10*	*HR@50*	*R10@50*	*HR@1*	*HR@10*	*HR@50*	*R10@50*	*HR@1*	*HR@10*	*HR@50*	*R10@50*
Porto	NeuTraj [41]	0.2601	0.4330	0.5540	0.7843	0.2629	0.4043	0.5434	0.7667	0.4056	0.5663	0.6756	0.9324
	Traj2simvec [46]	0.2599	0.4540	0.5826	0.8244	0.3324	0.4706	0.5893	0.8093	0.4392	0.5999	0.6906	0.9402
	T3S [40]	0.2439	0.4305	0.5630	0.7964	0.3493	0.5386	0.6668	0.9226	0.4623	0.6140	0.7094	0.9597
	TMN-NM [39]	0.2542	0.4440	0.5748	0.8106	0.3905	0.5714	0.6861	0.9427	0.4594	0.6146	0.7088	0.9614
	SIMformer *w/ Euc.*	0.2580	0.4503	0.5809	0.8205	0.4394	0.6137	0.7009	0.9595	0.4587	0.6101	0.7017	0.9590
	SIMformer *w/ Cos.*	**0.3685**	**0.5939**	**0.7337**	**0.9490**	0.4174	0.6051	0.7020	0.9560	0.4410	0.6164	0.7065	0.9565
	SIMformer *w/ Cheby.*	0.2736	0.4696	0.6001	0.8272	**0.4990**	**0.6971**	**0.7984**	**0.9811**	**0.5378**	**0.7174**	**0.8170**	**0.9821**
	Improvement	41.68%	30.81%	25.94%	15.11%	27.78%	22.00%	16.37%	4.07%	16.33%	16.73%	15.17%	2.15%
T-Drive	NeuTraj [41]	0.1018	0.2271	0.3564	0.4931	0.1334	0.3104	0.4857	0.6857	0.2630	0.4494	0.5710	0.8340
	Traj2simvec [46]	0.1188	0.2659	0.4117	0.5801	0.1459	0.3205	0.4888	0.6767	0.2494	0.4300	0.5530	0.8054
	T3S [40]	0.0919	0.2258	0.3691	0.5026	0.1935	0.4110	0.5701	0.8214	0.3174	0.4980	0.6132	0.8884
	TMN-NM [39]	0.0983	0.2334	0.3757	0.5248	0.2192	0.4442	0.6041	0.8623	0.3243	0.5085	0.6214	0.8988
	SIMformer *w/ Euc.*	0.1105	0.2436	0.3833	0.5289	0.3697	0.5708	0.6831	0.9543	0.3452	0.5250	0.6335	0.9126
	SIMformer *w/ Cos.*	**0.1988**	**0.3600**	**0.5256**	**0.7394**	0.3440	0.5346	0.6415	0.9284	0.2847	0.4519	0.5547	0.8468
	SIMformer *w/ Cheby.*	0.1450	0.2916	0.4372	0.5880	**0.4366**	**0.6679**	**0.7885**	**0.9804**	**0.4104**	**0.6135**	**0.7322**	**0.9441**
	Improvement	67.33%	35.38%	27.66%	27.46%	18.08%	17.01%	15.44%	2.73%	18.87%	16.85%	15.59%	3.46%
Geolife	NeuTraj [41]	0.1865	0.3165	0.4319	0.6107	0.2093	0.3558	0.5109	0.6843	0.3201	0.5163	0.6685	0.8587
	Traj2simvec [46]	0.1898	0.3394	0.4733	0.6649	0.2319	0.3890	0.5361	0.7065	0.3320	0.5308	0.6775	0.8664
	T3S [40]	0.1550	0.2876	0.4167	0.5954	0.1834	0.3587	0.5362	0.7130	0.2827	0.4976	0.6680	0.8559
	TMN-NM [39]	0.1792	0.3058	0.4233	0.6043	0.2781	0.4838	0.6461	0.8427	0.3351	0.5393	0.6906	0.8871
	SIMformer *w/ Euc.*	0.2001	0.3286	0.4368	0.6271	0.3502	0.5678	0.7047	0.9047	0.3519	0.5646	0.7058	0.9062
	SIMformer *w/ Cos.*	**0.2331**	**0.3859**	**0.5203**	**0.7005**	0.2991	0.4961	0.6349	0.8385	0.3059	0.4892	0.6293	0.8400
	SIMformer *w/ Cheby.*	0.1873	0.3125	0.4240	0.6329	**0.3870**	**0.6137**	**0.7628**	**0.9241**	**0.3654**	**0.5823**	**0.7473**	**0.9155**
	Improvement	16.49%	13.70%	9.93%	5.35%	10.51%	8.08%	8.24%	2.14%	3.84%	3.13%	5.88%	1.03%
AIS	NeuTraj [41]	0.1618	0.3377	0.4926	0.6912	0.1514	0.3229	0.4999	0.6614	0.2901	0.5126	0.6581	0.8557
	Traj2simvec [46]	0.1813	0.3604	0.5046	0.7184	0.2225	0.4038	0.5726	0.7240	0.3643	0.5916	0.7150	0.9126
	T3S [40]	0.1487	0.3298	0.5005	0.6901	0.1149	0.3012	0.5154	0.6710	0.3292	0.5664	0.7103	0.8959
	TMN-NM [39]	0.1541	0.3326	0.4951	0.6861	0.2465	0.4799	0.6648	0.8496	0.3758	0.6013	0.7274	0.9272
	SIMformer *w/ Euc.*	0.1589	0.3308	0.4883	0.6808	0.3166	0.5611	0.7232	0.9172	0.3730	0.5969	0.7251	0.9249
	SIMformer *w/ Cos.*	**0.2173**	**0.4215**	**0.6005**	**0.7816**	0.2403	0.4531	0.6313	0.8211	0.2773	0.4870	0.6417	0.8521
	SIMformer *w/ Cheby.*	0.1766	0.3558	0.5126	0.7048	**0.3558**	**0.5968**	**0.7588**	**0.9276**	**0.3952**	**0.6283**	**0.7675**	**0.9326**
	Improvement	19.83%	16.95%	19.02%	8.80%	12.36%	6.36%	4.93%	1.13%	5.15%	4.49%	5.51%	0.58%

