GPE: Global Position Embedding for Trajectory Similarity Computation

Yuanjun Liu
School of Computer Science and Technology
Soochow University
Suzhou, Jiangsu, China
yjliu1@stu.suda.edu.cn

Guanfeng Liu
School of Computing
Macquarie University
Sydney, New South Wales, Australia
guanfeng.liu@mq.edu.au

Qingzhi Ma
School of Computer Science and Technology
Soochow University
Suzhou, Jiangsu, China
qzma@suda.edu.cn

Zhixu Li
School of Information
Renmin University of China
Beijing, China
zhixuli@ruc.edu.cn

Lei Zhao
School of Computer Science and Technology
Soochow University
Suzhou, Jiangsu, China
zhaol@suda.edu.cn

An Liu*
School of Computer Science and Technology
Soochow University
Suzhou, Jiangsu, China
anliu@suda.edu.cn

Abstract

Trajectory similarity computation is a fundamental functionality in trajectory data mining, with wide-ranging applications in location-based services. Position embedding, which transforms GPS points into embedding vectors, plays a critical role in learning-based trajectory similarity models. The quality of these embeddings significantly impacts the performance of the models on downstream tasks. Existing methods fail to satisfy all good properties, i.e., global, continuous, unique, and dynamic, thereby limiting the development of trajectory similarity computation in both local and global scenarios. Inspired by linear counting systems, such as the decimal system, we first propose the λ-base circular system to embed positions on the circle, then introduce the multi-base global embedding method GPE to encode global positions into vectors. Experiments conducted on five real-world datasets with nine baseline methods demonstrate that the GPE achieves state-of-the-art performance across four key evaluations in downstream tasks.

CCS Concepts

• **Information systems → Location based services**; **Data mining**; *Global positioning systems.*

Keywords

Spatial Position Embedding, Trajectory Similarity Computation, Trajectory Data Mining

ACM Reference Format:
Yuanjun Liu, Guanfeng Liu, Qingzhi Ma, Zhixu Li, Lei Zhao, and An Liu. 2025. GPE: Global Position Embedding for Trajectory Similarity Computation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3736986

KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15481072.

1 Introduction

With the widespread availability of GPS devices and increasing use of location-based services [5, 19, 30], massive volumes of trajectory data are being collected, typically represented as sequences of discrete positions on Earth's surface. Quantifying the similarity between trajectories is a fundamental research problem for many data analysis tasks, including trajectory clustering, classification, and k-nearest neighbor (k-nn) search. These tasks have broad real-world applications, such as traffic management [3, 21, 26] for urban computing, mobile pattern mining [17, 31] for analyzing the track of animals and tracking epidemics during COVID-19, privacy protecting [12, 48] for data publishing, and anomaly detection [37, 47] for management. A number of methods have been proposed to compute trajectory similarity, including both rule-based classical approaches [1, 39, 55], and deep learning approaches, which can be further categorized based on the position embedding techniques, into grid-based [10, 11, 13, 16, 25, 29, 52, 53], road network based [14, 23, 57, 58], POI-based [20, 49, 59], predefined rules based [18, 32, 38, 41, 42, 50], Grid + predefined rules based [4, 51], and learned rules based [28] methods.

Position embedding methods, which transform GPS points into fixed-length vectors, are essential for deep learning-based trajectory similarity computation models, since neural networks rely on embedding vectors as input, as shown in Figure 1. A good position embedding method should possess four key characteristics: global, continuous, unique, and dynamic. However, existing position embedding methods fail to satisfy all these characteristics simultaneously, as shown in Table 1.

Global indicates that the embedding exists for any position on the Earth. However, the grid and road network based methods are

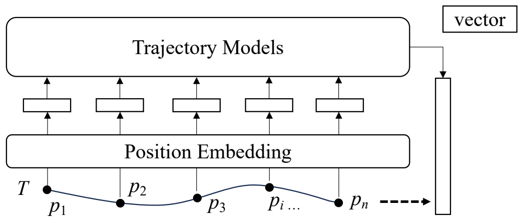

Figure 1: Position Embedding Connects Trajectory Data and Trajectory Models

Table 1: Categories of Position Embedding Techniques

Techniques	Global	Continuous	Unique	Dynamic
Grid	×	×	×	✓
Road Network	×	×	×	✓
POI	✓	×	✓	✓
Predefined Rules	✓	✓	✓	×
Grid + Predef. Rules	×	×	✓	✓
Learned rules	✓	✓	×	✓
GPE (ours)	✓	✓	✓	✓

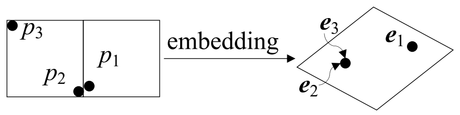

Figure 2: The Illustration of Drawback of Grid-based Methods

limited to specific regions, making them unable to generate valid embeddings for all GPS points.

Continuous requires that similar positions have similar embeddings, and **unique** ensures that different positions have different embeddings. For instance, given three different positions p_1, p_2, and p_3 with relation $\text{dis}(p_1, p_2) < \text{dis}(p_2, p_3)$ shown in Figure 2, their ideal embeddings $\boldsymbol{e}_1, \boldsymbol{e}_2$, and $\boldsymbol{e}_3$ should satisfy $\boldsymbol{e}_1 \neq \boldsymbol{e}_2 \neq \boldsymbol{e}_3$, and $\text{dis}(\boldsymbol{e}_1, \boldsymbol{e}_2) < \text{dis}(\boldsymbol{e}_2, \boldsymbol{e}_3)$ or $\text{sim}(\boldsymbol{e}_1, \boldsymbol{e}_2) > \text{sim}(\boldsymbol{e}_2, \boldsymbol{e}_3)$. However, grid-based and road network based methods are not continuous and unique. For instance, in grid-based methods, all positions in the same grid share the same embedding, violating the unique property, and the embeddings at the border change abruptly, making them discontinuous. After grid embedding, the $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ become identical, since p_2 and p_3 are in the same grid, and $\boldsymbol{e}_2$ and $\boldsymbol{e}_1$ are far away, which is contradicting their spatial relations. Moreover, the POIs are discrete thus their embeddings are not continuous, and the learn rules based methods can not guarantee the uniqueness.

Dynamic caters to various dimensions of the neural network models. For example, the TrajGAT [53] demands the embeddings of 32 dimensions, the TMN [50] expects the 128-dimensional embeddings, and t2vec [25] requires the embeddings of 256 dimensions. Therefore, a good position embedding method is expected to yield dynamic embeddings. However, the predefined rules based methods [18, 32, 38, 41, 42, 50] do not satisfy such property. They compute features as embeddings, such as coordinates, velocities, distances, angles, and accelerated speed. The rules are predefined, and the number of rules is fixed, thus the dimensions are low and can not change, thus those methods are not dynamic.

Apart from the classical local setting that trains and tests on the same dataset, we also analyze the global zero-shot cross-city scenario, that is, training on one dataset and testing on other datasets, which are generally located in different cities. There are lots of applications for the scenario [15, 35, 36, 42]. For example, the map software company wants to update a new model to areas all over the world simultaneously and instantly. Yet, there are two challenges. First, training models for different cities one by one is time-consuming. Second, there are lots of areas that do not have a high quantity of trajectory data to train a model, such as villages and mountainous areas, so it's unable to train a model and provide service there. However, if the company *trains a good model on one good dataset and then deploys it to other cities directly while maintaining good performance*, there will be no above problems, which demands the adaptability of the position embedding methods.

Existing methods lack adaptability in the zero-shot cross-city scenario. On the one hand, the grid-based and road network based methods depend on the cities' area and road network, to transform trajectory into the sequence of grids or edges. However, above information varies significantly across cities, so they cannot be directly applied in cross-city scenarios. Additionally, road network based methods fail to function in areas without road network data, such as marine regions or locations where data privacy restrictions apply. On the other hand, the rules-based methods are sensitive to the geographic coordinates. Specifically, the coordinate ranges differ substantially between cities, while the weights of neural networks are delicate. As a result, models trained on one city often perform poorly when tested on other cities.

Our contributions. In this paper, we propose the GPE to generate high-quality embeddings to meet the aforementioned criteria, i.e., global, continuous, unique, and dynamic. Inspired by the decimal counting system, i.e., the 10-base linear system, we first propose the λ-base circular system Φ^λ to encode a position on the circle precisely index by index. Then, we propose the λ-base global system GPE$^\lambda$ which produces the embedding of positions on the globe. Subsequently, the multi-base global system GPE$^\infty$, i.e., the GPE is introduced by taking advantage of multi bases, to encode any position on the globe. Finally, We evaluate our method against nine baselines on four evaluations in trajectory computing downstream tasks using five real-world datasets. Experimental results demonstrate that our methods outperform existing approaches and achieve superior overall performance in both local and global scenarios adaptively. The contributions are summarize as following.

- We propose the λ-base circular system Φ^λ to encode positions on the circle, and develop the multi-base global system GPE to encode positions on the globe.
- GPE meets four key characteristics of global, continuous, unique, and dynamic, which guarantees its high-quality, and GPE also satisfies the properties of equal similarity and equal distance, which benefits the trajectory computation.
- We evaluate our methods against nine baselines on four evaluations in trajectory computing using five real-world datasets. Experimental results demonstrate that our methods outperform existing approaches and achieve superior overall performance in both local and global scenarios.

The rest of thw paper is organized as follows. Related work is discussed in Section 2. The definitions and problem statement are given in Section 3. Section 4 details the method, and Section 5 conducts the experiments. Finally, we conclude our work in Section 6.

2 Related Work

2.1 Trajectory Similarity Computation

Classical trajectory similarity measures rely on predefined matching rules to assess similarity. DTW [1] warps trajectories in a nonlinear way and matches points to their nearest point, then measures the distances between paired points. LCSS [46], EDR [7], and ERP [6] compute the similarity based on the string distance. LCSS [46] and EDR [7] consider two points as equivalent if the distance between them is below a threshold, then apply string similarity methods to compute trajectory similarity. ERP [6] enhances the EDR by introducing the penalty distance. Fréchet and Hausdorff distances [43] treat trajectories as curves and sets of points, then compute the distance between curves and sets respectively. MD [22] identifies the shortest trajectory merged by two trajectories, and OWD [27] transforms trajectories into lists of grids and then calculates the distance of grids. EDwP [39] and BDS [44] compute the average distance of matched segments as the distance of trajectories. STED [34], STLC [40], and STS [24] incorporate time into similarity calculations. ITS [55] regularizes the distribution of points and then calculates the distance of trajectories efficiently and robustly.

