Towards Robust Trajectory Embedding for Similarity Computation: When Triangle Inequality Violations in Distance Metrics Matter

Jianing Si[†], Haitao Yuan[‡], Nan Jiang[†], Minxiao Chen[†], Xiao Ma[†], Shangguang Wang[†]

[†]*BUPT, China*   [‡]*NTU, Singapore*

[†]{sijianing, jn_bupt, chenminxiao, maxiao18, sgwang}@bupt.edu.cn,   [‡]haitao.yuan@ntu.edu.sg

Abstract—**Trajectory similarity is a cornerstone of trajectory data management and analysis. Traditional similarity functions often suffer from high computational complexity and a reliance on specific distance metrics, prompting a shift towards deep representation learning in Euclidean space. However, existing Euclidean-based trajectory embeddings often face challenges due to the triangle inequality constraints that do not universally hold for trajectory data. To address this issue, this paper introduces a novel approach by incorporating non-Euclidean geometry, specifically hyperbolic space, into trajectory representation learning. We present the first-ever integration of hyperbolic space to resolve the inherent limitations of the triangle inequality in Euclidean embeddings. In particular, we achieve it by designing a Lorentz distance measure, which is proven to overcome triangle inequality constraints. Additionally, we design a model-agnostic framework LH-plugin to seamlessly integrate hyperbolic embeddings into existing representation learning pipelines. This includes a novel projection method optimized with the Cosh function to prevent the diminishment of distances, supported by a theoretical foundation. Furthermore, we propose a dynamic fusion distance that intelligently adapts to variations in triangle inequality constraints across different trajectory pairs, blending Lorentzian and Euclidean distances for more robust similarity calculations. Comprehensive experimental evaluations demonstrate that our approach effectively enhances the accuracy of trajectory similarity measures in state-of-the-art models across multiple real-world datasets. The LH-plugin not only addresses the triangle inequality issues but also significantly refines the precision of trajectory similarity computations, marking a substantial advancement in the field of trajectory representation learning.**

I. INTRODUCTION

Computing trajectory similarity is crucial for various tasks in trajectory data management and analytics, as referenced in several studies [1]–[5]. In particular, trajectory similarity metrics effectively measure the distance between two distinct trajectories, which is vital for many downstream applications such as trajectory clustering [6]–[9], efficient trajectory querying and retrieval [10], [11], and the optimization of traffic conditions [12]–[14]. To address the needs of these diverse applications, previous research has developed a range of methods for calculating trajectory distances, including DTW [15], [16], SSPD [17], EDR [18], and Hausdorff [19]. However, these methods are inherently constrained by computational challenges due to their super-quadratic time complexity. Therefore, recent advancements in trajectory similarity research have incorporated representation learning techniques to enhance the efficiency of similarity computations, as highlighted in several
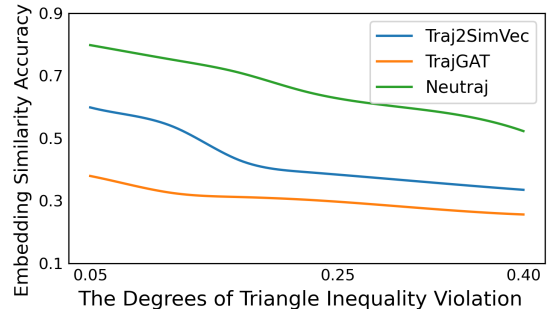

Fig. 1: Embedding Accuracy v.s. Triangle Inequality Violation

studies [20]–[25]. These approaches first transform trajectories into high-dimensional embeddings in Euclidean space, and then use some distance functions between embeddings such as Euclidean distance to measure the trajectory similarity.

However, this approach overlooks a critical issue: ***While metrics between different embeddings in Euclidean space are constrained by the triangle inequality [26]–[28], the similarities derived using traditional measures such as DTW and EDR on raw trajectories may not adhere to this constraint.*** As illustrated in Figure 1, we evaluate the performance of existing advanced trajectory representation learning methods by sampling trajectories with varying degrees of triangular inequality violation[1]. The results indicate that the accuracy of these methods significantly deteriorates as the degree of triangular inequality violation increases. To overcome the limitations of Euclidean space in trajectory representation learning, we introduce the concept of non-Euclidean Hyperbolic space, which shows promise in bypassing these constraints [29]–[31]. Moreover, to effectively implement this in trajectory representation learning, we identify and address three critical challenges with corresponding solutions detailed in this paper.

Challenge 1 (C1): *Designing a suitable distance function for the hyperbolic space.* The ideal distance function should be non-negative, with a trajectory being most similar to itself, and should not be constrained by the triangle inequality.

Challenge 2 (C2): *Effectively achieving hyperbolic embedding using current trajectory encoding models.* Directly

---

[1]The triangular inequality violation among trajectories refers to the proportion of trajectory pairs that do not satisfy the triangle inequality. This is quantified by the average relative violation, as defined in Section V-A.

transforming trajectories into hyperbolic space vectors is challenging, as these vectors must adhere to specific hyperbolic constraints that existing models cannot effectively handle. Therefore, developing a model-agnostic approach for hyperbolic embedding is essential, as it ensures compatibility with real-world applications and seamless integration with existing embedding frameworks without requiring modifications.

Challenge 3 (C3): *Adapting to varied magnitudes of triangle inequality constraints among different trajectory pairs.* The extent of triangle inequality constraints varies across different similarity functions and datasets, necessitating a dynamic mechanism that ensures robust embeddings adaptable to diverse scenarios.

Our proposed solution: First, to address **C1**, we design the Lorentz distance, calculated via the Lorentz inner product, to quantify the similarity between embeddings. We demonstrate that this distance is not bound by the triangle inequality, making it a robust measure of similarity. Second, for **C2**, we introduce a model-agnostic **L**orentzian **H**yperbolic plugin (**LH**-plugin) that seamlessly integrates with any existing trajectory embedding model without necessitating alterations to the original frameworks. We first explore a vanilla hyperbolic projection method for converting Euclidean embeddings into hyperbolic ones. However, upon observing and proving that this projection can diminish distances as embedding norms increase, we refine our approach. In particular, we develop an enhanced projection technique using the hyperbolic cosine function, ensuring a more effective and theoretically sound mapping that avoids the issue of diminishing distances. Lastly, to tackle **C3**, we assess the impact of varying triangle inequality constraints across datasets and similarity measures, and then introduce the Dynamic Fusion Distance, which adjusts to the unique characteristics of each trajectory pair. This approach dynamically blends Lorentzian and Euclidean distances, creating a hybrid metric tailored to the magnitude of triangle inequality constraints.

In summary, we make the following contributions:

- To the best of our knowledge, we are the first to integrate hyperbolic space into trajectory representation learning. We design the Lorentz distance, rigorously proving its ability to overcome triangle inequality constraints in this space. (Section II)
- We develop a model-agnostic framework **LH**-plugin that facilitates the integration of hyperbolic embeddings into the existing trajectory representation learning pipeline. We provide a theoretical justification for this optimization. (Sections III and IV)
- We introduce a dynamic fusion method that adjusts the distance according to trajectory pair combinations. This innovative approach adapts to the diverse magnitudes of triangle inequality constraints encountered across trajectory pairs. (Section V)
- We conduct comprehensive experiments with several state-of-the-art trajectory embedding models across various real-world datasets. Our results validate the effective integration

of the **LH**-plugin into these models. (Section VI)

II. PRELIMINARY

In this section, we first introduce the original pipeline of trajectory embedding in Euclidean space and discuss the triangle inequality problem encountered by this method for similarity calculation in Section II-A. Next, we propose an innovative Lorentz distance between different embeddings within the hyperbolic space in Section II-B, demonstrating its effectiveness as a distance measure while avoiding the triangle inequality constraint. Finally, we present the motivation behind our plugin-based approach for trajectory embedding in hyperbolic space in Section II-C.

A. Trajectory Embedding

Trajectory. The trajectory $T = [p_1, ..., p_n]$ is a sequence of points where each point $p_i = (lon_i, lat_i)$ (or $p_i = (lon_i, lat_i, t_i)$) is a tuple, and lon_i, lat_i (and t_i) respectively represent longitude, latitude (and timestamp).

Trajectory Similarity with Distance Measurement. Given two trajectories $T_a = [p_{a_1}, ..., p_{a_n}]$ and $T_b = [p_{b_1}, ..., p_{b_m}]$, the similarity between T_a and T_b is measured by specific distance metrics on the corresponding point sequences, such as DTW [15], Hausdorff [19], SSPD [17], EDR [18], etc. In particular, the higher the similarity, the lower the trajectory distance. For simplicity, we use $\mathbf{Dist}_*(\cdot, \cdot)$ to denote the trajectory distance. For instance, the DTW trajectory distance can be succinctly described as follows:

$$\mathbf{Dist}_D(T_a, T_b) = DTW[p_{a_n}, p_{b_m}]$$
$$DTW[p_{a_i}, p_{b_j}] = d(p_{a_i}, p_{b_j}) + \min \big(DTW[p_{a_{i-1}}, p_{b_j}], \quad (1)$$
$$DTW[p_{a_i}, p_{b_{j-1}}], DTW[p_{a_{i-1}}, p_{b_{j-1}}] \big)$$

where $d(p_{a_i}, p_{b_j})$ represents the Euclidean distance between points p_{a_i} and p_{b_j}.

Remark. To prevent potential misunderstandings, we use $\mathbf{Dist}_*(\cdot, \cdot)$ interchangeably to represent either trajectory similarity or distance in this paper.

Trajectory Embedding. The concept of trajectory embedding involves identifying an encoding function $g(\cdot)$ called the Trajectory Embedding model, which maps a trajectory onto a d-dimensional vector, represented as $\mathbf{x} = g(T) \in \mathbb{R}^d$.