Table 4: Effectiveness in ranking error (↓) on Porto dataset.

Model	DTW			Hausdorff			Fréchet		
	k=10	*k=50*	*k=100*	*k=10*	*k=50*	*k=100*	*k=10*	*k=50*	*k=100*
NeuTraj [41]	15.75	399.92	1543.90	16.12	423.98	1643.34	15.23	364.74	1357.41
Traj2simvec [46]	16.29	400.09	1528.22	14.89	391.55	1526.35	14.35	343.87	1297.75
T3S [40]	16.26	406.03	1552.41	15.49	376.12	1394.67	14.39	338.65	1257.39
TMN-NM [39]	16.26	401.92	1532.77	15.03	362.67	1342.57	14.57	345.03	1279.75
SIMformer *w/ Euc.*	16.07	398.30	1518.62	14.18	343.84	1289.51	14.27	344.53	1284.90
SIMformer *w/ Cos.*	**14.30**	**326.53**	**1185.95**	14.14	342.07	1283.07	14.11	336.33	1263.00
SIMformer *w/ Cheby.*	15.67	384.22	1457.39	**12.70**	**278.79**	**984.27**	**12.15**	**261.61**	**902.05**
Improvement	8.73%	15.01%	18.62%	10.17%	18.50%	23.29%	13.88%	22.22%	28.26%

Table 5: Training and inference efficiency on Porto.

Model	# Params	Training		Inference	
		t_{epoch}	t_{total}	*GPU Usage*	t_{infer}
NeuTraj [41]	**0.12M**	96.39s	9542.24s	2197 MiB	58.370 ms
Traj2simvec [46]	0.25M	**5.60s**	**1233.01s**	1149 MiB	4.759 ms
T3S [40]	155.15M	25.96s	7841.42s	3395 MiB	7.533 ms
TMN-NM [39]	0.18M	23.97s	7455.89s	1660 MiB	8.913 ms
SIMformer	0.16M	7.53s	3236.66s	**1025 MiB**	**3.419 ms**

across all datasets; results for other datasets are available in our code repository due to space limitations.

5.3 Efficiency Evaluation

We evaluated the training and inference efficiency of different models. Here, "inference" refers to the process of converting trajectories into representations. We used Porto with 2,000 trajectories to assess training efficiency and 10,000 trajectories to assess inference efficiency. During the training phase, we adhered to the batch size recommendations specified in the original papers of the baseline models. For the inference phase, we set the batch size to 1 for all models to ensure a fair comparison. Each experiment was conducted 10 times, and the average values were reported.