These classical similarity measures generally have $O(n^2)$ time complexity, where n is the length of trajectory, i.e., the number of points. On the one hand, it is time-consuming when n is large. On the other hand, classical measures lack the ability to incorporate learnable semantic information for improvement.

Deep learning based approaches use neural networks, such as RNNs, GNNs, and Transformer, to convert trajectories into representation vectors, and then compute the similarity of vectors as the similarity of trajectories. t2vec [25], At2vec [29], NeuTraj [52], T3S [51], TMN [50], and RSTS [9] simply utilize the RNNs to represent trajectories. GTS [57], KGTS [10], GRLSTM [58], TrajGAT [53], and ST2Vec [14] first build graphs based on road network or grids, then obtain the embedding of each node or grid based on GNNs, followed by RNNs for trajectory representation. START [23] and JGRM [32] replace RNNs with Transformer after processing GAT on the road network. Aries [16] combines LSTM and Transformer to model the trajectories, while MMTEC [28] leverages the CDE network for trajectory representation.

2.2 Position Embedding in Trajectory Computation

The existing position embedding methods can be categorized according to how they process the position information.

In grid-based methods, given a rectangular area that bounds all positions, the space is divided into grids. All locations within a grid share the same embedding. t2vec [25], At2vec [29], Aries [16], MSD-OATD [47], and E2DTC [13] use the word2vec to pre-train grid embeddings, while KGTS [10] applies the RotatE for pre-training. NeuTraj [52] does not pre-train, but lets the embeddings evolve during training. TrajGAT [53] constructs the hierarchical grid graph for its GAT network, then samples paths on the graph to pre-train the embeddings by word2vec. Instead of allocating each grid an independent embedding, Traj2Hash [11] presents each grid with its coordinate and induces the embedding of the grid from the combination of the embeddings of the coordinate.

While grid-based methods perform well in downstream tasks, they have several limitations. First, storing embeddings for each grid can be space-intensive, and pre-training is time-consuming. Second, they are not continuous and unique, i.e., all positions in the same grid share the same embedding, and the embeddings of positions near the boundaries change suddenly, which leads to errors for margin positions. Third, the spaces of various cities are diverse, they need to train on the new datasets to compute on new cities, which is not adaptive for global cross-city scenario.

Road network based methods employ the map-matching technique to transform trajectories from the sequence of positions to the sequence of road vertices or edges. GTS [57] and ST2Vec [14] employ word2vec to generate vertex embeddings. GRLSTM [58] uses TransH to pre-train the embeddings of vertexes and edges. START [23] derives edge embeddings from features of road network structure and trajectory mobility.

The drawbacks of the above methods contribute to the dependency of road networks. First, obtaining high-precision map data is challenging, since data privacy and security have raised more concerns recently. Additionally, road networks are not available in all settings, such as trajectories of ships at sea or animal movement in remote areas. Second, they are not adaptive in global scenario, since the road networks of different cities are diverse. Moreover, they are not continuous and unique, similar to the grid-based methods.

POI-based methods [20, 49, 59] yield embeddings for positions assigned as the POI points. The POIs are unique and can be located anywhere on Earth, so the embeddings are unique and global. However, the POIs are discrete so the embeddings are not continuous.

Rule-based methods encode positions using predefined, such as coordinates, velocities, and distances, or using learned rules. TMN [50], AdapTraj [38], and TUTR [41] input the raw coordinates directly for similarity computation or trajectory prediction. FlightBERT++ [18] includes both coordinates and velocities as features, while MetaTraj [42] utilizes the offset distances of two dimensions, and combines them with the coordinates. JGRM [32] describes the positions with kinematic attributes, including longitude, latitude, speed, acceleration, angle delta, time delta, and distance. MMTEC [28] encodes coordinates by trigonometric functions with learnable parameters, where the parameters conduct linear transformation and trigonometric functions allow nonlinear transformation.

For the the predefined rules based methods, embedding dimensions are limited, as the number of rules is small and fixed, so they may not be ideal for modern deep-learning models. Additionally, designing appropriate rules requires considerable effort and domain knowledge. And the learned rules can not guarantee their qualities. Moreover, different cities have different ranges and shapes, which hinders the adaptability of the zero-shot cross-city scenario.

Several methods combine the grid with the predefined rules. T3S [51] incorporates both grid embeddings and raw coordinates, while TrajCL [4] merges grid embedding with XYRL features, where XY is longitude and latitude, R is the angle of two adjacent segments, L is the mean length of two adjacent segments, and a segment is a line between two adjacent positions. The grids and rules bring the properties of dynamic and unique, yet the drawbacks of being not global, continuous, and adaptive are inherited.

3 Preliminaries

Definition 1 (λ-base Linear System Ψ^λ). *Given a number $x \in [0, 1]$, a base $\lambda \geq 1$, and a number of indexes $i \geq 1$, the λ-base linear system Ψ^λ encodes x as*

$$\Psi^\lambda(x) = [\psi_1^\lambda(x), \psi_2^\lambda(x), \ldots, \psi_i^\lambda(x)], \tag{1}$$

where the i-th item ψ_i^λ is calculated as

$$\psi_i^\lambda(x) = \lfloor \lambda^i x \mod \lambda \rfloor. \tag{2}$$

Definition 2 (Circular Position). *A circular position is a real number $x \in [-\pi, \pi]$, where $[-\pi, \pi]$ is the circular space C with field 2π.*

Definition 3 (Global Position). *A global position, known as GPS point, is a tuple of longitude and latitude, i.e., $p = (lon, lat)$, $lon \in [-\pi, \pi] \subseteq C$, $lat \in [-\pi/2, \pi/2] \subseteq C$, where $[-\pi, \pi] \times [-\pi/2, \pi/2]$ is the global space $\mathcal{S}$.*

Definition 4 (Trajectory). *A trajectory T collected on the surface of the earth is defined as the sequence of global positions, i.e., $T = [p_1, p_2, \ldots, p_n]$, $p_i \in \mathcal{S}$.*

Definition 5 (Position Embedding). *Given a global position $p \in \mathcal{S}$, the embedding method $\mathcal{F}$ generates a dense vector of dimension $h > 0$, i.e., $\mathcal{F}^h(p) \in \mathbb{R}^h$.*

Problem Statement. Our objective is to construct a position embedding function $\mathcal{F}$ that maps a global position $p \in \mathcal{S}$ to a h-dimensional embedding vector $\boldsymbol{e} \in \mathbb{R}^h$, such that each trajectory can be transformed into a sequence of h-dimensional vectors, allowing a sequence model $\boldsymbol{m}$ to compute the representation vector of the trajectory and the similarity of trajectories.

A good embedding method should satisfy the following characteristics, where dis represents the Euclidean distance.

Definition 6 (Global). *$\mathcal{F}$ is global if any position has valid embedding, i.e., $\forall p \in \mathcal{S}, \mathcal{F}(p) \neq \varnothing$.*

Definition 7 (Continuous). *$\mathcal{F}$ is continuous if similar positions have similar embeddings, i.e., $\lim_{p' \to p} dis(\mathcal{F}(p'), \mathcal{F}(p)) = 0$.*

Definition 8 (Unique). *$\mathcal{F}$ is unique if distinct positions have distinct embeddings, i.e., $\forall p \neq p', \mathcal{F}(p') \neq \mathcal{F}(p)$.*

Definition 9 (Dynamic). *$\mathcal{F}$ is dynamic if it can generate valid embedding for any required dimension, i.e., $\forall h > 0, \mathcal{F}^h(p) \neq \varnothing$.*

4 GPE: Global Position Embedding

In this section, we first propose the circular counting system to represent a position in the circular space in Section 4.1. Then we introduce the GPE in Section 4.2, which generates the embedding of positions in the global space.

4.1 From Linear System to Circular System

To describe a position more precisely, the linear system used in Euclidean space narrows down the field one index by one index. For instance, the widely-used 10-base linear system splits the whole field into 10 equal parts, labels the sub-field with increasing numbers, and repeats this process.

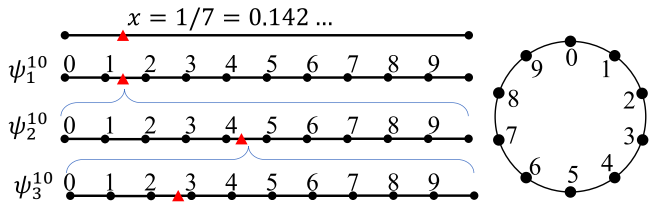

Figure 3: An Example of the 10-base Linear System

Figure 3 left illustrates that encoding the number $x = 1/7$ by the 10-base linear system. The system splits the whole field into 10 parts, finds the part containing x, and records $\psi_1^{10}(x) = 1$, where 10 is the base and 1 is the index and the recorded value. Then, it records the index 2 with value 4, i.e., $\psi_2^{10}(x) = 4$, so x is eventually described as '0.142857...'. During the process, the field of each index narrows down by 10 times, so the description is more and more precise. Specifically, the i-th item ψ_i^{10} has the field of 10^{-i}.

However, the linear system is not suitable for the circular space, which is enclosed, so the distance between positions differs from that in the linear space. For example, Figure 3 right shows that in a circular space, the distance between '1' and '9' should be 2, instead of 8 in the linear system.

To address this, we propose the circular system to encode circular positions. First, since the circular space is enclosed, we use the sine and cosine functions to encode positions simultaneously. On the one hand, the sine and cosine functions have a domain that covers the entire circular space, and their combination uniquely represents each position on the circle. For example, in Figure 3, while the sine values of '1' and '9' are the same, their cosine values differ, so their representations differ. On the other hand, the value of sine and cosine is enclosed, which helps to distinguish the correct distance on the circle. For instance, the sine and cosine values of the angle between '1' and '9' are the same as that between '1' and '3', indicating that the distance is computed correctly.