Trajectory Embedding for Similarity Computation. The primary objective of trajectory similarity embedding is to develop an encoder model such that the distance of embedding pair $(\mathbf{x}_a, \mathbf{x}_b)$, for any trajectory pair (T_a, T_b), closely approximates the provided trajectory distance measure, such as $\mathbf{Dist}_D(T_a, T_b)$. Prior studies have treated the corresponding trajectory embeddings $\mathbf{x}_a$ and $\mathbf{x}_b$ as vectors within Euclidean space, employing Euclidean distance to gauge the distance between these vectors. Given a database $\mathcal{T} = \{T_1, T_2, ..., T_N\}$ comprising N trajectories, alongside a distance measurement function $\mathbf{Dist}_*(\cdot, \cdot)$, the aim of trajectory similarity embedding is to discover a mapping $g(\cdot)$ that reduces the disparity between the Euclidean distance among trajectory embeddings $\mathbf{d}_{Eu}(g(T_i), g(T_j))$ and the actual distance $\mathbf{Dist}_*(T_i, T_j)$ to the minimum. This process can be depicted as follows:
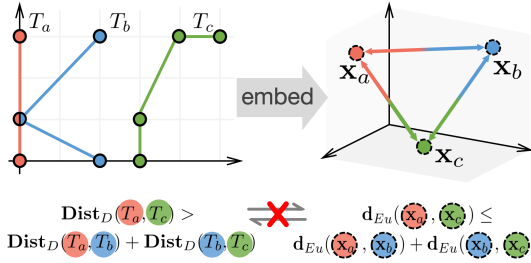
Fig. 2: An Example of Triangle Inequality Contradiction in Trajectory Embedding

$$\underset{g}{\operatorname{argmin}} \sum_{T_i, T_j \in \mathcal{T}} \left| \operatorname{Dist}_*(T_i, T_j) - \mathbf{d}_{Eu}(g(T_i), g(T_j)) \right| \quad (2)$$

Triangle Inequality in Trajectory Embedding. Because the trajectory embedding space is a d-dimensional Euclidean space, the embedding distance between different trajectories satisfies the triangle inequality constraint as follows:

$$\forall T_a, T_b, T_c \in \mathcal{T}, \quad \mathbf{x}_* = g(T_*)$$
$$\text{s.t. } \mathbf{d}_{Eu}(\mathbf{x}_a, \mathbf{x}_b) + \mathbf{d}_{Eu}(\mathbf{x}_b, \mathbf{x}_c) \geq \mathbf{d}_{Eu}(\mathbf{x}_a, \mathbf{x}_c) \quad (3)$$

Triangle Inequality Violation in Trajectory Similarity. Many trajectory distances, such as DTW and SSPD, don't satisfy the triangle inequality. Specifically, we can use a specific example to clarify this issue as follows.

Example 1. *In Figure 2, let* $T_a = [(0,0), (0,1), (0,3)]$, $T_b = [(2,0), (0,1), (2,3)]$ *and* $T_c = [(3,0), (3,1), (4,3), (5,3)]$. *Using the DTW distance of Formula 1, the calculated trajectory distances are* $\operatorname{Dist}_D(T_a, T_b) = 4$, $\operatorname{Dist}_D(T_b, T_c) = 9$, *and* $\operatorname{Dist}_D(T_a, T_c) = 15$. *Notably,* $\operatorname{Dist}_D(T_a, T_c) > \operatorname{Dist}_D(T_a, T_b) + \operatorname{Dist}_D(T_b, T_c)$, *which is contradicted with the constraint in Formula 3.*

Consequently, for most similarity methods that do not satisfy the triangle inequality, the approximations of similarity through representation learning might inevitably deviate from the true similarity values. In certain cases, this discrepancy could even amplify, leading to distortions in the trajectory embeddings.

B. Hyperbolic Space and Lorentz Distance

Hyperbolic Space. To avoid the triangle inequality constraint of high-dimensional embeddings in Euclidean space, we introduce the non-Euclidean hyperbolic space, which has the potential to bypass this limitation [29]–[31]. In particular, this hyperbolic space is built on the Lorentz inner product, denoted as $\langle \mathbf{a}, \mathbf{b} \rangle = -a_0 b_0 + \sum_{i=1}^n a_i b_i$, where $\mathbf{a} = [a_0, \cdots, a_n] \in \mathbb{R}^{n+1}$ and $\mathbf{b} = [b_0, \cdots, b_n] \in \mathbb{R}^{n+1}$ are two $n+1$-dimensional vectors. This space consists of all high-dimensional vectors that satisfy a specific condition within the Lorentz inner product, as described below.

Definition 2 (Hyperbolic Space).

$$\mathcal{H}(\beta) = \{\mathbf{a} \in \mathbb{R}^{n+1} : \langle \mathbf{a}, \mathbf{a} \rangle = -\beta, \ a_0 \geq \sqrt{\beta}\} \quad (\beta > 0) \quad (4)$$

where $\langle \mathbf{a}, \mathbf{a} \rangle$ *represents the self Lorentz inner product, and* β *is a hyper-parameter that influences the shape of the space.*

Lorentz Distance. Given a hyperbolic space $\mathcal{H}(\beta)$, we define the function *Lorentz Distance* based on the Lorentz inner product as follows:

Definition 3 (Lorentz Distance).

$$\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) = |\langle \mathbf{a}, \mathbf{b} \rangle| - \beta \quad (\mathbf{a}, \mathbf{b} \in \mathcal{H}(\beta)) \quad (5)$$

where $|\langle \mathbf{a}, \mathbf{b} \rangle|$ *means the absolute value of* $\langle \mathbf{a}, \mathbf{b} \rangle$.

We can use the function *Lorentz Distance* to measure the distance between any two embeddings in the space due to its non-negativity and minimal value of self-distance as described in Lemma 4. Additionally, it can relax the constraint of the triangle inequality as explained in Lemma 5.

Lemma 4. $\forall \mathbf{a}, \mathbf{b} \in \mathcal{H}(\beta)$, $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) \geq 0$, *and* $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) = 0$ *only when* $\mathbf{a} = \mathbf{b}$.

Proof. Suppose $\boldsymbol{a} = [a_0, \cdots, a_n]$ *and* $\boldsymbol{b} = [b_0, \cdots, b_n]$ *are any two embeddings in the space* $\mathcal{H}(\beta)$. *We have* $a_0 \geq \sqrt{\beta}$, $b_0 \geq \sqrt{\beta}$, $a_0^2 - \sum_{i=1}^n a_i^2 = \beta$, *and* $b_0^2 - \sum_{i=1}^n b_i^2 = \beta$. *Define a function* $f(x) = \beta x^2 - 2\langle \boldsymbol{a}, \boldsymbol{b} \rangle x + \beta$. *We can further express this function by extending the Lorentz inner product and replacing* β *in the quadratic coefficient and constant term with* $a_0^2 - \sum_{i=1}^n a_i^2$ *and* $b_0^2 - \sum_{i=1}^n b_i^2$, *respectively, as follows:*

$$f(x) = (a_0^2 - \sum_{i=1}^n a_i^2)x^2 + 2(a_0 b_0 - \sum_{i=1}^n a_i b_i)x + (b_0^2 - \sum_{i=1}^n b_i^2)$$
$$= (a_0 x + b_0)^2 - \sum_{i=1}^n (a_i x + b_i)^2 \quad (6)$$

Let $x = -\frac{b_0}{a_0} < 0$. *We have:*

$$f\left(-\frac{b_0}{a_0}\right) = -\sum_{i=1}^n \left(-a_i \frac{b_0}{a_0} + b_i\right)^2 \leq 0 \quad (7)$$

Since the quadratic coefficient $\beta > 0$, *we have* $f(-\infty) > 0$ *and* $f(+\infty) > 0$. *Therefore, the quadratic equation* $f(x) = 0$ *has at least one root. The discriminant* Δ *of* $f(x) = 0$ *is given by:*

$$\Delta = (2\langle \boldsymbol{a}, \boldsymbol{b} \rangle)^2 - 4\beta^2 \geq 0 \quad (8)$$

Thus, we have $\langle \boldsymbol{a}, \boldsymbol{b} \rangle^2 \geq \beta^2 \Rightarrow |\langle \boldsymbol{a}, \boldsymbol{b} \rangle| \geq \beta \Rightarrow \mathbf{d}_{Lo}(\boldsymbol{a}, \boldsymbol{b}) = |\langle \boldsymbol{a}, \boldsymbol{b} \rangle| - \beta \geq 0$. *In particular, if* $\mathbf{d}_{Lo}(\boldsymbol{a}, \boldsymbol{b}) = |\langle \boldsymbol{a}, \boldsymbol{b} \rangle| - \beta = 0$, *we have* $f(x) \geq 0$ *and only one root for the quadratic equation* $f(x) = 0$. *Considering that* $f(-\frac{b_0}{a_0}) \leq 0$, *the root should be* $x = -\frac{b_0}{a_0} < 0$. *The root can also be computed as* $x = \frac{-(-2\langle \boldsymbol{a}, \boldsymbol{b} \rangle)}{2\beta} = \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{\beta}$ *according to the root-finding algorithm. Given that* $|\langle \boldsymbol{a}, \boldsymbol{b} \rangle| - \beta = 0$ *and* $x < 0$, *we find the root* $x = -1$. *Therefore,* $x = -\frac{b_0}{a_0} = -1 \Rightarrow a_0 = b_0$. *Additionally:*

$$f(-1) = (b_0 - a_0)^2 - \sum_{i=1}^n (b_i - a_i)^2 = -\sum_{i=1}^n (b_i - a_i)^2 = 0 \quad (9)$$

Thus, $b_i - a_i = 0$ *is satisfied for each* $i \in [0, n]$, *implying* $\boldsymbol{b} = \boldsymbol{a}$.

Lemma 5. $\exists \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{H}(\beta)$, *the triangle inequality* $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) \leq \mathbf{d}_{Lo}(\mathbf{a}, \mathbf{c}) + \mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c})$ *is not satisfied.*

Proof. We need to find at least one set of three embeddings $\mathbf{a}, \mathbf{b}, \mathbf{c}$ *that does not satisfy this inequality. Given any* $a_0 > \sqrt{\beta}$, *consider its corresponding embedding* $\mathbf{a} = [a_0, a_1, \ldots, a_n] \in \mathcal{H}(\beta)$. *Thus, we have* $\langle \mathbf{a}, \mathbf{a} \rangle = a_0^2 - \sum_{i=1}^n a_i^2 = -\beta$. *Next, consider another embedding* $\mathbf{b} = [a_0, -a_1, \ldots, -a_n]$. *We also have* $\langle \mathbf{b}, \mathbf{b} \rangle = a_0^2 - \sum_{i=1}^n (-a_i)^2 = -\beta$. *Therefore,* $\mathbf{b}$ *also belongs to* $\mathcal{H}(\beta)$. *Finally, consider the embedding* $\mathbf{c} = [\sqrt{\beta}, 0, \ldots, 0]$, *which also belongs to the space due to* $\langle \mathbf{c}, \mathbf{c} \rangle = -(\sqrt{\beta} \cdot \sqrt{\beta}) = -\beta$. *Now, we compute the distances between the three embeddings:*

$$\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) = |\langle \mathbf{a}, \mathbf{b} \rangle| - \beta = |-a_0^2 + \sum_{i=1}^n -a_i^2| - \beta$$
$$= |-2a_0^2 + (a_0^2 - \sum_{i=1}^n a_i^2)| - \beta = 2a_0^2 - 2\beta$$

$$\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{c}) = |\langle \mathbf{a}, \mathbf{c} \rangle| - \beta = |-a_0\sqrt{\beta}| - \beta = a_0\sqrt{\beta} - \beta$$

$$\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c}) = |\langle \mathbf{b}, \mathbf{c} \rangle| - \beta = |-a_0\sqrt{\beta}| - \beta = a_0\sqrt{\beta} - \beta$$
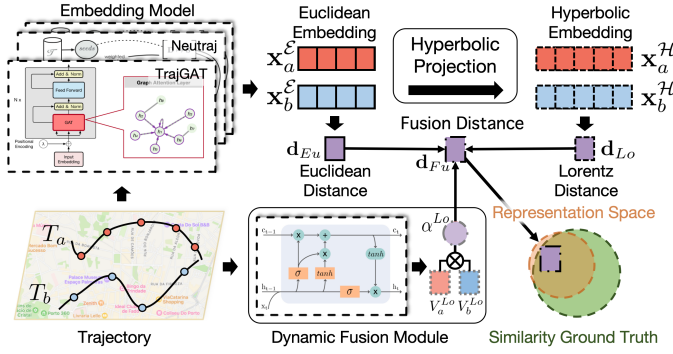
Fig. 3: The Framework Overview of **LH**-plugin

Therefore, we have:
$$\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{c}) + \mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c}) = 2a_0\sqrt{\beta} - 2\beta$$
Comparing this with $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b})$:
$$2a_0\sqrt{\beta} - 2\beta < 2a_0^2 - 2\beta = \mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b})$$
Thus, we have found three embeddings in the space that do not satisfy the triangle inequality.