The efficiency is displayed in Table 5, including the comparison of model size. SIMformer has the fastest speed and the lowest GPU memory usage among all models during inference. This can be attributed to two factors: 1) the simplicity of our model – a single-layer transformer encoder; and 2) the inherent parallelism of the self-attention mechanism [29], allowing it to be more efficient than LSTM models that require sequential processing of data [17]. Additionally, our model ranks second in training efficiency, boasting fewer parameters and a shorter training time. Traj2simvec distinguishes itself with the shortest training time, attributed to its unique and efficient sampling strategy, which only necessitated one pair of positive and negative instances per anchor trajectory. Despite incorporating many advanced techniques, NeuTraj does not significantly increase the parameter count of the LSTM itself, resulting in the least number of parameters. In contrast, T3S uses a large trainable look-up table, leading to the highest number of parameters.

5.4 Scalability Evaluation

Efficiency Scalability. Following [41], we evaluated the efficiency scalability. We sampled five subsets from Porto, ranging from 1k to 500k. The average query processing time (including inference time) for a top-50 similarity search is reported in Table 6. We only compared SIMformer with brute-force and non-learning methods, as learning-based methods merely differ in inference time. The

Table 6: Scalability evaluation on Porto.

Measures	Methods	Average Query Processing Time (top-50)				
		1k	*5k*	*10k*	*200k*	*500k*
DTW	BruteForce	0.285s	1.423s	2.857s	56.359s	142.990s
	Non-learning [25]	0.152s	0.716s	1.418s	28.244s	69.314s
	SIMformer	**0.0181s**	**0.0188s**	**0.0195s**	**0.0221s**	**0.0226s**
Hausdorff	BruteForce	0.148s	0.739s	1.479s	28.921s	75.495s
	Non-learning [27]	0.056s	0.184s	0.360s	7.053s	17.404s
	SIMformer	**0.0122s**	**0.0122s**	**0.0124s**	**0.0127s**	**0.0127s**
Fréchet	BruteForce	0.248s	1.244s	2.493s	50.149s	125.722s
	Non-learning [8]	0.173s	0.808s	1.534s	31.511s	78.644s
	SIMformer	**0.0166s**	**0.0168s**	**0.0168s**	**0.0195s**	**0.0195s**

Measures	Methods	R10@100				
		10k	*100k*	*200k*	*500k*	*1M (aug.)*
DTW	Traj2simvec	0.8650	0.6386	0.5625	0.4766	0.4079
	TMN-NM	0.8747	0.6543	0.5756	0.4882	0.4215
	SIMformer	**0.9645**	**0.8105**	**0.7412**	**0.6563**	**0.5789**
Hausdorff	Traj2simvec	0.8156	0.6483	0.6191	0.5806	0.5679
	TMN-NM	0.8719	0.6902	0.649	0.5979	0.5908
	SIMformer	**0.9895**	**0.9209**	**0.8878**	**0.8313**	**0.7990**
Fréchet	Traj2simvec	0.9247	0.8218	0.7729	0.7079	0.6855
	TMN-NM	0.9399	0.8429	0.7994	0.7392	0.7234
	SIMformer	**0.9594**	**0.8788**	**0.8377**	**0.7787**	**0.7560**

specialized non-learning algorithms for each measure were implemented, including a DTW approximation algorithm [25] that offers optimal or near-optimal alignments with time and memory complexity of $O(n)$; a linear-time greedy algorithm to approximate the discrete Fréchet [8]; and a near-linear complexity algorithm for computing the Hausdorff distance [27]. The results indicate that all methods exhibit linear growth w.r.t. dataset size; but SIMformer demonstrates a significantly more moderate growth rate, achieving over 1000x speedup than non-learning methods on the 500k dataset.