Then, we adapt the basic methodology of the λ-base system, that is, narrowing down the field by λ times each index, which captures more and more precise descriptions. The λ-base circular system Φ^λ is defined as follows:

Definition 10 (λ-base Circular System Φ^λ). *Given a circular position $x \in [-\pi, \pi]$, a base $\lambda \geq 1$, and a number of indexes $i \geq 0$, the λ-base circular system Φ^λ encodes x as*

$$\Phi^\lambda(x) = [\phi_0^\lambda(x), \phi_1^\lambda(x), \phi_2^\lambda(x), \ldots, \phi_i^\lambda(x)], \tag{3}$$

where the i-th item ϕ_i^λ is calculated as

$$\phi_i^\lambda(x) = [\sin(\lambda^i x), \cos(\lambda^i x)]. \tag{4}$$

Specifically, $\phi_0^\lambda = [\sin(x), \cos(x)]$ when $i = 0$ and $\Phi^1(x) = \phi_0^1(x) = [\sin(x), \cos(x)]$ when $\lambda = 1$.

Compared to the Equation (2), on the one hand, the Equation (4) does not invoke the mod operation since the sine and cosine have the functionality of mod. On the other hand, the Equation (4) does not invoke the $\lfloor \cdot \rfloor$ operation, since we aim to provide a precise description but $\lfloor \cdot \rfloor$ roughs the description. Therefore, the λ-base

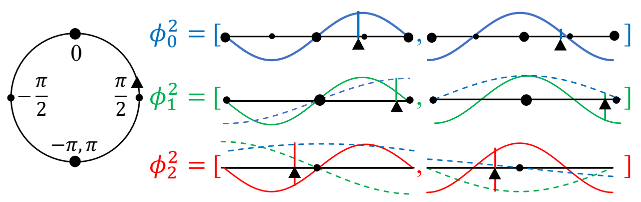

Figure 4: An Example of the 2-base Circular System

Table 2: Comparison of Linear System and Circular System

Index	Ψ^λ		Φ^λ	
	Item	Field	Item	Field
0	0	1	$\phi_0^\lambda(x)$	2π
1	$\psi_1^\lambda(x)$	λ^{-1}	$\phi_1^\lambda(x)$	$2\pi\lambda^{-1}$
2	$\psi_2^\lambda(x)$	λ^{-2}	$\phi_2^\lambda(x)$	$2\pi\lambda^{-2}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
i	$\psi_i^\lambda(x)$	λ^{-i}	$\phi_i^\lambda(x)$	$2\pi\lambda^{-i}$

linear system describes a number as the sequence of integers, i.e., $\Psi^\lambda(x) \in \{0, 1, \cdots, \lambda - 1\}^h, x \in \mathbb{R}$, but the λ-base circular system describes a circular position as the sequence of real numbers, i.e. $\Phi^\lambda(x) \in [-1, 1]^h, x \in C$.

Figure 4 illustrates the example of the 2-base circular system with 3 items, the x marked in the red triangle is slightly less than $\frac{\pi}{2}$. The first item ϕ_0^2 computes the sine and cosine values of x. Then, the ϕ_1^2 enlarges x by 2 times, narrowing down its field by 2 times simultaneously. The dotted lines in the picture of ϕ_1^2 are the enlarged pictures of ϕ_0^2, so it's clear that the field of ϕ_1^2 is half of ϕ_0^2. Finally, the ϕ_2^2 computes the sine and cosine values of 2^2x with the field of $\frac{\pi}{2^2}$. In its picture, the black dotted lines are the pictures of ϕ_1^2, and the gray dotted lines are the pictures of ϕ_0^2.

Table 2 compares the λ-base linear system Ψ^λ and λ-base circular system Φ^λ. First, the index of Φ^λ starts from 0 rather than 1, and the ϕ_0^λ has the field of whole 2π and thus can capture global information. Second, the i-th item ϕ_i^λ has the field of $2\pi\lambda^{-i}$ instead of λ^{-i}, since the whole field of the circular space is 2π. Third, each item ϕ_i^λ has two numbers actually, i.e., the sine and cosine values, thus given the total number of indexes i, Φ^λ yields a vector sized of $2i$.

The λ-base circular system Φ^λ satisfies good properties to generate valid and unique embedding and has the ability to distinguish the pattern of movement in trajectories.

Theorem 1. *Φ^λ is* ***continuous****.*

Proof. Each component of Φ^λ, i.e, the $\sin(\lambda^i x)$ or $\cos(\lambda^i x)$, is the combination of elementary functions, so each component is continuous, and Φ^λ is continuous. □

The ϕ has the domain of $(-\infty, \infty)$ mathematically, which covers the circular space, i.e., $C = [-\pi, \pi]$. Therefore, for any position on the circle, Φ^λ can yield a valid embedding, i.e., $\forall x \in C, \Phi(x) \neq \varnothing$.

Theorem 2 (Bijection). *Let $\mathbb{E} = \{\Phi^\lambda(x) | x \in C\}$, $\forall x_1, x_2 \in C$, $x_1 \neq x_2$, $\Phi^\lambda(x_1) \neq \Phi^\lambda(x_2)$.*

Proof. Consider the first items $\phi_0^\lambda = [\sin(x), \cos(x)]$. Given $x_1 \neq x_2$, we have $\sin(x_1) \neq \sin(x_2)$ or $\cos(x_1) \neq \cos(x_2)$. Thus $\Phi^\lambda(x_1) \neq \Phi^\lambda(x_2)$, Φ^λ is an one-to-one function. □

This property guarantees that for all position x on the circle, the λ-base circular system Φ^λ can yield a **unique** embedding. The Φ^λ is also **dynamic**. Given the length h, the Φ^λ can generate h numbers by setting the maximal index as $h/2$.

The adaptability of Φ^λ attributes to two reasons. First, the value of Φ^λ is transformed into the range of [-1,1] thanks to the sine and cosine functions, so the embeddings for any datasets and cities have the same value range, which helps the following models to compute adaptively. Second, the following properties indicate that Φ^λ is good at detecting the movement, which benefits computing trajectory similarity.

Theorem 3 (Equal Distance of ϕ). *Given y, $\forall x_1, x_2$, $dis(\phi_i^\lambda(x_1), \phi_i^\lambda(x_1 + y)) = dis(\phi_i^\lambda(x_2), \phi_i^\lambda(x_2 + y))$, where dis is the Euclidean distance.*

Proof. Let

$$\begin{aligned} f(x, y) &= (\sin(x + y) - \sin(x))^2 + (\cos(x + y) - \cos(x))^2 \\ &= -2\sin(x + y)\sin(x) - 2\cos(x + y)\cos(x) + 2 \\ &= 2 - 2\cos(y), \end{aligned}$$

$\text{dis}(\phi_i^\lambda(x), \phi_i^\lambda(x + y)) = \sqrt{f(\lambda^i x, \lambda^i y)} = \sqrt{2 - 2\cos(\lambda^i y)}$. Thus, $\text{dis}(\phi_i^\lambda(x_1), \phi_i^\lambda(x_1 + y)) = \text{dis}(\phi_i^\lambda(x_2), \phi_i^\lambda(x_2 + y))$. □

Theorem 4 (Equal Similarity of ϕ). *Given y, $\forall x_1, x_2$, $sim(\phi_i^\lambda(x_1), \phi_i^\lambda(x_1 + y)) = sim(\phi_i^\lambda(x_2), \phi_i^\lambda(x_2 + y))$, where sim is the cosine similarity.*

Proof. $\text{sim}(\phi_i^\lambda(x), \phi_i^\lambda(x + y)) =$

$$\frac{\sin(\lambda^i x)\sin(\lambda^i x + y) + \cos(\lambda^i x)\cos(\lambda^i x + y)}{\sqrt{\sin^2(\lambda^i x) + \cos^2(\lambda^i x)}\sqrt{\sin^2(\lambda^i x + y) + \cos^2(\lambda^i x + y)}}$$

$= cos(y)$,

Thus, $\text{sim}(\phi_i^\lambda(x_1), \phi_i^\lambda(x_1 + y)) = \text{sim}(\phi_i^\lambda(x_2), \phi_i^\lambda(x_2 + y))$. □

Since each item ϕ satisfies equal distance and equal similarity, the Φ^λ also satisfies the equal distance and equal similarity, i.e., $y, \forall x_1, x_2 \in C$, $\text{dis}(e_1, e_1') = \text{dis}(e_2, e_2')$ and $\text{sim}(e_1, e_1') = \text{sim}(e_2, e_2')$, where $e_1 = \Phi^\lambda(x_1), e_1' = \Phi^\lambda(x_1 + y), e_2 = \Phi^\lambda(x_2), e_2' = \Phi^\lambda(x_2 + y)$.

The above properties help the computation of trajectory data. Specifically, two objects move equal distance y from disparate positions x_1 and x_2. The embeddings of positions are different, but Φ^λ can distinguish that the changed distance and similarity are equal.

4.2 GPE: Multi-base Global System

Based on the λ-base circular system Φ^λ, we first construct the λ-base global system GPE$^\lambda$ to encode global positions.

Definition 11 (λ-base Global System GPE$^\lambda$). *Given a base $\lambda \geq 1$ and a global position $p = (lon, lat) \in \mathcal{S}$,*

$$GPE^\lambda(p) = [\Phi^\lambda(lon), \Phi^\lambda(lat)]. \quad (5)$$

Given the total number of index i, it generates a vector sized of $4i$ for any position $p \in \mathcal{S}$, which is **global** and dynamic. Since GPE$^\lambda$ is the concatenation of Φ^λ, it inherits the properties such as continuous, unique, equal distance, and equal similarity.

Although GPE$^\lambda$ can generate valid embeddings, the single base system is not expandable. First, the number of indexes of the single base system has limitations in real applications. For example, ϕ^2_{25}, i.e., $\lambda = 2$ and $i = 25$, and ϕ^3_{16}, i.e., $\lambda = 3$ and $i = 16$, achieve a precision of about 1 meter, while the GPS has the noise of about 15 meters [45], thus using larger i for a single base λ is impractical. Second, the fields of the single-base system are fixed, while the multi-base system provides multiple perspectives, allowing the following models to extract more information.

Therefore, we introduce the multi-base global system GPE$^\infty$, denoted as GPE, which leverages the advantages of the multi-base system and can choose bases automatically.