C. Embedding Methods in Hyperbolic Space

The intuitive method for mapping trajectories into hyperbolic space is to directly transform operators from Euclidean to hyperbolic space, constructing hyperbolic neural network layers. For instance, Ganea et al. [32] adapt structures like MLP and CNN from Euclidean space to hyperbolic neural networks by combining Möbius gyrovector spaces with the Riemannian geometry of the Poincaré model. This allows the entire trajectory embedding process to operate in hyperbolic space, requiring the model to be built from scratch. However, this method has two major drawbacks: 1) Current GPU optimizations are designed for standard neural network operators in Euclidean space, and there is no comparable optimization for hyperbolic neural network operators. This results in significantly reduced efficiency for training and inference in purely hyperbolic models. 2) Adapting existing trajectory embedding methods to pure hyperbolic transformations is highly complex. Aside from neural networks, there are no standard hyperbolic space solutions for data processing and training in these specific embedding methods.

In contrast, we introduce the *Plugin-based* approach in this paper. This method retains as much of the Euclidean space components in trajectory embedding as possible, only introducing hyperbolic space in the output phase by mapping Euclidean vectors to hyperbolic ones. This approach preserves the original embedding pipeline, fully utilizing the efficiency of neural networks in Euclidean space while maintaining the techniques used in existing embedding methods. As a result, it enables a model-agnostic transition to hyperbolic embedding without sacrificing speed or flexibility.

III. Framework Overview

Motivation. Existing trajectory embedding works, such as those described in [22], [33]–[35], typically assess trajectory similarity using Euclidean distance between embeddings, a measure constrained by the triangle inequality. As explored in Section II-B, adopting hyperbolic space and the Lorentz distance can mitigate these limitations. Accordingly, our goal is to enhance trajectory embeddings by incorporating these insights. Rather than developing an entirely new embedding model, we introduce a lightweight framework named **LH**-plugin. This framework is designed to adapt existing embedding models, providing benefits in both training and inference efficiency, as discussed in Section II-C. Furthermore, the plugin-based approach ensures plug-and-play compatibility, avoiding disruptions to the existing model's workflow and enhancing its practical robustness.

As shown in Figure 3, the framework incorporates two additional modules into the existing embedding pipeline: the *Hyperbolic Projection* module, which maps embeddings into hyperbolic space, and the *Dynamic Fusion* module, which dynamically fuses the Euclidean distance and Lorentz distance with a learnable ratio to measure trajectory similarity. Below, we explain how these modules enhance the existing similarity/distance computation workflow with trajectory embeddings.

1. Generating Embeddings in Euclidean Space. Given two trajectories T_a and T_b, we first use existing embedding models such as **Neutraj** and **TrajGAT** to generate embeddings, denoted as $\mathbf{x}_a^{\mathcal{E}}$ and $\mathbf{x}_b^{\mathcal{E}}$, which are vectors in Euclidean space.
2. Mapping Embeddings to Hyperbolic Space. We design a projection algorithm in the *Hyperbolic Projection* module to transform $\mathbf{x}_a^{\mathcal{E}}$ and $\mathbf{x}_b^{\mathcal{E}}$ into hyperbolic embeddings, $\mathbf{x}_a^{\mathcal{H}}$ and $\mathbf{x}_b^{\mathcal{H}}$, respectively.
3. Computing Fusion Distance. While hyperbolic embeddings mitigate triangle inequality constraints, there are instances where Euclidean distance is more appropriate. To leverage the strengths of both distances, we fuse them rather than relying solely on Lorentz distance. The fusion ratio varies across trajectory pairs, necessitating the *Dynamic Fusion* module. This module generates a ratio α^{Lo} for the given trajectory pair (T_a, T_b) and uses it to fuse the Euclidean distance $\mathbf{d}_{Eu}$ and hyperbolic distance $\mathbf{d}_{Lo}$ into the fusion distance $\mathbf{d}_{Fu}$. The $\mathbf{d}_{Fu}$ and the ground truth distance are then used to compute the loss for training models within the *Dynamic Fusion* module.

IV. Hyperbolic Projection

In this section, we introduce the *Vanilla Projection* method and analyze the challenges related to distance degradation in detail in Section IV-A. To address these shortcomings, we subsequently introduce the *Cosh Projection* method in Section IV-B, which is designed to mitigate the identified issues effectively.

A. Vanilla Hyperbolic Projection

In an n-dimensional Euclidean space, an embedding $\mathbf{x}^{\mathcal{E}} \in \mathbf{R}^n$ possesses n degrees of freedom. In contrast, embeddings in an n-dimensional hyperbolic space exhibit only $n-1$ degrees of freedom due to an inherent constraint imposed by the space's definition. To align the degrees of freedom between these spaces, it is necessary to map an n-dimensional Euclidean embedding into a $(n+1)$-dimensional hyperbolic

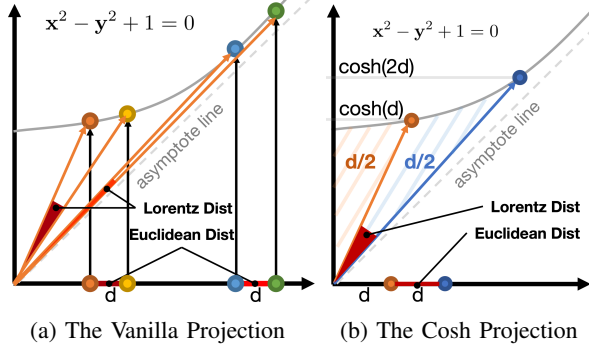(a) The Vanilla Projection   (b) The Cosh Projection

Fig. 4: The comparison between Vanilla Hyperbolic Projection and Cosh Hyperbolic Projection

embedding $\mathbf{x}^{\mathcal{H}} \in \mathbf{R}^{n+1}$. Specifically, each point within the Euclidean space $\mathcal{E}$ is uniquely mapped to a corresponding point in the hyperbolic space $\mathcal{H}$, thereby establishing a bijective relationship between the two spaces. This property is crucial in representation learning, ensuring that distinct trajectories T and T', represented by different Euclidean vectors $\mathbf{x}^{\mathcal{E}}$ and $\mathbf{x}'^{\mathcal{E}}$, do not converge to the same hyperbolic point $\mathbf{x}^{\mathcal{H}}$ through the mapping function ϕ. Failure to maintain such distinct mappings would result in trajectory confusion, leading to erroneous similarity assessments among trajectories. To achieve this bijective mapping, a straightforward approach involves a direct projection technique. This technique extends points from the Euclidean space $\mathcal{E}$ into the hyperbolic space by introducing an additional dimension. Dimensions $[1, n]$ are preserved as-is, while the new dimension's value is determined to satisfy the hyperbolic plane $\mathcal{H}(\beta)$ conditions. We refer to this mapping strategy as the *Vanilla Hyperbolic Projection* as follows:

$$\phi(\mathbf{x}^{\mathcal{E}}) \rightarrow \mathbf{x}^{\mathcal{H}} : \ x_i^{\mathcal{H}} = \begin{cases} x_{i-1}^{\mathcal{E}} & \text{if } 1 \leq i \leq n, \\ \sqrt{\sum_{j=1}^{n}(x_j^{\mathcal{E}})^2 + \beta} & \text{if } i = 0. \end{cases}$$

Lorentz distance degradation analysis. However, a significant limitation of the Vanilla Hyperbolic Projection is its failure to preserve distances between different embeddings measured by the Lorentz distance, particularly when the magnitudes of the original embeddings are substantial. We employ a two-dimensional hyperbolic space for illustration to elucidate this issue more clearly. As depicted in Figure 4a, we simplify the hyperbolic space to the equation $x^2 - y^2 + 1 = 0$, represented by the gray hyperbola. We consider points in $\mathcal{E}$ positioned along the positive x-axis. The black arrows in the figure demonstrate the projection process, while the orange arrows indicate the resultant vectors in the hyperbolic space. The red cones between the two vectors represent the angle between them. Significantly, the size of this angle exhibits a positive correlation with the Lorentz distance, particularly when the magnitudes of the Euclidean embeddings are large. This observation inspires us to deduce a theorem as follows.

Theorem 6. *For any two vectors* $\mathbf{a} = (a_0, \cdots, a_n)$ *and* $\mathbf{b} = (b_0, \cdots, b_n)$ *in* $\mathcal{H}(\beta)$, *if* $a_0, b_0 \gg \beta$ *and* $a_0, b_0 \gg |a_0 - b_0|$, *then the Lorentz distance* $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) \rightarrow 0$.

Proof. First, consider the unit vector $\mathbf{e} = (1, 0, \cdots, 0)$ *along the x-axis. We can compute the angles between the unit vector and* $\mathbf{a}$, $\mathbf{b}$

as follows:
$$\mathbf{a}, \mathbf{b} \in \mathcal{H} \Rightarrow a_0^2 - \sum_{i=1}^{n} a_i^2 = \beta, \quad b_0^2 - \sum_{i=1}^{n} b_i^2 = \beta$$

$$\cos(\mathbf{a}, \mathbf{e}) = \frac{\mathbf{a} \cdot \mathbf{e}}{|\mathbf{a}| \cdot |\mathbf{e}|} = \frac{a_0}{\sqrt{a_0^2 + \sum_{i=1}^{n} a_i^2}} = \frac{a_0}{\sqrt{2a_0^2 - \beta}}$$

$$\cos(\mathbf{b}, \mathbf{e}) = \frac{\mathbf{b} \cdot \mathbf{e}}{|\mathbf{b}| \cdot |\mathbf{e}|} = \frac{b_0}{\sqrt{b_0^2 + \sum_{i=1}^{n} b_i^2}} = \frac{b_0}{\sqrt{2b_0^2 - \beta}}$$

Given $a_0, b_0 \gg \beta$, *we have:*
$$\cos(\mathbf{a}, \mathbf{e}) \approx \cos(\mathbf{b}, \mathbf{e}) \approx \sqrt{2}/2$$

Thus, the angle between $\mathbf{a}$ *and* $\mathbf{b}$ *is near zero, implying* $\cos(\mathbf{a}, \mathbf{b}) = 1$. *Next, we compute the Lorentz distance between* $\mathbf{a}$ *and* $\mathbf{b}$:

$$\langle \mathbf{a}, \mathbf{b} \rangle = -a_0 b_0 + \sum_{i=1}^{n} a_i b_i = -2a_0 b_0 + \mathbf{a} \cdot \mathbf{b} = -2a_0 b_0 + |\mathbf{a}||\mathbf{b}| \cos(\mathbf{a}, \mathbf{b})$$