Effectiveness Scalability. The scalability of SIMformer was also assessed concerning similarity search accuracy. Five subsets, from Porto, ranging from 10k to 500k, were randomly sampled (excluding the 2k data used for training). The dataset was further augmented to 1M by duplicating it and adding Gaussian noise ($\mu = 0, \sigma = 1$, equivalent to 100 meters) to the duplicated trajectory points. R10@100 performance was assessed using 1,000 queries. We compared SIMformer with Traj2simvec and TMN-NM, the best-performing baselines for DTW and Hausdorff/Fréchet in terms of hit ratio and recall (Table 3). From the scalability results in Table 6, the general trend is that all competitors exhibit decreasing effectiveness when the dataset size grows. Nonetheless, SIMformer maintains a hit rate of over 50% for DTW and over 75% for the other two distances, even on the 1M dataset augmented with noise. In contrast, the performances of Traj2simvec and TMN-NM deteriorate sharply, with the recalls on DTW drop below 50% at 500k. For Hausdorff, SIMformer's recall on the 500k dataset was even higher than Traj2simvec's on the 10k dataset. Moreover, it is evident that the gaps between SIMformer and the baselines are more remarkable on 100k+ datasets across all three distance measures and even exceed 20% for Hausdorff, showcasing the superior scalability of SIMformer.

5.5 Interpretability Study

According to the theoretical analysis in Section 4.2, Euclidean-based similarity functions are anticipated to have a significantly smaller feasible solution space compared to tailored ones. This assertion is validated here. For SIMformer with and without utilizing the tailored similarity function, we respectively calculated the mean and the standard deviation (STD) of learned representations across all 128 dimensions. Figure 4 shows the mean and STD for each dimension when the dataset is Porto and target measure is DTW. It indicates that *minimal variation exists within each dimension when employing the Euclidean-based* similarity function, suggesting that the solutions are confined to a small area. Conversely, utilizing the tailored function allows for more flexibility across dimensions, aligning with our theoretical judgement. This phenomenon is called concentration effect [49] in previous literature, leading to poor performance in downstream tasks. Notably, this effect is observed consistently across all four datasets × three distance measures. For other datasets and measures, we also computed the STD of the representations for each dimension. The averaged values is reported (Table 7), confirming that our previous conclusion again.

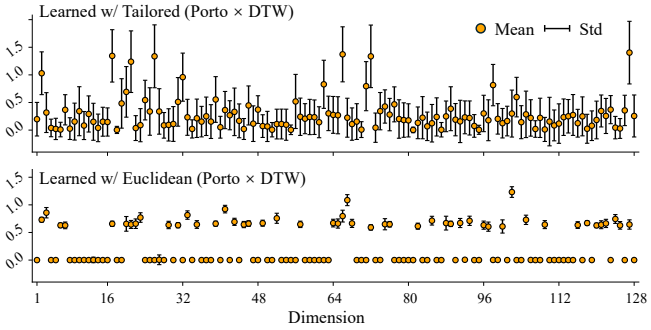


Figure 4: Distributions of learned representations.

Table 7: Average STD of learned representations.

Measures	Repr. Sim. Func.	Dataset			
		Porto	*T-Drive*	*Geolife*	*AIS*
DTW	Euclidean	0.0315	0.0324	0.0336	0.0674
	Tailored	**0.3114**	**0.2559**	**0.2714**	**0.2628**
Hausdorff	Euclidean	0.0248	0.0802	0.0478	0.0834
	Tailored	**0.0647**	**0.1517**	**0.1034**	**0.1481**
Fréchet	Euclidean	0.0303	0.0903	0.0487	0.0985
	Tailored	**0.0817**	**0.2306**	**0.1502**	**0.2145**

5.6 Hyperparameter Study

We tested three hyperparameters of SIMformer's encoder: the number of heads in multi-head self-attention, hidden dimensions of representation, and the number of transformer encoder layers. Figure 5 presents the results on Porto, from which we can see that by increasing the scale of SIMformer, the model performance can be further improved. For example, increasing the number of layers from 1 to 4 led to 3% performance boost across the hit ratios on DTW. However, these improvements are accompanied by an increase in model complexity. Considering the model efficiency and simplicity, we opted for moderate parameters: 16 heads, 128 dimensions, and 1 layer for the encoder.