Definition 12 (Multi-base Global System GPE$^\infty$, GPE). *Given a global position $p = (lon, lat) \in \mathcal{S}$,*

$$GPE^\infty(p) = [\Phi^\infty(lon), \Phi^\infty(lat)], \tag{6}$$

where Φ^∞ is the multi-base circular system, i.e.,

$$\begin{aligned}\Phi^\infty &= \Phi^1_\epsilon \cup \Phi^2_\epsilon \cup \Phi^3_\epsilon \cup \Phi^5_\epsilon \cup \cdots \cup \Phi^{\Gamma_\tau}_\epsilon \\ &= [\phi^1_0, \phi^2_1, \ldots, \phi^2_{I^\epsilon_2}, \phi^3_1, \ldots, \phi^3_{I^\epsilon_3}, \ldots, \phi^{\Gamma_\tau}_{I^\epsilon_{\Gamma_\tau}}, \ldots].\end{aligned} \tag{7}$$

The Γ_τ and I^ϵ_λ are two strategies to choose the next base λ and the maximal index i of current base respectively. Note that in Φ^∞, the first item of λ-base system $\phi^\lambda_0, \lambda > 1$ disappeared after the $\cup$ operation, since $\phi^\lambda_0 = \phi^1_0$. Therefore, $\Phi^{\Gamma_\tau}_\epsilon \in \Phi^\infty$, yet $\sum |\Phi^{\Gamma_\tau}_\epsilon| \leq |\Phi^\infty|$.

There are two key issues should be solved from the single-base system to the multi-base system. The first is how to determine the number of items of each single-base system, and the second is how to choose the next base.

We employ the ϵ-cut strategy to determine the maximal number of items of each single base system Φ^λ. The threshold value ϵ can be viewed as the minimal field of the final item ϕ^λ_i. If the field of ϕ^λ_i is less than ϵ, we shall stop encoding based on the current system Φ^λ, and turn to the next system.

Definition 13 (ϵ-cut Strategy). *Given a threshold $\epsilon \in (0, 2\pi)$ and a base $\lambda \geq 1$, the maximal index i of ϕ^λ_i is*

$$I^\epsilon_\lambda = \begin{cases} 0, & \lambda = 1, \\ \lceil -\log_\lambda \frac{\epsilon}{2\pi} \rceil, & \lambda > 1. \end{cases} \tag{8}$$

After processing the current base λ, the straightforward choice for the next base is $\lambda + 1$, which may yield redundant items. For example, $\Phi^4 \subset \Phi^2$, since $\phi^4_i(x) = [\sin(4^i x), \cos(4^i x)] = [\sin(2^{2i}x), \cos(2^{2i}x)] = \phi^2_{2i}(x)$. All items in Φ^4 appear in Φ^2, thus the base 4 should be skipped. The τ-filter strategy skips the exponents of historical bases, filtering out the redundant items in the multi-base system. Let τ be the set of historical bases, i.e., $\tau = \{1, 2, 3, 5, \ldots, \Gamma_\tau\}$, the Γ_τ is described as following.

Definition 14 (τ-filter Strategy). *Given the historical base set τ, the next base λ can not be*

$$\lambda \leq \max \tau, \exists b \in \tau, b > 1, \log_b \lambda \in \mathbb{N}. \tag{9}$$

Γ_τ selects the smallest λ that does not satisfy Equation (9). However, Equation (9) requires quadratic time complexity to find $|\tau|$ bases, thus we apply the exponential sieve method to find all bases in $O(h \log h)$ time. We initialize an array $\boldsymbol{\Lambda}$ of size h with value 0, and $\boldsymbol{\Lambda}_\lambda$ is 1 means that λ is an exponent of a previous base and should be skipped, i.e., $\exists b \in \tau, \log_b \lambda \in \mathbb{N}$, and vice versa. We search λ from 2 such that $\boldsymbol{\Lambda}_\lambda$ is 0, and update $\boldsymbol{\Lambda}_{\lambda^i}, i \geq 1$ as 1 for each base λ after processing the base λ. The array size should be larger than the maximum element, i.e., $|\boldsymbol{\Lambda}| \geq \max \tau$. We set $|\boldsymbol{\Lambda}|$ as h since $\max \tau < h$.

Algorithm 1: GPE

```
   Input: position $p = (x, y)$, number $\epsilon$, integer $h$
   Output: embedding $\boldsymbol{e}$
1  $\boldsymbol{w} \leftarrow [1]; \boldsymbol{\Lambda}_\lambda \leftarrow 0, 1 \leq \lambda \leq h; \lambda \leftarrow 2$
2  while $|\boldsymbol{w}| \leq h/4$ do
3      while $\boldsymbol{\Lambda}_\lambda$ do
4          $\lambda{+}{+}$
5      end
6      for $i \leftarrow 1$ ; $i \leq \lfloor \log_\lambda h \rfloor$ ; $i{+}{+}$ do
7          $\boldsymbol{\Lambda}_{\lambda^i} \leftarrow 1$
8      end
9      for $i \leftarrow 1$ ; $i \leq \lceil -\log_\lambda \frac{\epsilon}{2\pi} \rceil$ and $|\boldsymbol{w}| \leq h/4$ ; $i{+}{+}$ do
10         $\boldsymbol{w} \leftarrow \boldsymbol{w} \cup [\lambda^i]$
11     end
12 end
13 $\boldsymbol{e} \leftarrow \bigcup_{w \in \boldsymbol{w}} [\sin(wx), \cos(wx)] \cup \bigcup_{w \in \boldsymbol{w}} [\sin(wy), \cos(wy)]$
```

Theorem 5. $\max \tau < h$.

Proof. Without loss of generality, we assumue that $h \geq 4$ and is the multiple of 4. The GPE algorithm needs $h/4$ parameters of trigonometric functions, and each base $\lambda > 1$ yields $I^\epsilon_\lambda \geq 1$ parameters, so $|\tau| \leq h/4$. Let N be $\max \tau$, and the biggest N comes when $\epsilon \longrightarrow 2\pi$. In this worst case, $I^\epsilon_\lambda = 1$ for all $\lambda > 1, |\tau| = h/4$, and the max τ is just after the bases and the skipped experiments, i.e., $N = |\tau| + \sum_{\lambda \in \tau, \lambda > 1} \log_\lambda h + 1$, where $\sum \log_\lambda h$ counts the skipped experiments, and 1 represents the omitted ϕ^1_0 of base 1. Let $f(h) = \frac{3h}{4} - \sum_{\lambda=2}^{h/4} \log_\lambda h - 1, f'(h) = 3/4 - \frac{1}{h}\sum_{\lambda=2}^{h/4} \frac{1}{\ln \lambda}$. Note that $\frac{1}{h}\sum_{\lambda=2}^{h/4} \frac{1}{\ln \lambda} < \frac{1}{h}\frac{h}{4}\frac{1}{\ln 2} < \frac{1}{2}$, so $f'(h) > 0$, f increases, and $f(4) > 0$, so $f > 0$, $\sum_{\lambda=2}^{h/4} \log_\lambda h + 1 < \frac{3h}{4}$. Note that $\log_a h \leq \log_b h, a \geq b > 1$, and $\tau_i \geq i$, $(\tau = \{1, 2, 3, 5, \ldots\})$, so $\sum_{\lambda \in \tau, \lambda > 1} \log_\lambda h < \sum_{\lambda=2}^{h/4} \log_\lambda h + 1$. Recall that $\sum_{\lambda=2}^{h/4} \log_\lambda h + 1 < \frac{3h}{4}$ and $|\tau| \leq h/4$, so $|\tau| + \sum_{\lambda \in \tau, \lambda > 1} \log_\lambda h + 1 < h$, thus $N < h$, $\max \tau < h$. □

The detail of the GPE is formalized in Algorithm 1. Given the global position $p = (x, y)$ to be transformed, where x is the longitude and y is the latitude, the threshold ϵ of the minimal field, and the dimension h, GPE generates a dense vector $\boldsymbol{e}$ of size h. Line 1 computes the parameter of ϕ^1_0, i.e., 1, and initializes the sieve array $\boldsymbol{\Lambda}$. Then, lines 2-12 computes the parameters of ϕ^λ_i with automatically chosen λ and i, until the size of $\boldsymbol{w}$ reaches expected $h/4$. The next λ starts from 2, and lines 3-8 choose the λ by the τ-filter strategy. For current base λ, lines 9-11 calculates at most I^ϵ_λ items, bounded by the ϵ-cut strategy. To ensure that the length of embedding does not exceed h, the process exits early when the size of $\boldsymbol{w}$ reaches $h/4$. Finally, line 13 calculates the embedding based on previously computed parameters $\boldsymbol{w}$.

The algorithm has $O(h \log h)$ time complexity and $O(h)$ space complexity. Lines 3-8 is $O(h \log h)$ time complexity to find all bases, where going through τ needs $O(h)$ time, and updating the Λ needs $O(\log_\lambda h)$ time. Lines 9-11 are $O(h)$ time complexity to compute $h/4$ parameters. Finally, line 13 consumes $O(h)$ time to generate embedding $\boldsymbol{e}$. The algorithm records $\boldsymbol{w}$ and Λ, whose length is $O(h)$, thus the space complexity is $O(h)$.

Note that given the same ϵ and h, the $\boldsymbol{w}$ is the same, thus Algorithm 1 can be transformed into the batch mode to process n position simultaneously, by repeating line 13 for each position. In this case, the batch-GPE has $O(h \log h + nh)$ time complexity. In other words, it's linear time complexity for n when h is fixed. Moreover, compared with other embeddings methods that require pre-training to achieve the dynamic property, the GPE does not need any pre-training process, which is efficient.

Given parameter ϵ, GPE can generate valid vector for any position on the Earth and any demanded size $h > 0$ of embeddings, thus it is global and dynamic. Since the GPE is the concatenation of Φ, GPE is continuous, i.e., $\lim_{p' \to p} \text{dis}(\text{GPE}(p'), \text{GPE}(p)) = 0$, is unique, i.e., $\forall p_1 \neq p_2, \text{GPE}(p_1) \neq \text{GPE}(p_2)$, and is equal distance and equal similarity, i.e., given $\Delta p, \forall p_1, p_2 \in \mathcal{S}, \text{dis}(e_1, e_1') = \text{dis}(e_2, e_2')$ and $\text{sim}(e_1, e_1') = \text{sim}(e_2, e_2')$, where $e_1 = \text{GPE}(p_1), e_1' = \text{GPE}(p_1 + \Delta p), e_2 = \text{GPE}(p_2), e_2' = \text{GPE}(p_2 + \Delta p)$.