Given $\cos(\mathbf{a}, \mathbf{b}) \approx 1$, *we get:*

$$\langle \mathbf{a}, \mathbf{b} \rangle \approx -2a_0 b_0 + \sqrt{\sum_{i=0}^{n} a_i^2}\sqrt{\sum_{i=0}^{n} b_i^2} = -2a_0 b_0 + \sqrt{2a_0^2 - \beta}\sqrt{2b_0^2 - \beta}$$

$$= \frac{(2a_0^2 - \beta)(2b_0^2 - \beta) - 4a_0^2 b_0^2}{\sqrt{2a_0^2 - \beta}\sqrt{2b_0^2 - \beta} + 2a_0 b_0} = \frac{\frac{\beta^2}{a_0^2} - 2\beta(1 + \frac{b_0^2}{a_0^2})}{\sqrt{2 - \frac{\beta}{a_0^2}}\sqrt{2\frac{b_0^2}{a_0^2} - \frac{\beta}{a_0^2} + 2\frac{b_0}{a_0}}}$$

Since $a_0, b_0 \gg \beta$ *and* $a_0, b_0 \gg |a_0 - b_0|$, *we have* $\frac{\beta}{a_0^2} \approx 0$, $\frac{b_0}{a_0} = \frac{a_0 + (b_0 - a_0)}{a_0} = 1 + \frac{b_0 - a_0}{a_0} \approx 1$ *and hence* $\langle \mathbf{a}, \mathbf{b} \rangle \approx \frac{-2\beta \cdot 2}{2 + 2} = -\beta$. *Thus, the Lorentz distance is* $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) = |\langle \mathbf{a}, \mathbf{b} \rangle| - \beta \approx \beta - \beta = 0$.

According to Theorem 6, it becomes apparent that mapping two sets of points from Euclidean space to hyperbolic space, each set separated by the same Euclidean distance, results in distorted Lorentz distances. Furthermore, if the Euclidean distance of a pair of points in $\mathcal{E}$ keeps constant and the pair moves further from the origin, this angle diminishes, leading the Lorentz distance between the hyperbolic embeddings from the Vanilla Hyperbolic Projection to approach zero. Unfortunately, embeddings with similar directional properties typically arise from proximal objects, and hence identifying the most similar objects among those that are close remains a critical challenge in representation learning.

B. Cosh Hyperbolic Projection

Motivation. The limitation of the Vanilla Hyperbolic Projection stems from its mapping approach, which essentially translates the magnitude of the original Euclidean vector, $\sqrt{\sum_{i=1}^{n} x_i^2}$, into a hyperbolic coordinate, $x_0 = \sqrt{\sum_{i=1}^{n} x_i^2 + \beta}$. This results in the mapped hyperbolic vectors becoming excessively large particularly when the magnitude is large, which tend to converge towards the hyperbolic asymptote line or the asymptotic hypersurface in higher dimensions as depicted by the gray dashed line in Figure 4a. Consequently, the Lorentz distance between these vectors tends to be zero. To address this issue, a nonlinear transformation of the magnitudes is required to prevent excessive enlargement of hyperbolic space magnitudes relative to their Euclidean counterparts. Our proposed solution employs the hyperbolic cosine function, defined as $\cosh(d) = \frac{e^d + e^{-d}}{2}$, to moderate this expansion.

Actually, the hyperbolic cosine function $\cosh(d)$ is derived from its definition via the hyperbolic angle. As illustrated in Figure 4b, considering the orange point in the hyperbola $x^2 - y^2 + 1 = 0$, its hyperbolic angle d is twice the area of

the hyperbolic sector formed by the vector at this point and the y-axis (the orange-shaded area). The hyperbolic cosine and sine values, $\cosh(d)$ and $\sinh(d)$, of this hyperbolic angle correspond to the x and y coordinates of this point, respectively. (Note: the coordinates are reversed here for consistency with Figure 4a to facilitate understanding). Utilizing the properties of the hyperbolic angle can significantly mitigate the issue of the Lorentz distance approaching zero due to the excessive magnitudes. To elucidate this further, we define the mapping process using the hyperbolic cosine function:

$$\phi_{cosh}(\mathbf{x}^{\mathcal{E}}) \rightarrow \mathbf{x}^{\mathcal{H}} : \quad x_i^{\mathcal{H}} = \begin{cases} \sqrt{\beta}\cosh(|x^{\mathcal{E}}|) & \text{if } i = 0, \\ x_{i-1}^{\mathcal{E}} \cdot k & others. \end{cases}$$

$$k = \frac{\sinh(|x^{\mathcal{E}}|)}{|x^{\mathcal{E}}|} \cdot \sqrt{\beta}, \qquad |x^{\mathcal{E}}| = \gamma_2\left(\sum_{i=0}^{n-1}(x_i^{\mathcal{E}})^2\right)$$

Here, $\gamma_c(x) = x^{\frac{1}{c}}$ represents a nonlinear compression of the Euclidean space distance. When $c = 2$, we have $\gamma_2(x) = \sqrt{x}$, and hence $|x^{\mathcal{E}}|$ represents the magnitude.

Effectiveness Analysis. To demonstrate that the Cosh Hyperbolic Projection can address the distance degradation issue, we first prove in Theorem 7 that the Lorentz distance in a 2-dimensional hyperbolic space remains non-diminished. Then, in Theorem 9, we extend this result to any $n+1$-dimensional space ($n > 2$), effectively reducing it to the 2-dimensional case and thus completing the proof.

Theorem 7. $\forall \mathbf{a} = (a_0, a_1),\ \mathbf{b} = (b_0, b_1) \in \mathcal{H}_2(\beta)$, *derived from the Cosh Hyperbolic Projection of the 1-dimensional values a, b in the Euclidean space $\mathcal{E}_1$, the Lorentz distance $\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b})$ depends on $|b - a|$ and non-diminished.*

Proof. According to Definition of ϕ_{cosh}, we have $a_0 = \sqrt{\beta}\cosh(a), a_1 = \sqrt{\beta}\sinh(a), b_0 = \sqrt{\beta}\cosh(b)$ and $b_1 = \sqrt{\beta}\sinh(b)$. Let $b > a$ and $d = b - a$, then we can conclude that:

$$\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) = |\beta\cosh(a)\cosh(b) - \beta\sinh(a)\sinh(b)| - \beta$$
$$= |\beta\cosh(a-b)| - \beta = \beta\left(\frac{e^d + e^{-d}}{2} - 1\right)$$

Theorem 8. *For any two vectors $\mathbf{a}, \mathbf{b} \in \mathcal{H}_{n+1}(\beta)$, if their components from the 1-st to the n-th dimension are linearly related, the $\mathbf{d}_{Lo}$ between them in $\mathcal{H}_{n+1}(\beta)$ is consistent with that of two corresponding vectors in $\mathcal{H}_2(\beta)$.*

Theorem 9. *Given any two vectors in a hyperbolic space $\mathcal{H}_{n+1}(\beta)$ with the projection ϕ_{cosh} on a Euclidean space $\mathcal{E}_n$, we can find a lower bound of the Lorentz distance between the two vectors, where the bound is non-dimished.*

Proof. For Any vector $\mathbf{b} = (p_0, p_1, ...p_n), \mathbf{c} = (q_0, q_1, ..., q_n) \in \mathcal{H}_{n+1}(\beta)$, We can always find a vector $\mathbf{a} \in \mathcal{H}(\beta)$ that is linearly related to $\mathbf{b}$ from the 1-st to n-th dimensions and identical to $\mathbf{c}$ in the 0-th dimension as follows:

$$\mathbf{a} = (q_0, kp_1, ..., kp_n) \quad k = \sqrt{\frac{\sum_{i=1}^n q_i^2}{\sum_{i=1}^n p_i^2}}$$

For the sake of simplicity, we can use $\mathbf{a}$ to re-express $\mathbf{b}$ and $\mathbf{c}$ as follows:

$$\mathbf{a} = (a_0, a_1, ..., a_n),\ \mathbf{b} = (a_0\mu_{\mathbf{a}}, \alpha a_1, ...\alpha a_n),\ \mu_{\mathbf{a}} = \frac{\sqrt{\sum_{i=1}^n a_i^2 \alpha^2 + \beta}}{\sqrt{\sum_{i=1}^n a_i^2 + \beta}}$$

$$\mathbf{c} = \left(a_0, a_1(1+\delta_1), ..., a_{n-1}(1+\delta_{n-1}), \sqrt{a_n^2 - \sum_{i=1}^{n-1}(\delta_i^2 + 2\delta_i)a_i^2}\right)$$

where we use p_i and q_i to inversely solve for the relevant parameters:

$$\alpha = \frac{1}{k}, \quad \delta_i = \frac{q_i}{kp_i} - 1, \quad a_0 = q_0, \quad a_i = kp_i$$

Next, we can compute $\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c})$ for the arbitrary vectors $\mathbf{b}, \mathbf{c}$ and $\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{a})$ for $\mathbf{a}, \mathbf{b}$ as follows:

$$\mathbf{d}_{Lo}(\mathbf{a}, \mathbf{b}) = |a_0^2\mu_{\mathbf{a}} - \alpha\sum_{i=1}^n a_i^2| - \beta$$

$$\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c}) = \left|a_0^2\mu_{\mathbf{a}} - \sum_{i=1}^{n-1}\alpha(1+\delta_i)a_i^2 - \alpha a_n^2\sqrt{1 - \sum_{i=1}^{n-1}(\delta_i^2 + 2\delta_i)\frac{a_i^2}{a_n^2}}\right| - \beta$$

Thus, we can demonstrate that $\mathbf{d}_{Lo(\mathbf{b},\mathbf{c})} \geq \mathbf{d}_{Lo(\mathbf{b},\mathbf{a})}$ as follows:

$$\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c}) - \mathbf{d}_{Lo}(\mathbf{b}, \mathbf{a}) = \alpha a_n^2\left(1 - \sqrt{1 - \sum_{i=1}^{n-1}\frac{a_i^2}{a_n^2}(\delta_i^2 + 2\delta_i)} - \sum_{i=1}^{n-1}\frac{a_i^2}{a_n^2}\delta_i\right)$$

$$\geq \alpha a_n^2\left(1 - \sqrt{1 - 2\sum_{i=1}^{n-1}\frac{a_i^2}{a_n^2}\delta_i} - \sum_{i=1}^{n-1}\frac{a_i^2}{a_n^2}\delta_i\right) \geq 0$$

*In addition, according to **Lemma 8**, $\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{a})$ cannot collapse to zero due to they are linearly related from the 1-th to the n-th dimension. Thus, if using ϕ_{cosh}, for any vector $\mathbf{b}, \mathbf{c}$ in $\mathcal{H}$, their $\mathbf{d}_{Lo}(\mathbf{b}, \mathbf{c})$ has its lower boundary and non-diminished.*

The advantage of this approach lies in the proportionality between the input to the hyperbolic cosine function and the area of the shaded region in Figure 4b. Specifically, the red Euclidean distance d and the hyperbolic sector (blue shaded area) are linearly correlated. As the Euclidean magnitude increases, the area of the hyperbolic sector also increases proportionally, causing the red angle in the figure to increase correspondingly. This proportional increase significantly mitigates the issue of the Lorentz distance approaching zero due to large magnitudes.