6 DISCUSSION

Applicability vs. Feature Distribution. The tailored similarity functions proposed in this paper are designed to mitigate the curse of dimensionality. However, *if the curse of dimensionality is not severe in the data, the improvement from our method may also be limited.* Theoretically, it has been proved that singly distributed data often

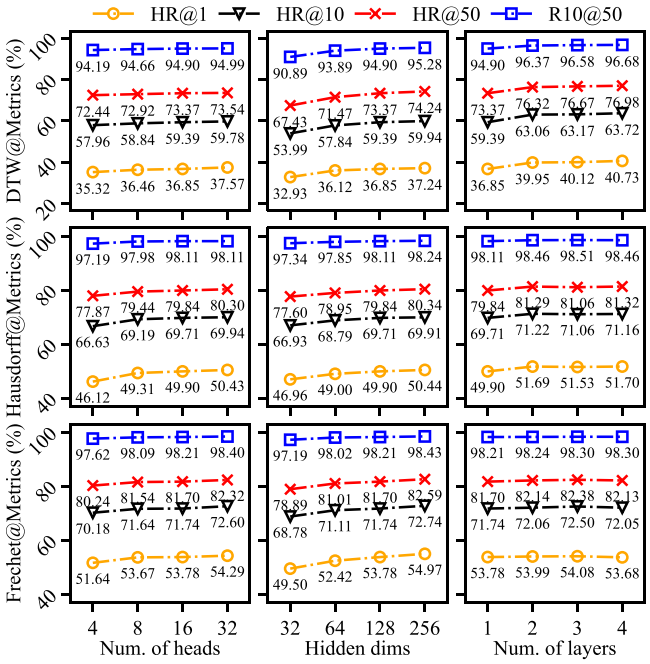


Figure 5: Encoder's hyperparameter analysis on Porto.

suffer more from the curse of dimensionality than multiply distributed data because the different distributions inherently provide some separability [7, 49]. To illustrate this, we plotted the ground truth similarity distributions for various datasets (Figure 6(a)). It is evident that the similarity distributions vary significantly across different datasets and distance measures. Overall, the similarity distribution for T-Drive is simple, followed by Porto, with Geolife and AIS becoming increasingly complex. Specifically, Geolife and AIS can be considered multiply distributed data: Geolife includes multiple transportation modes, while AIS data are highly unevenly distributed along fixed shipping routes [15]. We then calculated the relative performance improvement (on top-k query accuracy) achieved by replacing Euclidean distance with tailored ones and displayed these results in bar charts (Figure 6(b)). We can find that *as the data distribution becomes more complex, the performance improvement diminishes*. Consequently, we can hypothesize that, in real-world scenarios, if the spatial distribution of trajectory data is highly uneven (e.g., containing many separated clusters), the benefits of using tailored similarity functions may be less pronounced.

Applicability vs. Distance Measures. The proposed framework only addresses two specific cases (Equation 13), leaving another important category of distance measures: edit distance-based measures like EDR and ERP, unaddressed. A key characteristic of these measurements is that their output consists of two **heterogeneous** components: the **matched** part and the **unmatched** part. In contrast, DTW sums all matches, whereas Fréchet and Hausdorff only consider the best one match. We have conducted some exploratory experiments and found that Euclidean, cosine, and Chebyshev all perform poorly on these heterogeneous measurements. Cosine only considers directional similarity, making it hard to precisely account for both components simultaneously. Chebyshev focuses solely on optimal local information and fails to handle the overall matching