5 Experiments

We evaluate our methods with nine related methods on five real-world datasets. Section 5.1 provides details about the base model, datasets, compared methods, and evaluation metrics. Section 5.2 and Section 5.3 display the experimental results. Section 5.4 analyzes the parameters, runing time, and the base selection strategies. Moreover, Section 5.5 shows the case study results.

5.1 Experimental Setup

Our method and the experimental code are implemented in Python and Pytorch. The platform runs the Ubuntu 16.04 operating system with Intel 6248R CPU, 256GB RAM, and an NVIDIA V100 32G GPU.

Base Model. To avoid the distraction of various model architectures and setups, we verify different embedding methods with the same model and the same configurations. The base model is a 2-layer bidirectional LSTM following three dense layers with SiLU and BatchNorm layers. The position embedding changes according to the compared methods, and others remain the same. We concatenate the outputs of both two directions of the final LSTM layer, then input it to the linear layers. The sizes of the embedding and the hidden vectors are both 128. We employ the noise contrastive learning [8] to train the model as following, , i.e., $\mathcal{L}_{CL} = -\log \frac{\exp(\text{sim}(x,x^+)/\varsigma)}{\exp(\text{sim}(x,x^+)/\varsigma) + \sum_{x^- \in \boldsymbol{x}^-} \exp(\text{sim}(x,x^-)/\varsigma)}$, where ς is the temperature, set as 1 by default. For each trajectory x, we generate its positive sample x^+ by adding noise following [25], including dropping the points with the ratio of 0.1 and shifting the position of points within the range of about 20 meters randomly. We regard other raw and noised trajectories in the same batch as the negative samples $\boldsymbol{x}^-$. We train the model by the AdamW optimizer with default configurations, i.e., the learning rate of 0.001 and so on. The batch size is set to 1024. We train at least 2 and at most 20 epochs, and stop training if the loss does not descend for 100 iterations.

Table 3: Dataset Statistics

Dataset	#Trajectories after Processing
T-drive	165,055
Porto	1,577,389
Roma	216,700
AIS	54,589
GeoLife	167,927

Datasets. The experiments are conducted on five real-world trajectory datasets. T-drive [54] is generated by over 33,000 taxis during 3 months, and GeoLife [56] is generated by 182 users in over three years. Note that both the two datasets are collected in Beijing, China, so the positions and the shapes of their bounding box are the same. Porto [33] is generated by 442 taxis during 1 year in Porto, Portugal, and Roma [2] is collected by approximately 320 taxis in over 30 days in Rome, Italy. AIS [1] is a collection of marine vessel trajectories of the first 3 months of 2024.

We split the trajectories if the gap of time of two adjacent points exceeds 30 minutes, 1 minute, and 1 hour for T-drive, Roma, and AIS datasets respectively. We cut the trajectory if its length is beyond 200, with 100 overlapping points, and discard trajectories with lengths less than 20. Table 3 shows the statistics of all datasets after pre-processing. We treat the final 5,000 trajectories as the test set, and the regard remaining trajectories as the train set.

Baselines. We compare the proposed GPE method with nine related position embedding methods. **Grid** [10, 13, 16, 25, 52] is widely used by discretizing the space into grids. We set the grid size as 150 meters generally, and set it to 17 sea miles for the AIS dataset. We pre-train the grid by the word2vec technique following [25]. **XY** [38, 41, 50] utilize the raw coordinates, we add a linear layer, which transforms 2 dimension vectors to 128 dimension embeddings, to generate useable embeddings. Besides raw coordinates, **XYVV** [18] computes the velocities, **XYDD** [42] calculates the offset distances, and **XYSARD** [32] employs the speed, acceleration, angle, and distance to describe the positions. We add a linear layer on them to generate 128-dimensional embeddings. **TriW** [28] employs the trigonometric functions with learnable parameters to encode the positions to embeddings. **XYG** [51] combines the grid embeddings with the raw coordinates, and **XYRLG** [4] considers the rotation angle and length features. We employ a linear layer to transform the XY and XYRL features into 64-dimensional vectors, then concatenate them with pre-trained 64-dimensional grid embeddings. **GXGY** [11] induces the embedding of each grid from the combination of the embeddings of its coordinates. We pre-train the embeddings following [11]. Since the predefined rules based methods require neural network to achieve various embedding size, we add linear layers for all compared methods until the total number of linear layers reaches three.

For the global zero-shot cross-city scenario, runing the baselines in unseen cities directly is impractical, since the shape and

[1] https://marinecadastre.gov/ais/

position of cities are various. We employ the space shifting strategy, denoted as SS, to migrate positions from other cities to the training city, to deploy the baselines. Given $p^A = (lon^A, lat^A)$ from other city with bounding box $lon^A \in [lon^A_{min}, lon^A_{max}]$ and $lat^A \in [lat^A_{min}, lat^A_{max}]$, SS conducts a linear transformation to convert it to $p^B = (lon^B, lat^B)$ in the training city with bounding box $lon^B \in [lon^B_{min}, lon^B_{max}]$ and $lat^B \in [lat^B_{min}, lat^B_{max}]$, i.e., $lon^B = (lon^A - lon^A_{min})\frac{lon^B_{max} - lon^B_{min}}{lon^A_{max} - lon^A_{min}} + lon^B_{min}$, and $lat^B = (lat^A - lat^A_{min})\frac{lat^B_{max} - lat^B_{min}}{lat^A_{max} - lat^A_{min}} + lat^B_{min}$. We do not include road network based and POI-baesd methods, since different cities have different road network and POI information, which makes it impossible for them to execute the global zero-shot cross-city scenario.

Evaluations. Inspired by [25], we employ the following metrics to evaluate the performance of methods. For each trajectory $T = [p_1, p_2, \dots]$ in the test dataset D, we split it in into odd index trajectory $T_A = [p_1, p_3, \dots]$ and even index trajectory $T_B = [p_2, p_4, \dots]$, which forms the two datasets D_A and D_B. Let s_{ij} be the similarity of trajectory $T_i \in D_A$ and $T_j \in D_B$, γ_i be the rank of s_{ij} in $(s_{i1}, s_{i2}, \dots, s_{i|D|})$. **Mean Rank** is defined as $MR = \frac{1}{|D|}\sum_{i=1}^{|D|}\gamma_i$, where $MR \geq 1$, and a lower MR means better performance. **Mean Reciprocal Rank** is defined as $MRR = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{\gamma_i}$, where $MRR \in [0,1]$, and a higher MRR means better performance. Let $\mathbb{1}\gamma_i$ be 1 if $\gamma_i = 1$, otherwise is 0, then the **Mean Precision** is defined as $MP = \frac{1}{|D|}\sum_{i=1}^{|D|}\mathbb{1}\gamma_i$, where $MP \in [0,1]$, and a higher MP means better results. Let η_i be the k-nn results of $T_i \in D$ in D, and η'_i be the k-nn results of $T_i \in D_A$ in D_B, the k**-nn Precision** is defined as $KP@k = \frac{1}{|D|}\sum_{i=0}^{|D|}\frac{|\eta'_i \cap \eta_i|}{k}$, where $KP \in [0,1]$, and a higher KP means better performance. We set $k = 10$ by default.

5.2 Local Scenario Results

In this section, we experiment on the classical scenario, that is, training and testing on the same dataset. Limited by space, Table 4 left shows the average results on five datasets, and the detail results of all datasets are displayed at Section A. The **bold font** means the top-1 results, and the underbar font indicates the top-2 results.

The GPE is the top-1 on all evaluations, which indicates that the proposed GPE method have the best performance on four evaluations compared with nine baselines. Specifically, from the results of MR, we can see that the GPE yields almost the ground truth results. The grid-based methods perform better than rules-based methods, indicating the effect of pre-training strategy, yet they are dominated by the GPE which does require any pre-training.

5.3 Global Scenario Results

In this section, we conduct experiments on the zero-shot cross-city scenario, that is, training on one dataset and testing on others. This scenario aims to verify the adaptability of the position embedding methods. Limited by space, Table 4 right shows the averaged MRR results of training on the one dataset and testing on the other four unseen datasets, and the details are displayed at Section A.

GPE outperforms others, since it's the best on all datasets with significant improvements. GPE performances well on unseen datasets, while others degenerate sharply. An important observation is that, training on high-quality data such as T-drive can bring the GPE good performance on unseen datasets, which is close to and even better than the optimal results of the classical scenario. This observation supports the high feasibility of the zero-shot cross-city application, that is, *training a good model on one single city and deploying it to other cities directly while maintaining good performance.*

Table 4: Average Results

	Local				Global (MRR↑)				
	MR↓	MRR↑	MP↑	KP↑	T-drive	Porto	Roma	AIS	GeoLife
Grid	1.587	0.905	0.847	0.548	0.263	0.036	0.039	0.032	0.181
TriW	4.941	0.857	0.797	0.558	0.607	0.437	0.254	0.040	0.346
XY	250.8	0.383	0.318	0.469	0.169	0.148	0.110	0.040	0.069
XYVV	47.13	0.624	0.555	0.360	0.451	0.302	0.369	0.082	0.285
XYDD	100.2	0.356	0.276	0.463	0.190	0.127	0.119	0.035	0.193
XYRLG	3.510	0.800	0.725	0.501	0.259	0.157	0.130	0.186	0.291
XYG	27.26	0.726	0.667	0.448	0.221	0.178	0.217	0.026	0.301
XYSARD	485.6	0.276	0.230	0.216	0.170	0.205	0.073	0.137	0.108
GXGY	2.859	0.872	0.818	0.649	0.257	0.034	0.020	0.035	0.207
GPE	**1.107**	**0.974**	**0.951**	**0.674**	**0.989**	**0.873**	**0.849**	**0.882**	**0.858**
Improv.	30.2%	6.9%	10.4%	2.5%	38.2%	43.6%	48.0%	69.6%	51.2%

5.4 Analyses

In this section, we analyze the effects of the parameters ϵ, the embedding size h, the training time, and the base selection strategies.