Complexity Analysis of Hyperbolic Projection. The complexity of our projection layer is linear to the dimensionality d of the vectors, denoted as $O(d)$. This time complexity is negligible compared to existing embedding models. For instance, the embedding complexities of Neutraj and TrajGAT are $O(Ld^2)$, which is significantly greater than $O(d)$, where L is the average number of trajectory points.

V. Dynamic Fusion Distance

In this section, we first introduce the variability of triangle constraints in different trajectory similarity functions in Section V-A. To enhance the robustness of trajectory embeddings by addressing this variability, we design a dynamic fusion distance module that integrates Euclidean distance and Lorentz distance in Section V-B.

A. Triangle Constraint Violation Variability

Given a specific trajectory dataset $\mathcal{T}$, the distances measured by different similarity functions, such as DTW and EDR, tolerate varying degrees of inequality constraint violations. To quantitatively analyze this variability, we use a violation flag to label whether a tuple distance follows the inequality constraint as follows:

$$\mathbf{TVF}(T_i, T_j, T_k) = \begin{cases} 1 & \text{if } \max(Sim[k|i,j], Sim[i|j,k], Sim[j|i,k]) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Sim[k|i,j] = f(T_i, T_j) - f(T_i, T_k) - f(T_j, T_k)$$

where $f(T_i, T_j)$ denotes a given trajectory similarity function.

Next, we propose two metrics, namely the *Ratio of Violation* and *Average Relative Violation*, to represent the ratio of violating constraints and the average degree of violation, respectively.

TABLE I: Constraint Variability on Chengdu and Porto

Dataset	Chengdu			Porto		
Method	DTW	SSPD	EDR	DTW	SSPD	EDR
RV	19.26%	28.61%	13.02%	25.28%	27.84%	16.66%
ARVS	0.1469	0.1252	0.2327	0.1590	0.1211	0.3179

Definition 10 (Ratio of Violation). *The ratio of violation for a given dataset $\mathcal{T}$ is denoted as $\boldsymbol{RV}(\mathcal{T})$, and is computed as follows:*
$$\boldsymbol{RV}(\mathcal{T}) = \sum_{T_i, T_j, T_k \in \mathcal{T}} \frac{\boldsymbol{TVF}(T_i, T_j, T_k)}{\mathcal{C}_N^3} \quad (N = |\mathcal{T}|)$$

Definition 11 (Average Relative Violation). *The average relative violation is denoted as $\boldsymbol{ARVS}(\mathcal{T})$, and is computed as follows:*
$$\boldsymbol{ARVS}(\mathcal{T}) = \frac{\sum_{T_i, T_j, T_k \in \mathcal{T}} \boldsymbol{RVS}(T_i, T_j, T_k) \boldsymbol{TVF}(T_i, T_j, T_k)}{\sum_{T_i, T_j, T_k \in \mathcal{T}} \boldsymbol{TVF}(T_i, T_j, T_k)}$$

$$\boldsymbol{RVS}(T_i, T_j, T_k) = \begin{cases} \frac{Sim[k|i,j]}{f(T_k, T_i) + f(T_k, T_j)} & \begin{array}{l} \text{if } f(T_i, T_j) \geq f(T_i, T_k), \\ f(T_i, T_j) \geq f(T_j, T_k) \end{array} \\ \frac{Sim[i|j,k]}{f(T_i, T_j) + f(T_i, T_k)} & \begin{array}{l} \text{if } f(T_j, T_k) \geq f(T_i, T_j), \\ f \geq f(T_i, T_k) \end{array} \\ \frac{Sim[j|i,k]}{f(T_j, T_k) + f(T_j, T_i)} & \begin{array}{l} \text{if } f(T_i, T_k) \geq f(T_i, T_j), \\ f(T_i, T_k) \geq f \end{array} \end{cases}$$

As reported in Table I, we computed the three metrics on the **Chengdu** and **Porto** datasets. The results demonstrate the variability of different similarity functions across these datasets. According to the definition of **RV**, approximately one-fifth of the trajectory triplets violate the triangle inequality constraint. Additionally, the **ARVS** indicates the challenge of accurately representing true similarity using embeddings in Euclidean space. To address this, leveraging hyperbolic space provides a more effective representation of these trajectories.

B. Distance Fusion

Motivation. Although some cases break the constraints, many data points still adhere to the inequality constraint. Therefore, relying solely on the Lorentz distance in the mapped hyperbolic space is ineffective; the Euclidean distance should also be considered. Consequently, given the variability of the constraints, we advocate for a dynamic method to identify trajectories constrained by the triangle inequality. This approach allows for the adaptive allocation of Lorentzian and Euclidean distances based on the varying magnitudes of the constraints.

To design an effective dynamic fusion procedure, it is crucial to avoid introducing additional computational complexity. For instance, a model that generates a weighting factor for the Lorentzian distance relative to the Euclidean distance for each pair of input trajectories would need to process every pair, resulting in quadratic time complexity. This would drastically increase the time required for similarity computation, contradicting the goal of enhancing speed using trajectory representation learning.

Instead, we propose a method that maintains the linear time complexity of the original pipeline. We achieve this by generating a Lorentz factor vector and a Euclidean factor vector for each trajectory. The proportion of the Lorentzian distance between any two trajectories is then determined by

TABLE II: Different Embedding Models and Network Types

Embedding Model	RNN	Attn	GNN	Preprocess
TrajGAT		✓	GAT	Quad-Tree
Neutraj	GRU			Grid-Cell
Traj2SimVec	LSTM			Sub-Trajectory
ST2Vec	LSTM	✓		

the ratio of the inner product of their Lorentz factor vectors to the inner product of their Euclidean factor vectors as follows:
$$V_a^{Lo}, V_a^{Eu} = \mathbf{Encoder}(T_a) \qquad V_b^{Lo}, V_b^{Eu} = \mathbf{Encoder}(T_b)$$
$$\alpha_{a,b}^{Lo} = \frac{V_a^{Lo} \cdot V_b^{Lo}}{V_a^{Lo} \cdot V_b^{Lo} + V_a^{Eu} \cdot V_b^{Eu}}$$

where the **Encoder** is a sequence-to-vector model that accepts trajectory sequences and generates a vector. The first half of the vector, denoted as V_*^{Lo}, represents the Lorentz factor embedding, while the second half, denoted as V_*^{Eu}, represents the Euclidean factor embedding. The inner product of these vectors between different trajectories produces the Lorentz proportion α^{Lo}, which balances the weights of the Lorentzian and Euclidean distances. The process of generating a dynamic fused distance from trajectory embeddings is as follows:
$$\mathbf{d}_{Fu}(T_a, T_b) = \alpha_{a,b}^{Lo} \cdot \mathbf{d}_{Lo}(\mathbf{x}_a^{\mathcal{H}}, \mathbf{x}_b^{\mathcal{H}}) + (1 - \alpha_{a,b}^{Lo}) \cdot \mathbf{d}_{Eu}(\mathbf{x}_a^{\mathcal{E}}, \mathbf{x}_b^{\mathcal{E}})$$

As for the model selection of **Encoder**, we experimented with several common Seq2Vec models. To accommodate various trajectory embedding models without introducing additional time complexity, we selected the LSTM, which has the lowest complexity and scales linearly with trajectory length. Since this model is straightforward and not the primary focus of this paper, we will not elaborate further.

Complexity Analysis. The Lorentz distance $\mathbf{d}_{Lo}$, similar to the Euclidean distance $\mathbf{d}_{Eu}$, is calculated as the dot product between vectors, with a computational complexity of $O(d)$, where d is the dimensionality. Therefore, the overall complexity of computing the fused distance $\mathbf{d}_{Fu}$ remains $O(d)$.

VI. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate our proposed framework **LH**-plugin. In particular, we focus on answering the following questions.

- **Q1**: Can the **LH**-plugin address the triangle inequality issue in trajectory embeddings? **Q1.1 (Section VI-B)**: Do embeddings generated with **LH**-plugin significantly enhance accuracy in trajectory similarity retrieval? **Q1.2 (Section VI-C)**: Does **LH**-plugin bring the triangular relationships between embeddings closer to the ground truth, thereby relaxing the triangle inequality constraint?
- **Q2**: How does the **LH**-plugin influence the training and inference efficiency of embedding models? **Q2.1 (Section VI-D)**: Does the increased inference latency affect the efficiency of similarity trajectory retrieval? **Q2.2 (Section VI-E)**: What is the scalability of the **LH**-plugin framework for training embedding models?

- **Q3 (Section VI-F)**: Is the **LH**-plugin framework robust in enhancing similarity retrieval effectiveness across varying model effectiveness?
- **Q4 (Section VI-G)**: Quantitative analysis of our three principal contributions in the **LH**-plugin.
- **Q5 (Section VI-H)**: Quantitative analysis of hyper-parameters.

A. Experimental Setup

Datasets. We select several datasets commonly used by trajectory similarity methods for experiments.

(1) **Chengdu** [36]: The Chengdu dataset comprises 5.22 million ride-hailing trajectories, situated in Chengdu, China, provided by DiDi Inc. It includes a total of 1.05 billion coordinate points, with an average trajectory length of 32.5 km, which is commonly used for spatial similarity measures.

(2) **Porto** [37]: The Porto dataset consists of 1.49 million taxi route trajectories, located in Porto, Portugal, and is open source on Kaggle. It encompasses over 80 million points, with an average trajectory length of 6.12 km, which is commonly used for spatial similarity measures.

(3) **T-Drive** [38]: The T-Drive is a spatio-temporal trajectory dataset that includes trajectories from 10,357 taxis over one week. It consists of approximately 15 million data points, covering a total distance of 9 million kilometers.

Embedding models. To evaluate the adaptability of the **LH**-plugin across various scenarios, we select four advanced trajectory embedding models as follows:

- **Neutraj** [33] is a spatial trajectory embedding model that utilizes LSTM networks and a gridded neighbor table to store and update proximity information between trajectory points.
- **TrajGAT** [22] is designed for handling spatial similarity computation in long trajectories. In particular, it employs a pre-constructed quadtree topology and integrates graph attention networks to save GPU memory and enhance performance for long trajectories.
- **Traj2SimVec** [34] is another representation learning model for spatial trajectory similarity. It introduces innovative sub-trajectory and trajectory matching information, achieving high performance in pure spatial tasks.
- **ST2Vec** [35] addresses similarity issues in spatio-temporal trajectories by incorporating time encoding into trajectory encoding and combining them through co-attention mechanisms.

In summary, the select models represent the most commonly employed neural networks for trajectory embedding, including RNN variants, attention mechanisms, and graph neural networks. Additionally, these models utilize a diverse array of trajectory preprocessing methods, such as sub-trajectory segmentation, grid-cell partitioning, and tree-based indexing structures as shown in Table II.