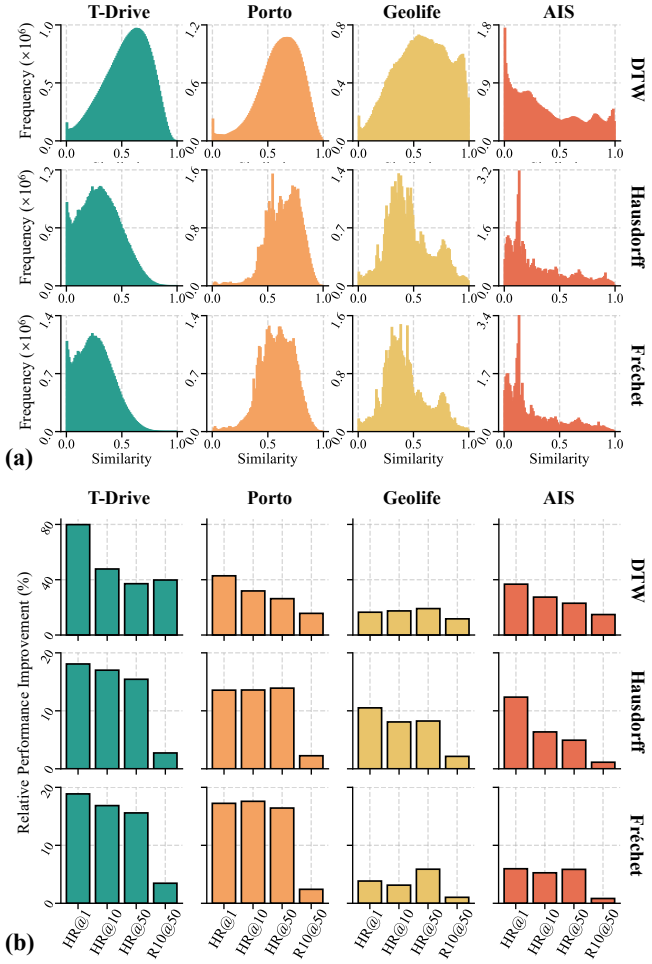


Figure 6: (a) Ground truth similarity distribution across different datasets and distance measures. (b) Relative performance improvements in top-k query accuracy by replacing Euclidean-based similarity function with the tailored ones.

situation. While Euclidean distance has an advantage in precision and can capture global information, it still suffers from the curse of dimensionality issue. Therefore, there is a need to *tailor another kinds of similarity functions* that can mitigate the curse of dimensionality while precisely measuring the heterogeneous components.

7 CONCLUSION

In this study, we proposed a simple yet powerful method for free-space trajectory similarity learning. Extensive experiments demonstrated the effectiveness, efficiency, and scalability of our method. We will further investigate tailored similarity functions for a wider range of distance measures, aiming to revealing the overarching design principles in the future.

ACKNOWLEDGMENTS

This work is supported by JST SPRING JPMJSP2108, JSPS KAKENHI Grant Number JP24K02996, JP23K17456, JP23K25157, JP23K28096, and JST CREST Grant Number JPMJCR21M2, JPMJCR22M2.

REFERENCES

[1] Pankaj K Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. 2018. Subtrajectory clustering: Models and algorithms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 75–87.

[2] Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. 2016. Approximating Dynamic Time Warping and Edit Distance for a Pair of Point Sequences. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, Vol. 51. 6:1–6:16.

[3] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*. Springer, 420–434.

[4] Stefan Atev, Grant Miller, and Nikolaos P Papanikolopoulos. 2010. Clustering of vehicle trajectories. *IEEE transactions on intelligent transportation systems* 11, 3 (2010), 647–657.

[5] Asma Belhadi, Youcef Djenouri, Jerry Chun-Wei Lin, and Alberto Cano. 2020. Trajectory outlier detection: Algorithms, taxonomies, evaluation, and open challenges. *ACM Transactions on Management Information Systems (TMIS)* 11, 3 (2020), 1–29.

[6] Philippe C Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. 2016. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems* 17, 11 (2016), 3306–3317.

[7] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is "nearest neighbor" meaningful?. In *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*. Springer, 217–235.

[8] Karl Bringmann and Wolfgang Mulzer. 2016. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry* 7, 2 (2016), 46–76.

[9] Yanchuan Chang, Egemen Tanin, Gao Cong, Christian S Jensen, and Jianzhong Qi. 2024. Trajectory similarity measurement: An efficiency perspective. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2293–2306.

[10] Wei Chen, Yuxuan Liang, Yuanshao Zhu, Yanchuan Chang, Kang Luo, Haomin Wen, Lei Li, Yanwei Yu, Qingsong Wen, Chao Chen, et al. 2024. Deep learning for trajectory data management and mining: A survey and beyond. *arXiv preprint arXiv:2403.14151* (2024).

[11] Connor Colombe and Kyle Fox. 2021. Approximating the (Continuous) Fréchet Distance. In *37th International Symposium on Computational Geometry (SoCG 2021)*, Vol. 189. 26:1–26:14.

[12] Zipei Fan, Quanjun Chen, Renhe Jiang, Ryosuke Shibasaki, Xuan Song, and Kota Tsubouchi. 2019. Deep multiple instance learning for human trajectory identification. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 512–515.

[13] Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S Jensen. 2022. Spatio-temporal trajectory similarity learning in road networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 347–356.

[14] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. 2021. A graph-based approach for trajectory similarity computation in spatial networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 556–564.

[15] Danlei Hu, Lu Chen, Hanxi Fang, Ziquan Fang, Tianyi Li, and Yunjun Gao. 2024. Spatio-temporal trajectory similarity measures: A comprehensive survey and quantitative study. *IEEE Transactions on Knowledge and Data Engineering* 36, 5 (2024), 2191–2212.