Parameters ϵ. Limited by the space, Table 5 shows the average results on five datasets of various evaluations, with their ranks and the summary rank. ϵ varies in [1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1-1e-8]·2π. On the one hand, tiny ϵ leads bad results, as the GPE degenerates to the single-based GPE[2] system, which losses the benefit of the multi-bases. On the other hand, huge ϵ brings worse performance, which is unable to capture precise descriptions. We set ϵ as 2e-6π by default, since it has the best summary result.

Table 5: Average Results on Parameter ϵ

$\frac{\epsilon}{2\pi}$	MR↓		MRR↑		MP↑		KP↑		Sum
	Value	Rank	Value	Rank	Value	Rank	Value	Rank	Rank
1e-8	1.135	3	0.973	3	0.951	3	0.627	6	4
1e-7	1.156	4	0.971	5	0.947	5	0.642	3	5
1e-6	1.107	1	0.974	1	0.951	2	0.674	2	**1**
1e-5	1.255	5	0.974	1	0.952	1	0.637	4	2
1e-4	1.117	2	0.972	4	0.948	4	0.692	1	2
1e-3	1.312	6	0.925	7	0.848	7	0.629	5	6
1e-2	1.613	7	0.937	6	0.898	6	0.627	7	7
1e-1	4.192	8	0.872	8	0.824	8	0.607	8	8
1-1e-8	6.803	9	0.815	9	0.743	9	0.588	9	9

Dimension h. Table 6 left shows the MRR results of various embedding size h when training and testing on the T-drive dataset. It's clear that the GPE has the best performance on all dimensions. The performances of predefined rules based methods fluctuate on different dimensions due to lacking of dynamic, and the grid + predefined rules based methods are also affected.

Training Time. Table 6 right shows the time of pre-training the embeddings of positions and training of the trajectory similarity model on the T-drive dataset. On the whole, the GPE has the second shortest training time. Specifically, the GPE does not need to pre-train the positions, and helps the model converge faster.

Table 6: Analyses of h and Training Time on T-drive

	h (Local, MRR↑)				Training Time (s↓)
	32	64	128	256	(Position + Model)
Grid	0.788	0.863	0.897	0.900	714 (12+702)
TriW	0.876	0.851	0.874	0.837	563 (0+563)
XY	0.724	0.536	0.049	0.046	660 (0+660)
XYVV	0.414	0.440	0.359	0.117	560 (0+560)
XYDD	0.449	0.468	0.054	0.042	**499 (0+499)**
XYRLG	0.290	0.416	0.550	0.045	2053 (10+2043)
XYG	0.355	0.703	0.197	0.039	676 (10+666)
XYSARD	0.130	0.145	0.123	0.119	1702 (0+1702)
GXGY	<u>0.911</u>	<u>0.912</u>	<u>0.915</u>	<u>0.917</u>	912 (378+534)
GPE	**0.925**	**0.914**	**0.925**	**0.925**	<u>552 (0+552)</u>

Base Selection Strategies. To verify the effect of our τ-filter strategy, we compare four different base selection strategies. OneByOne simply select the next base greedily, Prime skips the products of previous bases, SingleBase uses only one base 2, and our τ-filter strategy skips the exponents of previous bases. To amplify the effect of bases, we set embedding size as 256. The selected bases and the average local results are shown in Table 7.

The content of bases 4 and 9 are duplicated with bases 2 and 3 in OneByOne, leaving no room for base 10. The Prime misses bases 6 and 10, while bases 11, 13, and base 17 can't compensate that. And SingleBase lacks multiple perspectives from multi-bases, so our τ-filter strategy outperforms others. In a nutshell, selecting multiple bases is important, duplicate exponents should be skipped, but skipping products would result in missing critical information.

Table 7: Average Local Results of Base Selection Strategies

	Bases (h=256)	MR↓	MRR↑	MP↑	KP↑
OneByOne	2,3,4,5,6,7	1.113	0.975	0.953	0.704
Prime	2,3,5,7,11,13,17	1.108	0.973	0.951	0.693
SingleBase	2	1.240	0.967	0.944	0.685
τ (Ours)	2,3,5,6,7,10	**1.106**	**0.976**	**0.955**	**0.726**

5.5 Case Studies

In this section, we conduct more evaluations on other architectures.

Performance on Other Architectures. We replace the LSTM model with the GNN model with two GATconv layers, and the Transformer model with two selt-attention encoder layers, respectively. For GNN, each trajectory forms a graph of shape '1', with node features generated by GPE based on their positions.

Limited by space, Table 8 shows the average local MRR results that train and test on all datasets, and the global MRR results that train on the T-drive dataset and test on others. The results demonstrate that GPE performs well on both GNN and Transformer architectures. The detail results are displayed in Section A.3.

Performance on Road Network. We integrate GPE with ST2Vec [14], a road network-based model that uses node2vec to encode road network nodes, to verify GPE's effectiveness in road network settings, and compare GPE with the node2vec embedding method.

Table 8: Results on Other Architectures

	GNN (MRR↑)		Transformer (MRR↑)	
	Local	Global	Local	Global
Grid	0.873	0.234	0.893	0.257
TriW	0.765	<u>0.537</u>	0.741	<u>0.460</u>
XY	0.606	0.121	0.512	0.179
XYVV	0.634	0.314	0.661	0.369
XYDD	0.770	0.318	0.547	0.120
XYRLG	<u>0.921</u>	0.432	0.883	0.298
XYG	0.827	0.304	<u>0.910</u>	0.334
XYSARD	0.032	0.028	0.191	0.114
GXGY	0.840	0.220	0.865	0.253
GPE	**0.977**	**0.993**	**0.972**	**0.981**
Improv.	5.6%	45.6%	6.2%	52.1%

Following [14], we conduct the experiments on the T-drive dataset, and adopt the HR@10↑ metric, where the ground-truth results are obtained by the TP, DITA, LCRS, and NetERP distances.

First, we simply replace the nodes' features with GPE embeddings based on the nodes' positions. Second, we concatenate node2vec and GPE embeddings while maintaining the total dimension. The results in Table 9 indicate that, GPE performs better than the node2vec method, and GPE can complement existing methods by providing spatial information while node2vec provides topological features.

Table 9: Results on Road Network

	TP	DITA	LCRS	NetERP
node2vec (h=128)	0.515	0.448	0.177	0.330
GPE (h=128)	<u>0.521</u>	<u>0.452</u>	<u>0.199</u>	<u>0.383</u>
GPE+node2vec (h=64+64)	**0.543**	**0.464**	**0.206**	**0.395**

6 Conclusion

In this paper, we investigated the problem of embedding for spatial positions, which is crucial for learning based trajectory similarity computation models. The drawbacks of existing methods hindered the performance of downstream tasks in real-world applications. We proposed the novel λ-base circular system to describe positions precisely by narrowing down the field index by index. Furthermore, we introduced the multi-base global embedding method GPE to encode positions on the globe, satisfying the good properties of global, continuous, unique, and dynamic. We conducted extensive experiments on nine baselines on five real-world datasets and four important evaluations, and the results showed that GPE dominated all compared methods in both local and global scenarios.

Acknowledgments

This work is supported by Natural Science Foundation of China (Grant No. 62272332), by Postgraduate Research & Practice Innovation Program of Jiangsu Province (Grant No. KYCX25_3466), and by Project Funded by Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] Donald J. Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD*. 359–370.

[2] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. 2014. CRAWDAD roma/taxi. https://dx.doi.org/10.15783/C7QC7M

[3] Vaida Ceikute and Christian S. Jensen. 2015. Vehicle Routing with User-Generated Trajectory Data. In *MDM*. 14–23.

[4] Yanchuan Chang, Jianzhong Qi, Yuxuan Liang, and Egemen Tanin. 2023. Contrastive Trajectory Similarity Learning with Dual-Feature Attention. In *ICDE*. 2933–2945.

[5] Bing-Jyue Chen, Chiok Yew Ho, and De-Nian Yang. 2024. AFTER: Adaptive Friend Discovery for Temporal-Spatial and Social-Aware XR. In *ICDE*. 2639–2652.

[6] Lei Chen and Raymond T. Ng. 2004. On The Marriage of Lp-norms and Edit Distance. In *VLDB*. 792–803.

[7] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *SIGMOD*. 491–502.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*. 1597–1607.

[9] Ziwen Chen, Ke Li, Silin Zhou, Lisi Chen, and Shuo Shang. 2022. Towards robust trajectory similarity computation: Representation-based spatio-temporal similarity quantification. *WWW* 26, 4 (2022), 1271–1294.

[10] Zhen Chen, Dalin Zhang, Shanshan Feng, Kaixuan Chen, Lisi Chen, Peng Han, and Shuo Shang. 2024. KGTS: Contrastive Trajectory Similarity Learning over Prompt Knowledge Graph Embedding. *AAAI* 38, 8 (2024), 8311–8319.

[11] Liwei Deng, Yan Zhao, Jin Chen, Shuncheng Liu, Yuyang Xia, and Kai Zheng. 2024. Learning to Hash for Trajectory Similarity Computation and Search. In *ICDE*. 4491–4503.

[12] Yulan Dong and Dechang Pi. 2018. Novel Privacy-preserving algorithm based on frequent path for trajectory data publishing. *Knowledge-Based Systems* 148 (2018), 55–65.

[13] Ziquan Fang, Yuntao Du, Lu Chen, Yujia Hu, Yunjun Gao, and Gang Chen. 2021. E2DTC: An End to End Deep Trajectory Clustering Framework via Self-Training. In *ICDE*. 696–707.

[14] 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 *KDD*. 347–356.

[15] Ziquan Fang, Dongen Wu, Lu Pan, Lu Chen, and Yunjun Gao. 2022. When Transfer Learning Meets Cross-City Urban Flow Prediction: Spatio-Temporal Adaptation Matters. In *IJCAI*. 2030–2036.

[16] Chunhui Feng, Zhicheng Pan, Junhua Fang, Jiajie Xu, Pengpeng Zhao, and Lei Zhao. 2022. Aries: Accurate Metric-based Representation Learning for Fast Top-k Trajectory Similarity Query. In *CIKM*. 499–508.