Evaluation Metrics. For the spatial similarity computation, we select **DTW**, **SSPD**, and **EDR** as benchmark similarity measures, which do not satisfy the triangle inequality constraint. For spatio-temporal trajectory similarity, we employ

TP, **DITA**, and **secret Fréchet** distance to assess the effectiveness of the **LH**-plugin.

To gauge accuracy, we follow established practices [22], [33], [35], using hit rate $HR@\alpha$ and NDCG [39] as evaluation metrics, where higher values indicate more precise trajectory similarity computation. Here, HR@α represents the hit rate of top α, indicating the proportion of the most similar top α in the embedded results relative to the most similar top α in the ground truth. While, NDCG [39] reflects the similarity between the ranking of the embedded results and the ranking of the ground truth, and is commonly used to measure the retrieval quality in search and recommendation systems.

Environment Settings. We implement and incorporate the **LH**-plugin based on PyTorch 2.0.1 into these original trajectory similarity models. Additionally, the experiments are conducted on Ubuntu 18.04.6 OS, utilizing the Intel Xeon Gold 6248 CPU and the NVIDIA Tesla V100 GPU, yielding the following experimental results.

B. Effectiveness Evaluation (Q1.1)

Spatial Trajectory. To validate the enhancement of the **LH**-plugin on spatial trajectory similarity over the original models, we compare the accuracy of the models before and after integrating the **LH**-plugin, demonstrating its effectiveness in trajectory similarity learning. The results are presented in Table III, where the hit rates are displayed as percentages. The results unequivocally demonstrate that:

(1) Almost universally across all spatial trajectory similarity methods, datasets, and models, there has been a significant improvement in hit rate accuracy after integrating the **LH**-plugin. Particularly noteworthy are **Neutraj** and **Traj2SimVec**, where, in the Chengdu dataset, both methods saw an approximate 30% increase in accuracy on EDR distance for instance. This clearly demonstrates that the **LH**-plugin effectively addresses the triangle inequality problem in spatial trajectory similarity representation learning, thereby enhancing the model's effectiveness in similarity computation.

(2) Models with higher original precision demonstrate a more substantial proportional improvement after integrating the **LH**-plugin (e.g., the DTW results under Neutraj and the SSPD results under Traj2SimVec). This improvement is attributed to the original models already nearing their performance limits in Euclidean space. Thus, the introduction of hyperbolic space significantly enhances the models' precision. Conversely, minimal improvements are observed in models with relatively low initial precision, such as **TrajGAT**'s calculation of EDR on the Porto dataset. In such instances, where the original precision is particularly low, the influence of the triangle inequality constraint on similarity accuracy is less pronounced, resulting in less significant precision gains post **LH**-plugin integration. This observation is consistent with Figure 1, where it is stated that *the triangle inequality contradiction has a more pronounced impact on methods with higher precision.*

(3) Cross-comparison among various similarity methods indicates that the DTW distance exhibits the most significant improvement after integrating the **LH**-plugin. This enhance-

TABLE III: Accuracy results on the Chengdu and Porto datasets

base model	sim	plugin	Chengdu					sim	plugin	Porto				
			HR@5(%)	HR@10(%)	HR@50(%)	NDCG10	NDCG50			HR@5(%)	HR@10(%)	HR@50(%)	NDCG10	NDCG50
Neutraj	DTW	Original	53.54	60.36	75.28	0.7526	0.8513	DTW	Original	36.8	41.82	52.09	0.4487	0.5452
		LH-plugin	68.33	73.99	84.11	0.7929	0.8790		**LH**-plugin	39.79	44.31	54.41	0.4916	0.5978
		%Increase	+27.62%	+22.58%	+11.73%	+5.35%	+3.25%		%Increase	+8.13%	+5.95%	+4.45%	+9.56%	+9.65%
	SSPD	Original	49.80	54.48	66.81	0.6112	0.7293	SSPD	Original	23.61	27.12	38.14	0.3046	0.4125
		LH-plugin	57.42	63.51	78.52	0.6484	0.8028		**LH**-plugin	27.26	31.30	43.27	0.3611	0.4943
		%Increase	+15.30%	+16.57%	+17.53%	+6.09%	+10.08%		%Increase	+15.46%	+15.41%	+13.45%	+18.55%	+19.83%
	EDR	Original	8.81	10.71	13.21	0.1373	0.1730	EDR	Original	9.54	10.23	11.60	0.1103	0.1268
		LH-plugin	12.06	14.46	1.48	0.1546	0.1968		**LH**-plugin	11.48	12.18	13.83	0.1539	0.1579
		%Increase	+36.89%	+35.01%	+32.32%	+12.60%	+13.76%		%Increase	+20.34%	+19.06%	+19.22%	+39.53%	+24.53%
TrajGAT	DTW	Original	27.07	30.31	36.69	0.3526	0.4843	DTW	Original	14.42	18.39	33.71	0.1422	0.2775
		LH-plugin	39.92	41.95	47.32	0.4877	0.5969		**LH**-plugin	17.24	21.52	35.66	0.1898	0.3605
		%Increase	+47.47%	+38.40%	+28.97%	+38.32%	+23.25%		%Increase	+19.56%	+17.02%	+5.78%	+33.47%	+29.91%
	SSPD	Original	38.61	43.73	55.37	0.4753	0.6033	SSPD	Original	20.07	25.35	37.15	0.2458	0.3515
		LH-plugin	45.1	50.33	61.74	0.5455	0.6832		**LH**-plugin	20.5	25.41	40.77	0.2562	0.3641
		%Increase	+16.81%	+15.09%	+11.50%	+14.77%	+13.24%		%Increase	+2.14%	+0.24%	+9.83%	+4.23%	+3.58%
	EDR	Original	5.74	6.63	8.19	0.0473	0.0737	EDR	Original	7.42	8.03	9.78	0.0775	0.0861
		LH-plugin	6.69	7.49	8.65	0.0725	0.0817		**LH**-plugin	7.46	8.10	9.81	0.0783	0.0892
		%Increase	+16.55%	+12.97%	+5.62%	+53.28%	+10.85%		%Increase	+0.54%	+0.87%	+0.31%	+1.03%	+3.60%
Traj2SimVec	DTW	Original	44.87	50.38	65.41	0.5071	0.6559	DTW	Original	19.94	24.31	37.45	0.2771	0.4226
		LH-plugin	47.11	53.3	68.4	0.5545	0.6990		**LH**-plugin	28.91	33.68	49.91	0.3239	0.4753
		%Increase	+4.99%	+5.80%	+4.57%	+9.35%	+6.57%		%Increase	+44.98%	+38.54%	+33.27%	+16.89%	+12.47%
	SSPD	Original	47.68	52.64	66.05	0.6092	0.7348	SSPD	Original	32.00	36.85	51.25	0.4374	0.5893
		LH-plugin	57.37	62.99	74.04	0.6818	0.7856		**LH**-plugin	38.29	44.81	60.86	0.4694	0.6139
		%Increase	+20.32%	+19.66%	+12.10%	+11.92%	+6.91%		%Increase	+19.66%	+21.60%	+18.75%	+7.32%	+4.17%
	EDR	Original	11.9	14.51	17.56	0.1746	0.2126	EDR	Original	20.33	20.52	17.54	0.2116	0.2083
		LH-plugin	15.89	19.68	25.08	0.1979	0.2312		**LH**-plugin	20.71	21.36	19.00	0.2230	0.2133
		%Increase	+33.53%	+35.63%	+42.82%	+13.34%	+8.75%		%Increase	+1.87%	+4.09%	+8.32%	+5.39%	+2.40%

TABLE IV: Accuracy results of ST2Vec on the T-Drive

sim	plugin	HR@5	HR@10	HR@50	NDCG10	NDCG50
TP	Original	40.21	46.61	58.54	0.4938	0.5952
	LH-plugin	47.46	53.62	64.32	0.5427	0.6578
	%Increase	+18.03%	+15.04%	+9.87%	+9.90%	+10.52%
DITA	Original	40.96	46.76	54.72	0.4923	0.5939
	LH-plugin	49.06	54.30	62.03	0.5896	0.6877
	%Increase	+19.78%	+16.12%	+13.36%	+19.76%	+15.79%
discret fréchet	Original	42.50	49.52	61.97	0.5306	0.6310
	LH-plugin	46.72	53.02	64.74	0.5680	0.6712
	%Increase	+9.91%	+7.07%	+4.47%	+7.05%	+6.37%

ment suggests that DTW distance is particularly impacted by the triangle inequality constraint.

Spatio-temporal Trajectory. To complement the evaluation of **LH**-plugin's performance on spatio-temporal trajectories, we contrast the results of the spatio-temporal similarity computation model **ST2Vec** before and after the addition of **LH**-plugin, as shown in Table IV. It is evident that:

(1) After integrating the **LH**-plugin, **ST2Vec** shows significant improvements in calculations of TP, DITA, and discrete Fréchet distances, with DITA distance notably improving by nearly 20%. This clearly demonstrates **LH**-plugin's capability in addressing the triangle inequality constraints inherent in spatio-temporal trajectory similarity learning. It also indirectly highlights that triangle inequality constraints indeed pose a bottleneck for similarity computations in representation learning models.

C. Triangle Inequality Violation Evaluation (**Q1.2**)

In this section, we empirically validate whether the inclusion of **LH**-plugin alleviates the triangle inequality constraint issue
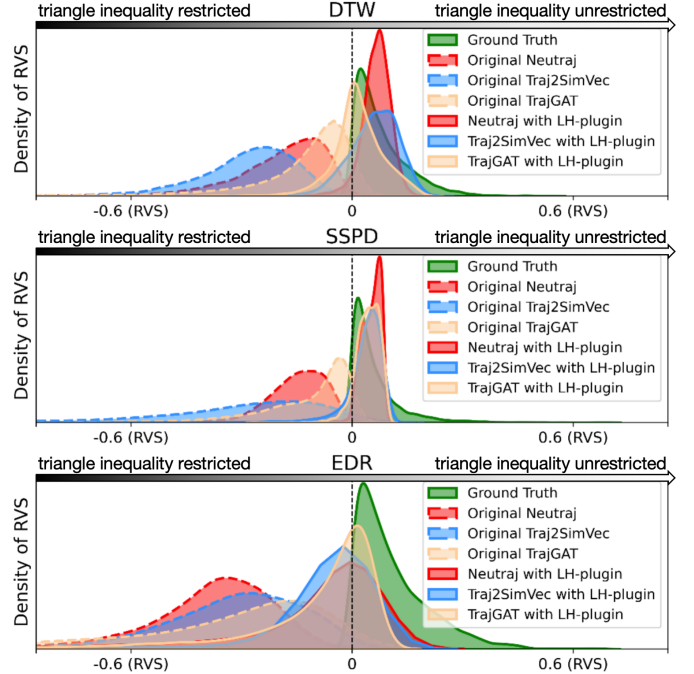

Fig. 5: RVS Distribution Comparison: Capabilities of Unconstraint from the Triangle Inequality

inherent in the original models. Specifically, we randomly select 1 million trajectory triplets from the Chengdu dataset that exhibit conflicts between similarity metrics and the triangle inequality (to facilitate the observation of results, non-conflicting triplets are not included to avoid diluting the outcomes; the proportion of conflicts can be seen in Table
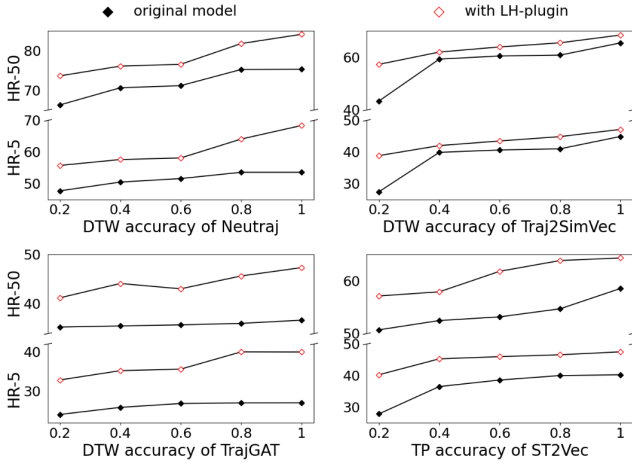
Fig. 6: Scalability Comparison of **LH**-plugin

Fig. 7: Robustness Comparison of **LH**-plugin

I under the **ARVS**). We record the Euclidean distances in the original model and the Fusion Distances in the model with the **LH**-plugin. Then, we calculate the **RVS** (defined in Section V-A) for the Similarity Ground Truth, Euclidean distance, and Fusion distance of these triplets respectively.