[16] Kaggle. 2015. *Porto Dataset – ECML/PKDD 15: Taxi Trajectory Prediction (I).* Retrieved October 22, 2024 from https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data

[17] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. 2019. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*. IEEE, 449–456.

[18] Satoshi Koide, Chuan Xiao, and Yoshiharu Ishikawa. 2020. Fast subtrajectory similarity search in road networks under weighted edit distance constraints. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2188–2201.

[19] Nikolaos Kouiroukidis and Georgios Evangelidis. 2011. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*. IEEE, 41–45.

[20] Hai Lan, Jiong Xie, Zhifeng Bao, Feifei Li, Wei Tian, Fang Wang, Sheng Wang, and Ailin Zhang. 2022. Vre: a versatile, robust, and economical trajectory data system. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3398–3410.

[21] Shengqiao Li. 2010. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics & Statistics* 4, 1 (2010), 66–70.

[22] Marine Cadastre. 2021. *AIS Data.* Retrieved October 22, 2024 from https://marinecadastre.gov/accessais/

[23] Tao Mei, Yong Rui, Shipeng Li, and Qi Tian. 2014. Multimedia search reranking: A literature survey. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 1–38.

[24] Fanrong Meng, Guan Yuan, Shaoqian Lv, Zhixiao Wang, and Shixiong Xia. 2019. An overview on trajectory outlier detection. *Artificial Intelligence Review* 52 (2019), 2437–2456.

[25] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[26] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed in-memory trajectory analytics. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. 725–740.

[27] Abdel Aziz Taha and Allan Hanbury. 2015. An efficient algorithm for calculating the exact Hausdorff distance. *IEEE transactions on pattern analysis and machine intelligence* 37, 11 (2015), 2153–2163.

[28] Bo Tang, Man Lung Yiu, Kyriakos Mouratidis, and Kai Wang. 2017. Efficient motif discovery in spatial trajectories using discrete fréchet distance. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)*. 378–389.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[30] Michel Verleysen and Damien François. 2005. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*. Springer, 758–770.

[31] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? estimating travel time based on deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[32] Sheng Wang, Zhifeng Bao, J Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast large-scale trajectory clustering. *Proceedings of the VLDB Endowment* 13, 1 (2019), 29–42.

[33] Wikipedia. 2024. *Gamma Function.* Retrieved October 22, 2024 from https://en.wikipedia.org/wiki/Gamma_function

[34] Wikipedia. 2024. *Hypercube.* Retrieved October 22, 2024 from https://en.wikipedia.org/wiki/Hypercube#Faces

[35] Wikipedia. 2024. *Inversion (discrete mathematics).* Retrieved October 22, 2024 from https://en.wikipedia.org/wiki/Inversion_(discrete_mathematics)

[36] Wikipedia. 2024. *Rank correlation.* Retrieved October 22, 2024 from https://en.wikipedia.org/wiki/Rank_correlation

[37] Dong Xie, Feifei Li, and Jeff M Phillips. 2017. Distributed trajectory similarity search. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1478–1489.

[38] Chuang Yang, Zhiwen Zhang, Zipei Fan, Renhe Jiang, Quanjun Chen, Xuan Song, and Ryosuke Shibasaki. 2022. EpiMob: Interactive visual analytics of citywide human mobility restrictions for epidemic control. *IEEE Transactions on Visualization and Computer Graphics* 29, 8 (2022), 3586–3601.

[39] Peilun Yang, Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, and Wenjie Zhang. 2022. TMN: trajectory matching networks for predicting similarity. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1700–1713.

[40] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3s: Effective representation learning for trajectory similarity computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2183–2188.

[41] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 1358–1369.

[42] Di Yao, Haonan Hu, Lun Du, Gao Cong, Shi Han, and Jingping Bi. 2022. Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2275–2285.

[43] Haitao Yuan and Guoliang Li. 2019. Distributed in-memory trajectory similarity search and join on road network. In *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 1262–1273.

[44] Jing Yuan, Yu Zheng, and Xing Xie. 2012. 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*. 186–194.

[45] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*. 99–108.

[46] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. 2020. Trajectory similarity learning with auxiliary supervision and optimal matching. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*. 11–17.

[47] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 1–41.

[48] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[49] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. 2012. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 5, 5 (2012), 363–387.