[17] Yunjun Gao, Ziquan Fang, Jiachen Xu, Shenghao Gong, Chunhui Shen, and Lu Chen. 2024. An Efficient and Distributed Framework for Real-Time Trajectory Stream Clustering. *TKDE* 36, 5 (2024), 1857–1873.

[18] Dongyue Guo, Zheng Zhang, Zhen Yan, Jianwei Zhang, and Yi Lin. 2024. FlightBERT++: A Non-autoregressive Multi-Horizon Flight Trajectory Prediction Framework. *AAAI* 38 (2024), 127–134.

[19] Yang Guo, Zhiqi Wang, Jin Xue, and Zili Shao. 2024. A Spatio-Temporal Series Data Model with Efficient Indexing and Layout for Cloud-Based Trajectory Data Management. In *ICDE*. 1171–1184.

[20] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. 2021. A Graph-based Approach for Trajectory Similarity Computation in Spatial Networks. In *KDD*. 556–564.

[21] Ping Han, Wenqing Wang, Qingyan Shi, and Jucai Yue. 2021. A combined online-learning model with K-means clustering and GRU neural networks for trajectory prediction. *Ad Hoc Networks* 117 (2021), 102476.

[22] Anas Ismail and Antoine Vigneron. 2015. A New Trajectory Similarity Measure for GPS Data. In *IWGS 2015*. 19–22.

[23] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. 2023. Self-supervised Trajectory Representation Learning with Temporal Regularities and Travel Semantics. In *ICDE*. IEEE.

[24] Guanyao Li, Chih-Chieh Hung, Mengyun Liu, Linfei Pan, Wen-Chih Peng, and S.-H. Gary Chan. 2021. Spatial-Temporal Similarity for Trajectories with Location Noise and Sporadic Sampling. In *ICDE*. 1224–1235.

[25] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *ICDE*. 617–628.

[26] Anqi Liang, Bin Yao, Bo Wang, Yinpei Liu, Zhida Chen, Jiong Xie, and Feifei Li. 2024. Sub-trajectory clustering with deep reinforcement learning. *VLDB J.* 33, 3 (01 May 2024), 685–702.

[27] Bin Lin and Jianwen Su. 2005. Shapes based trajectory queries for moving objects. In *ACM-GIS*. 21–30.

[28] Yan Lin, Huaiyu Wan, Shengnan Guo, Jilin Hu, Christian S. Jensen, and Youfang Lin. 2024. Pre-Training General Trajectory Embeddings With Maximum Multi-View Entropy Coding. *TKDE* 36, 12 (2024), 9037–9050.

[29] An Liu, Yifan Zhang, Xiangliang Zhang, Guanfeng Liu, Yanan Zhang, Zhixu Li, Lei Zhao, Qing Li, and Xiaofang Zhou. 2022. Representation Learning With Multi-Level Attention for Activity Trajectory Similarity Computation. *TKDE* 34, 5 (2022), 2387–2400.

[30] Yu Liu, Qian Ge, Wei Luo, Qiang Huang, Lei Zou, Haixu Wang, Xin Li, and Chang Liu. 2024. GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations. *TKDE* 36, 1 (2024), 184–198.

[31] Yuyu Luo, Wenbo Li, Tianyu Zhao, Xiang Yu, Lixi Zhang, Guoliang Li, and Nan Tang. 2020. DeepTrack: Monitoring and Exploring Spatio-Temporal Data: A Case of Tracking COVID-19. *Proc. VLDB Endow.* 13, 12 (2020), 2841–2844.

[32] Zhipeng Ma, Zheyan Tu, Xinhai Chen, Yan Zhang, Deguo Xia, Guyue Zhou, Yilun Chen, Yu Zheng, and Jiangtao Gong. 2024. More Than Routing: Joint GPS and Route Modeling for Refine Trajectory Representation Learning. In *WWW*. 3064–3075.

[33] Luis Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. 2013. Predicting Taxi–Passenger Demand Using Streaming Data. *IEEE Trans. Intell. Transp. Syst.* 14, 3 (2013), 1393–1402.

[34] Mirco Nanni and Dino Pedreschi. 2006. Time-Focused Clustering of Trajectories of Moving Objects. *J. Intell. Inf. Syst.* 27, 3 (2006), 267–289.

[35] Xiaocao Ouyang, Yan Yang, Wei Zhou, Yiling Zhang, Hao Wang, and Wei Huang. 2023. CityTrans: Domain-Adversarial Training With Knowledge Transfer for Spatio-Temporal Prediction Across Cities. *TKDE* 36, 1 (2023), 62–76.

[36] Daehee Park, Jaewoo Jeong, and Kuk-Jin Yoon. 2024. Improving Transferability for Cross-Domain Trajectory Prediction via Neural Stochastic Differential Equation. *AAAI* 38, 9 (2024), 10145–10154.

[37] Bardh Prenkaj and Paola Velardi. 2024. Unsupervised Detection of Behavioural Drifts With Dynamic Clustering and Trajectory Analysis. *TKDE* 36, 5 (2024), 2257–2270.

[38] Tangwen Qian, Yile Chen, Gao Cong, Yongjun Xu, and Fei Wang. 2024. AdapTraj: A Multi-Source Domain Generalization Framework for Multi-Agent Trajectory Prediction. In *ICDE*. 5048–5060.

[39] Sayan Ranu, Deepak P, Aditya D. Telang, Prasad Deshpande, and Sriram Raghavan. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *ICDE 2015*. 999–1010.

[40] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Kai Zheng, and Panos Kalnis. 2017. Trajectory Similarity Join in Spatial Networks. *Proc. VLDB Endow.* 10, 11 (2017), 1178–1189.

[41] Liushuai Shi, Le Wang, Sanping Zhou, and Gang Hua. 2023. Trajectory Unified Transformer for Pedestrian Trajectory Prediction. In *ICCV*. 9641–9650.

[42] Xiaodan Shi, Haoran Zhang, Wei Yuan, and Ryosuke Shibasaki. 2023. MetaTraj: Meta-Learning for Cross-Scene Cross-Object Trajectory Prediction. *TITS* 24, 12 (2023), 14000–14009.

[43] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB J.* 29, 1 (2020), 3–32.

[44] Na Ta, Guoliang Li, Yongqing Xie, Changqi Li, Shuang Hao, and Jianhua Feng. 2017. Signature-Based Trajectory Similarity Join. *IEEE Trans. Knowl. Data Eng.* 29, 4 (2017), 870–883.

[45] Na Ta, Guoliang Li, Yongqing Xie, Changqi Li, Shuang Hao, and Jianhua Feng. 2017. Signature-Based Trajectory Similarity Join. *TKDE* 29, 4 (2017), 870–883.

[46] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. 2002. Discovering Similar Multidimensional Trajectories. In *ICDE*. 673–684.

[47] Chenhao Wang, Lisi Chen, Shuo Shang, Christian S. Jensen, and Panos Kalnis. 2024. Multi-Scale Detection of Anomalous Spatio-Temporal Trajectories in Evolving Trajectory Datasets. In *KDD*. 2980–2990.

[48] Shuo Wang, Surya Nepal, Richard O. Sinnott, and Carsten Rudolph. 2019. P-STM: Privacy-Protected Social Tie Mining of Individual Trajectories. In *ICWS*. 1–10.

[49] Jiangnan Xia, Yu Yang, Senzhang Wang, Hongzhi Yin, Jiannong Cao, and Philip S. Yu. 2024. Bayes-Enhanced Multi-View Attention Networks for Robust POI Recommendation. *TKDE* 36, 7 (2024), 2895–2909.

[50] Peilun Yang, Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, and Wenjie Zhang. 2022. TMN: Trajectory Matching Networks for Predicting Similarity. In *ICDE*. 1700–1713.

[51] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *ICDE*. 2183–2188.

[52] 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 *ICDE*. 1358–1369.

[53] 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 *KDD*. 2275–2285.

[54] 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 *ACM-GIS*. 99–108.

[55] Liu Yuanjun, Liu An, Liu Guanfeng, Li Zhixu, and Zhao Lei. 2023. Towards Effective Trajectory Similarity Measure in Linear Time. In *DASFAA*. 283–299.

[56] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[57] Silin Zhou, Peng Han, Di Yao, Lisi Chen, and Xiangliang Zhang. 2023. Spatial-temporal fusion graph framework for trajectory similarity computation. *WWW* 26, 4 (2023), 1501–1523.

[58] Silin Zhou, Jing Li, Hao Wang, Shuo Shang, and Peng Han. 2023. GRLSTM: trajectory similarity computation with graph-based residual LSTM. In *AAAI*, Vol. 37. 4972–4980.

[59] Zhuang Zhuang, Tianxin Wei, Lingbo Liu, Heng Qi, Yanming Shen, and Baocai Yin. 2024. TAU: trajectory data augmentation with uncertainty for next POI recommendation. In *AAAI*. 9 pages.

A Details of Experimental Results

In the following tables, the **bold font** means the top-1 result, and the underbar font indicates the top-2 result, for each column.

A.1 Local Scenario Results

Mean Rank Results. Table 10 shows the mean rank results, and GPE outperforms other methods. On the one hand, GPE is the top-1 on four datasets, and is the top-2 on the remaining one dataset. On the other hand, the mean rank results of it on all datasets are less than 2, which means that the results are almost the ground truths, while others can not guarantee it. Moreover, among compared nine baselines, the GXGY and Grid methods are better than others, which indicates the benefit of pre-training technique, yet it is not as good as GPE, which does not require any pre-training process.

Table 10: Mean Rank Results

	T-drive	Porto	Roma	AIS	GeoLife
Grid	1.396	1.718	1.251	1.462	2.106
TriW	3.242	14.829	2.427	1.928	2.280
XY	448.036	36.687	3.985	36.484	729.188
XYVV	103.892	51.350	2.956	22.257	56.362
XYDD	406.730	13.279	24.101	17.758	39.291
XYRLG	8.494	2.592	1.025	2.778	2.660
XYG	125.309	3.398	1.043	3.929	2.633
XYSARD	180.514	51.165	1241.594	19.283	938.009
GXGY	1.267	**1.154**	2.097	4.875	4.902
GPE	**1.186**	1.184	**1.021**	**1.092**	**1.054**

Mean Reciprocal Rank Results. Table 11 shows the mean reciprocal rank results, and GPE dominants others obviously. Compared to the mean rank that treats all rank results equally to compute mean, the mean reciprocal rank gives more weight to small ranks. Therefore, the results in the table indicate that most of the results of the proposed GPE method have a small rank.