Remark. For the selected triplets that violate the triangle inequality, the Ground Truth **RVS** values should all exceed zero, indicating that the metric encounters a conflict with such triplets, where higher values signify a more pronounced conflict. In terms of model learning using Euclidean or Fusion distances, a positive **RVS** demonstrates that the model can generate distance triples that are not restricted by the triangle inequality. Conversely, a negative **RVS** indicates adherence to the triangle inequality constraint. Furthermore, the closer the distribution of **RVS** values aligns with the Ground Truth, the greater the precision achieved by the model.

Here, we present the probability density of **RVS** under different methods, as shown in Figure 5, from which we draw the following conclusions:

(1) The Euclidean embeddings generated by the original models (i.e., the part of the dashed line) fall almost entirely on the negative half-axis, representing the original models are strictly constrained by the triangle inequality. However, the Ground Truth values are almost all located on the positive half-axis, such conflicts of triangle inequality constraint in DTW, SSPD, and EDR significantly impact the learning accuracy in the Euclidean space.

(2) After incorporating the **LH**-plugin, the distribution of **RVS** in different models shows a trend towards the positive half-axis, thus becoming closer to the ground truth. This suggests that the inclusion of the **LH**-plugin indeed addresses the issue of the triangle inequality constraint in representation learning. For instance, considering the **Neutraj** method in the context of DTW distance, the negative instances in its **RVS** distribution nearly disappear after the integration of the **LH**-plugin, which corresponds to the substantial improvement observed in its accuracy results (HR@5 increased from 53.54% to 68.33%, as seen in Table III).
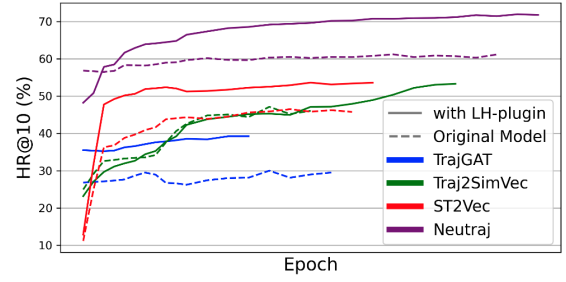
D. Efficiency Evaluation (*Q2.1*)

In this section, we assess whether the **LH**-plugin significantly increases inference latency, potentially impacting the trajectory retrieval process. The integration of new modules into existing representation learning frameworks is likely to extend the inference time. Our goal is to determine if the trajectory retrieval efficiency becomes unacceptably compromised after incorporating the **LH**-plugin compared to the baseline models. Prior studies indicate that trajectory embedding methods generally outperform non-learning-based rule optimization approaches (e.g., [16], [40]) due to the ability to pre-embed a large number of trajectories offline. We conduct our efficiency evaluation based on this pre-embedding strategy for trajectories, comparing the end-to-end latency of online similar trajectory retrieval before and after the integration of the **LH**-plugin. As detailed in Table V, the number of trajectories indicates the size of the trajectory database. While integrating the **LH**-plugin does result in increased retrieval latency, this increment is marginal—less than 0.05%—especially in real-world datasets that scale to millions of trajectories, rendering it negligible in comparison to the overall retrieval time.

E. Scalability Evaluation (*Q2.2*)

In this section, we aim to examine whether the introduction of the **LH**-plugin affects the scalability of the original models. We compare the impact of different training data sizes on accuracy. Specifically, we sample 20%, 40%, 60%, 80%, and 100% from the Chengdu dataset (TDrive for **ST2Vec**) to train different trajectory similarity models. The results, as depicted in Figure 6, lead us to conclude:
(1) With an increase in training data, there is a significant improvement in trajectory similarity accuracy across different models.
(2) The inclusion of the **LH**-plugin results in noticeable enhancements in accuracy across all data sizes, with the accuracy curves before and after adding the **LH**-plugin exhibiting similar shapes. Thus, it can be concluded that the **LH**-plugin can significantly improve the accuracy of trajectory similarity models regardless of the training data size.

F. Robustness Evaluation (*Q3*)

In this section, we aim to evaluate whether the **LH**-plugin is robust to enhance the similarity retrieval process. In the aforementioned experiments, we observe an intriguing phenomenon: the training curves with the **LH**-plugin are more

TABLE V: The Additional Latency Introduced by **LH**-plugin in Trajectory Retrieval

Trajectory number	TrajGAT			Neutraj			Traj2SimVec			ST2Vec		
	Original	with LH-plugin	%Increase	Original	with LH-plugin	%Increase	Original	with LH-plugin	%Increase	Original	with LH-plugin	%Increase
10k	0.447	0.455	1.80%	0.208	0.213	2.63%	0.186	0.191	2.68%	0.207	0.213	2.97%
100k	1.864	1.872	0.43%	1.626	1.632	0.34%	1.608	1.613	0.31%	1.625	1.631	0.38%
1m	15.601	15.609	0.05%	15.342	15.347	0.04%	15.336	15.341	0.03%	15.329	15.335	0.04%

stable compared to the original model. If there are significant fluctuations in accuracy in the original model, the inclusion of the **LH**-plugin results in a smoother curve. We illustrate this with the training curves using DTW distance as an example in Figure 7, thus we get the following conclusion:
(1) For the same original model, the smoother accuracy curve with the **LH**-plugin helps avoid premature convergence due to fluctuations, allowing for more thorough training.
(2) Additionally, the smoother accuracy curve with the **LH**-plugin brings another advantage: rapid convergence after reaching a plateau, which is particularly pronounced in the **TrajGAT** model.

G. Ablation Evaluation (*Q4*)

In this section, we aim to quantitatively to validate the effectiveness of our three principal contributions: **1) the Lorentz distance in Hyperbolic space**, **2) the Cosh Projection to Hyperbolic space**, and **3) the Dynamic Fusion Distance**, we structure the **LH**-plugin around these elements and conduct a series of incremental ablation experiments. The detailed results are presented in Table VI. Below, we discuss the ablation design and findings corresponding to these three key aspects:
(1) The Lorentz Distance in Hyperbolic Space: We assess whether integrating the Lorentz distance within hyperbolic space could substantially improve model accuracy by overcoming the limitations posed by the triangle inequality. This involves comparing the accuracy of the traditional Euclidean space representation learning ("original" row in Table VI) with that of the hyperbolic space representation using the Lorentz distance via Vanilla Projection ("lh-vanilla" row in Table VI). The findings indicate a notable increase in similarity accuracy across most metrics and models when switching to the Lorentz distance. Therefore, the substantial improvements noted in most models underscore the efficacy of the Lorentz metric in capturing deeper relational nuances that Euclidean metrics often overlook. However, the marginal improvements in some models suggest that the integration of hyperbolic space may require careful calibration to fully realize its potential, particularly in preserving semantic integrity during projection.
(2) The Cosh Projection to Hyperbolic Space: This part of the study aims to determine whether employing the Cosh Projection instead of the Vanilla Projection for mapping Euclidean embeddings into hyperbolic space could address the issue of Lorentz distance collapsing to zero with large-norm Euclidean embeddings (as outlined in Section 4.1), thereby significantly enhancing model accuracy. We compare the accuracy differences between the Vanilla Projection ("lh-vanilla" in Table VI) and the Cosh Projection ("lh-cosh" in Table VI). The results decisively demonstrate that switching to Cosh Pro-
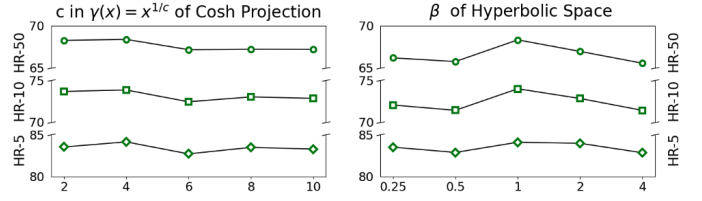

Fig. 8: Hyper-parameter Evaluation

jection substantially improved similarity accuracy. Therefore, the Cosh Projection emerges as a crucial enhancement over the Vanilla Projection method, particularly in addressing the challenge of distance diminishment with increasing Euclidean norms. This projection method stabilizes the expansion of trajectory embeddings into hyperbolic space, thereby maintaining consistent representational fidelity across varying magnitudes of embeddings. The pronounced improvements with the "lh-cosh" setup in our experiments demonstrate the critical role of this method in maintaining robust distance metrics, even under scaling challenges posed by larger-norm embeddings.
(3) The Dynamic Fusion Distance: Finally, to explore whether integrating the Dynamic Fusion Distance enhances model accuracy, we combine the Lorentz distance obtained via Cosh Projection with the conventional Euclidean distance using the Dynamic Fusion Module (described in Section V). This configuration represents the comprehensive application of the **LH**-plugin ("fusion-dist" in Table VI). When compared with results from solely using Euclidean distance ("original" in Table VI) and only Lorentz distance ("lh-cosh" in Table VI), the integration of Dynamic Fusion Distance demonstrates significant accuracy improvements across various models and metrics, validating the efficacy of this approach. The fusion approach not only accommodates the variability in triangle inequality constraints but also enhances the model's flexibility to accurately reflect the true similarity across diverse trajectory datasets.