Mean Precision Results. Table 12 shows the mean precision results, and GPE outperforms others obviously. For rule-based methods, the rapid fluctuation of coordinateshinders optimizing the model, while the GPE reduces the rapid fluctuation by normalizing the value of embedding in $[-1, 1]$ with trigonometric functions.

***k*-nn Precision Results.** Table 13 shows the k-nn precision results, and GPE is the top-1 on three datasets and in the top-2 on four datasets, while others can not achieve this.

Table 11: Mean Reciprocal Rank Results

	T-drive	Porto	Roma	AIS	GeoLife
Grid	0.897	0.847	0.964	0.923	0.892
TriW	0.874	0.751	0.915	0.839	0.905
XY	0.046	0.619	0.789	0.420	0.042
XYVV	0.343	0.625	0.957	0.491	0.702
XYDD	0.047	0.679	0.317	0.482	0.257
XYRLG	0.550	0.760	0.993	0.821	0.877
XYG	0.197	0.790	0.987	0.786	0.871
XYSARD	0.122	0.684	0.041	0.491	0.044
GXGY	0.915	0.973	0.945	0.660	0.868
GPE	**0.925**	**0.974**	**0.995**	**0.980**	**0.994**

Table 12: Mean Precision Results

	T-drive	Porto	Roma	AIS	GeoLife
Grid	0.813	0.761	0.945	0.873	0.843
TriW	0.791	0.715	0.871	0.766	0.850
XY	0.013	0.560	0.680	0.317	0.019
XYVV	0.242	0.577	0.939	0.387	0.630
XYDD	0.011	0.629	0.208	0.387	0.146
XYRLG	0.408	0.668	0.988	0.735	0.825
XYG	0.121	0.720	0.978	0.704	0.810
XYSARD	0.061	0.647	0.024	0.395	0.021
GXGY	0.842	0.952	0.928	0.540	0.827
GPE	**0.854**	**0.955**	**0.991**	**0.965**	**0.990**

Table 13: *k*-nn Precision Results

	T-drive	Porto	Roma	AIS	GeoLife
Grid	0.624	0.492	0.557	**0.545**	0.521
TriW	0.516	0.449	0.589	0.514	0.722
XY	0.416	0.459	0.645	0.373	0.452
XYVV	0.193	0.487	0.583	0.339	0.200
XYDD	0.350	0.435	0.663	0.437	0.431
XYRLG	0.413	0.480	0.644	0.420	0.549
XYG	0.224	0.443	0.694	0.389	0.491
XYSARD	0.167	0.472	0.020	0.398	0.023
GXGY	0.572	0.519	**0.805**	0.525	0.823
GPE	**0.652**	**0.561**	0.799	0.519	**0.839**

A.2 Global Scenario Results

Table 14, Table 15, Table 16, Table 17, and Table 18 show the mean reciprocal rank results of training on the T-drive, Porto, Roma, AIS, and the GeoLife dataset respectively, and testing on others. It's obvious that the GPE method significantly outperforms others.

A.3 Results on Other Architectures

Table 19 show the results on the GNN model with two GATconv layers. Specifically, the left shows the average results on five datasets,

Table 14: Mean Reciprocal Rank Results of Training on T-drive

	Porto	Roma	AIS	GeoLife	Average
Grid	0.039	0.029	0.043	0.940	0.263
TriW	0.443	0.280	0.726	0.979	0.607
XY	0.032	0.379	0.230	0.033	0.169
XYVV	0.159	0.733	0.136	0.774	0.451
XYDD	0.045	0.498	0.180	0.037	0.190
XYRLG	0.065	0.131	0.167	0.675	0.259
XYG	0.047	0.262	0.296	0.281	0.221
XYSARD	0.203	0.119	0.297	0.060	0.170
GXGY	0.038	0.028	0.042	0.920	0.257
GPE	**0.969**	**0.998**	**0.994**	**0.995**	**0.989**

Table 15: Mean Reciprocal Rank Results of Training on Porto

	T-drive	Roma	AIS	GeoLife	Average
Grid	0.042	0.028	0.045	0.029	0.036
TriW	0.569	0.267	0.615	0.298	0.437
XY	0.041	0.039	0.479	0.034	0.148
XYVV	0.092	0.419	0.455	0.243	0.302
XYDD	0.055	0.042	0.372	0.040	0.127
XYRLG	0.094	0.045	0.456	0.034	0.157
XYG	0.054	0.066	0.551	0.039	0.178
XYSARD	0.106	0.120	0.552	0.040	0.205
GXGY	0.043	0.028	0.037	0.029	0.034
GPE	**0.637**	**0.980**	**0.882**	**0.993**	**0.873**

Table 16: Mean Reciprocal Rank Results of Training on Roma

	T-drive	Porto	AIS	GeoLife	Average
Grid	0.042	0.039	0.044	0.031	0.039
TriW	0.299	0.093	0.490	0.136	0.254
XY	0.047	0.042	0.315	0.038	0.110
XYVV	0.230	0.096	0.346	0.803	0.369
XYDD	0.049	0.114	0.279	0.034	0.119
XYRLG	0.115	0.052	0.289	0.067	0.130
XYG	0.325	0.103	0.279	0.161	0.217
XYSARD	0.127	0.032	0.099	0.033	0.073
GXGY	0.020	0.018	0.026	0.015	0.020
GPE	**0.729**	**0.747**	**0.925**	**0.997**	**0.849**

and the right shows the averaged MRR results of training on the one dataset and testing on the other four unseen datasets. And Table 20 show the results on the Transformer model with two selt-attention encoder layers. GPE outperforms others on both two architectures in both local and global scenarios.

Table 17: Mean Reciprocal Rank Results of Training on AIS

	T-drive	Porto	Roma	GeoLife	Average
Grid	0.039	0.035	0.025	0.029	0.032
TriW	0.053	0.042	0.029	0.036	0.040
XY	0.051	0.043	0.029	0.036	0.040
XYVV	0.081	0.039	0.146	0.064	0.082
XYDD	0.041	0.039	0.028	0.032	0.035
XYRLG	0.132	0.052	0.192	0.367	0.186
XYG	0.030	0.029	0.023	0.022	0.026
XYSARD	0.095	0.039	0.170	0.243	0.137
GXGY	0.043	0.039	0.027	0.030	0.035
GPE	**0.794**	**0.763**	**0.986**	**0.987**	**0.882**

Table 18: Mean Reciprocal Rank Results of Training on GeoLife

	T-drive	Porto	Roma	AIS	Average
Grid	0.612	0.038	0.029	0.045	0.181
TriW	0.580	0.101	0.075	0.627	0.346
XY	0.066	0.009	0.036	0.163	0.069
XYVV	0.253	0.109	0.596	0.180	0.285
XYDD	0.381	0.055	0.019	0.319	0.193
XYRLG	0.565	0.054	0.163	0.381	0.291
XYG	0.539	0.134	0.205	0.324	0.301
XYSARD	0.108	0.040	0.092	0.192	0.108
GXGY	**0.770**	0.018	0.014	0.026	0.207
GPE	0.751	**0.757**	**0.996**	**0.926**	**0.858**

Table 19: Average Results on GNN

	Local				Global (MRR↑)				
	MR↓	MRR↑	MP↑	KP↑	T-drive	Porto	Roma	AIS	GeoLife
Grid	3.609	0.873	0.809	0.849	0.234	0.002	0.002	0.002	0.167
TriW	3.262	0.765	0.673	0.903	0.537	0.182	0.303	0.004	0.418
XY	9.862	0.606	0.473	0.881	0.121	0.116	0.025	0.002	0.204
XYVV	41.12	0.634	0.545	0.650	0.314	0.110	0.085	0.002	0.263
XYDD	12.82	0.770	0.699	0.796	0.318	0.215	0.123	0.002	0.274
XYRLG	1.979	0.921	0.877	0.783	0.432	0.198	0.169	0.041	0.346
XYG	3.947	0.827	0.753	0.848	0.304	0.026	0.030	0.002	0.241
XYSARD	1454	0.032	0.014	0.316	0.028	0.034	0.027	0.008	0.015
GXGY	3.839	0.840	0.793	0.905	0.220	0.001	0.002	0.002	0.219
GPE	**1.275**	**0.977**	**0.959**	**0.924**	**0.993**	**0.949**	**0.944**	**0.969**	**0.956**
Improv.	35.6%	5.6%	8.2%	1.9%	45.6%	73.4%	64.1%	92.8%	53.8%

Table 20: Average Results on Transformer

	Local				Global (MRR↑)				
	MR↓	MRR↑	MP↑	KP↑	T-drive	Porto	Roma	AIS	GeoLife
Grid	4.724	0.893	0.841	0.602	0.257	0.035	0.038	0.035	0.199
TriW	46.43	0.741	0.685	0.594	0.460	0.169	0.506	0.038	0.488
XY	62.17	0.512	0.426	0.499	0.179	0.170	0.096	0.035	0.187
XYVV	62.83	0.661	0.592	0.447	0.369	0.291	0.280	0.098	0.249
XYDD	83.85	0.547	0.469	0.618	0.120	0.145	0.058	0.035	0.114
XYRLG	4.939	0.883	0.831	**0.685**	0.298	0.090	0.096	0.257	0.267
XYG	1.828	0.910	0.861	0.671	0.334	0.124	0.096	0.036	0.288
XYSARD	786.0	0.191	0.129	0.142	0.114	0.150	0.136	0.070	0.116
GXGY	2.709	0.865	0.802	0.637	0.253	0.035	0.039	0.035	0.203
GPE	**1.321**	**0.972**	**0.951**	0.674	**0.981**	**0.906**	**0.872**	**0.919**	**0.894**
Improv.	27.7%	6.2%	9.0%	-	52.1%	61.5%	36.6%	66.2%	40.6%