H. Hyper-Parameter Evaluation (*Q5*)

In the **LH**-plugin, we have set the following two adjustable hyper-parameters, β and c.
The parameter β is a hyper-parameter in the hyperbolic space $\mathcal{H}(\beta) = \{\mathbf{a} \in \mathbb{R}^2 : \langle \mathbf{a}, \mathbf{a} \rangle = -\beta, \ a_0 \geq \sqrt{\beta}\}$, which controls the shape of the hyperbolic space. This parameter significantly influences the shape of the hyperbolic space near the origin, thereby affecting the accuracy of Lorentz distance when the norm of Euclidean embedding is small.
The parameter c in $\gamma(x) = x^{1/c}$, as mentioned in Section IV-B, is a non-linear function introduced to adjust the norm of Euclidean embeddings. When c is set to a larger value, it can alleviate the issue of large norms in Euclidean space to

TABLE VI: Ablation Evaluation

		DTW				SSPD				EDR			
		original	lh-vanilla	lh-cosh	fusion-dist	original	lh-vanilla	lh-cosh	fusion-dist	original	lh-vanilla	lh-cosh	fusion-dist
	HR@5	53.54	65.03	67.07	68.33	49.80	50.05	57.66	58.25	8.81	8.07	11.87	12.06
Neutraj	HR@10	60.36	71.67	73.54	73.99	54.48	55.18	63.03	65.72	10.71	10.79	13.51	14.46
	HR@50	75.28	83.64	83.87	84.11	66.81	67.52	78.56	80.32	13.21	14.67	15.89	17.48
	HR@5	27.07	36.85	39.11	39.92	38.61	40.78	42.55	45.10	5.74	6.02	6.51	6.69
TrajGAT	HR@10	30.31	38.96	41.79	41.95	43.73	46.44	48.89	50.33	6.63	6.72	7.05	7.49
	HR@50	36.69	42.31	47.06	47.32	55.37	58.11	60.81	61.74	8.19	8.15	8.33	8.65
	HR@5	44.87	45.66	46.49	47.11	47.68	51.61	56.02	57.37	11.90	11.98	13.45	15.89
Traj2SimVec	HR@10	50.38	51.23	51.99	53.30	52.64	55.73	60.35	62.99	14.51	15.11	16.72	19.68
	HR@50	65.41	65.97	66.89	68.40	66.05	68.12	71.86	74.04	17.56	17.99	20.14	25.08

some extent, but an excessively large c might lead to a loss of digital accuracy due to its actual implementation using finite Taylor series expansion.

The experimental results for these two hyper-parameters are shown in the Figure 8. Based on these results, we ultimately selected $c = 4$ and $\beta = 1$.

VII. RELATED WORK

A. Trajectory Representation Learning for Similarity Computation

Trajectory similarity is a fundamental area of research within trajectory data management and serves as a critical precursor to various downstream tasks. The computational complexity of traditional distance functions [15], [17]–[19], [41], [42] —is super-quadratic during both the computation and retrieval phases. Particularly in the scenarios involving similarity searches, it is necessary to perform pairwise similarity calculations for every trajectory in the database, which entails exhaustive pairwise comparisons between each pair of points in the trajectories.

The introduction of trajectory representation learning for similarity computation has advanced the field by vectorizing trajectories, allowing the use of vector distances to measure similarity between trajectories. This approach addresses the combinatorial complexity encountered during retrieval and computation. A pioneering work in this field is Li et al. [43], which first proposed a framework that uses deep learning networks to vectorize variable-length trajectories. It measures the similarity between trajectories by distances among vectors, employs methods like locality-sensitive hashing for preliminary screening of approximate trajectories, and subsequently computes the actual similarity distances. Further advancements were made by Yao et al. [33], which improved upon earlier efforts by splitting the trajectory space into grids and aggregating neighboring information. However, they did not capitalize on the information of sub-trajectories on the trajectory. To address this issue, Zhang et al. [34] introduced sub-trajectory information to enhance semantic similarity between trajectories based on sub-trajectory similarities. Nevertheless, the trajectory lengths processed by these models are often much shorter than those encountered in real-world data. Therefore, Yao et al. [22] focused on long trajectory issues in real-world data by employing quadtree structured encoding of trajectory

space and utilizing graph attention networks for trajectory embedding. Another approach to this issue involves simplifying trajectories by incorporating temporal dimension information. For example, Fang et al. [35] incorporated the time dimension into similarity representation learning, integrating time and space dimensions through co-attention mechanisms to jointly compute similarity.

Despite these advances, nearly all research in this field has focused on refining network models to map trajectories into Euclidean space. However, *a significant gap remains in addressing the inherent contradictions between Euclidean space and similarity metrics, such as the triangle inequality, which have yet to be thoroughly explored.*

B. Representation Learning in Hyperbolic Space

Hyperbolic space possesses a natural advantage in embedding representations with high semantic richness [44], particularly excelling at handling hierarchical dependencies such as tree-like structures. By hyperbolic space, extensive and intricate dependencies in the real world can be decomposed into tree-shaped forms, due to their potential tree-like structural features [45], [46]. Inspired by this finding, the study of learning hyperbolic embeddings has garnered significant attention in various domains of representation learning, primarily dividing into two orientations:

In the first field, some research endeavors have attempted to reconstruct existing neural network models within hyperbolic space to achieve end-to-end fully hyperbolic neural network representation learning. Ganea et al. [32], pioneers in this area, employ the formalism of Möbius gyrovector spaces with Riemannian geometry to derive various deep learning architectures in hyperbolic space, such as multinomial logistic regression and feed-forward networks. However, due to the fact that GPU operators are not optimized for computations related to hyperbolic space, the training and inference process of hyperbolic neural networks are evidently inefficient. Additionally, migrating existing representation learning methods from Euclidean space to hyperbolic space may have unintended side effects on their effectiveness.

Therefore, the other orientation of research opts for a hybrid network structure instead of a purely hyperbolic neural network pipeline, only in the final layer or a specific part, introducing the hyperbolic model to map Euclidean space embeddings into hyperbolic space. These approaches [47]–

[49], much similar to ours, using the hybrid network and resulting in an incremental plugin, can be utilized to improve existing Euclidean space methods. In this orientation, *previous research has rarely directly addressed the intuitive conflicts between representation learning and metrics space, such as triangle inequality.*

VIII. Conclusion

In this paper, we conduct an in-depth study of the triangle inequality issue in trajectory similarity representation learning, specifically the conflict between the triangle inequality constraint in Euclidean space and trajectory similarity metrics. To address this, we propose the **LH**-plugin, a model-agnostic plugin for any trajectory similarity representation learning model. This plugin utilizes the Lorentz distance in hyperbolic space to circumvent the triangle inequality constraint present in Euclidean space. It employs the Cosh function to optimize the projection from Euclidean space to hyperbolic space and dynamically adjusts the fusion of Lorentz distance and Euclidean distance based on different trajectory combinations using fusion distance. Extensive experiments were conducted on real-world datasets using various mainstream trajectory similarity representation learning models, validating the effectiveness of the **LH**-plugin.

REFERENCES

[1] W. Chen, Y. Liang, Y. Zhu, Y. Chang, K. Luo, H. Wen, L. Li, Y. Yu, Q. Wen, C. Chen, K. Zheng, Y. Gao, X. Zhou, and Y. Zheng, "Deep learning for trajectory data management and mining: A survey and beyond," *arXiv*, vol. abs/2403.14151, 2024.

[2] S. Wang, Z. Bao, J. S. Culpepper, and G. Cong, "A survey on trajectory data management, analytics, and learning," *CSUR*, vol. 54, pp. 1–36, 2021.

[3] C. L. da Silva, L. M. Petry, and V. Bogorny, "A survey and comparison of trajectory classification methods," in *BRACIS*, 2019, pp. 788–793.

[4] Z. Feng and Y. Zhu, "A survey on trajectory data mining: Techniques and applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.

[5] S. Wang, J. Cao, and S. Y. Philip, "Deep learning for spatio-temporal data mining: A survey," *TKDE*, vol. 34, no. 8, pp. 3681–3700, 2020.

[6] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *IJCNN*, 2017, pp. 3880–3887.

[7] X. Olive, L. Basora, B. Viry, and R. Alligier, "Deep trajectory clustering with autoencoders," in *ICRAT*, 2020.

[8] Z. Fang, Y. Du, L. Chen, Y. Hu, Y. Gao, and G. Chen, "E 2 dtc: An end to end deep trajectory clustering framework via self-training," in *ICDE*, 2021, pp. 696–707.

[9] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," *arXiv*, 2018.

[10] Z. Fang, S. Gong, L. Chen, J. Xu, Y. Gao, and C. S. Jensen, "Ghost: A general framework for high-performance online similarity queries over distributed trajectory streams," *POMD*, pp. 1–25, 2023.

[11] Y. Su, D. Yao, X. Zhou, Y. Zhang, Y. Fan, L. Bai, and J. Bi, "Tripsafe: Retrieving safety-related abnormal trips in real-time with trajectory data," in *SIGIR*, 2023, pp. 2446–2450.

[12] R. Song, W. Sun, B. Zheng, and Y. Zheng, "PRESS: A novel framework of trajectory compression in road networks," *VLDB*, pp. 661–672, 2014.

[13] M. Shaygan, C. Meese, W. Li, X. G. Zhao, and M. Nejad, "Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities," *TR-C*, vol. 145, p. 103921, 2022.

[14] H. Yuan and G. Li, "A survey of traffic prediction: from spatio-temporal data to intelligent transportation," *DSE*, vol. 6, no. 1, pp. 63–85, 2021.

[15] B. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *ICDE*, 1998, pp. 201–208.

[16] P. K. Agarwal, K. Fox, J. Pan, and R. Ying, "Approximating dynamic time warping and edit distance for a pair of point sequences," in *SoCG*, vol. 51, 2016, pp. 6:1–6:16.

[17] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, "Review and perspective for distance based trajectory clustering," *CoRR*, vol. abs/1508.04904, 2015.

[18] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *SIGMOD*, 2005, pp. 491–502.

[19] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *TITS*, vol. 11, no. 3, pp. 647–657, 2010.

[20] P. Yang, H. Wang, D. Lian, Y. Zhang, L. Qin, and W. Zhang, "TMN: trajectory matching networks for predicting similarity," in *ICDE*, 2022, pp. 1700–1713.

[21] Y. Chang, J. Qi, Y. Liang, and E. Tanin, "Contrastive trajectory similarity learning with dual-feature attention," in *ICDE*, 2023, pp. 2933–2945.

[22] D. Yao, H. Hu, L. Du, G. Cong, S. Han, and J. Bi, "Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation," in *KDD*, 2022, pp. 2275–2285.

[23] S. B. Yang, J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Lightpath: Lightweight and scalable path representation learning," in *SIGKDD*, 2023, pp. 2999–3010.

[24] S. Zhou, J. Li, H. Wang, S. Shang, and P. Han, "Grlstm: trajectory similarity computation with graph-based residual lstm," in *AAAI*, vol. 37, no. 4, 2023, pp. 4972–4980.

[25] P. Han, J. Wang, D. Yao, S. Shang, and X. Zhang, "A graph-based approach for trajectory similarity computation in spatial networks," in *KDD*, 2021, pp. 556–564.

[26] Euclid, *The Elements*. Dover Publications, 1956, originally published ca. 300 B.C.

[27] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. McGraw-Hill, 1976.

[28] R. M. Wald, *General Relativity*, 1st ed. University of Chicago Press, 1984.

[29] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *WWW*, 2017, pp. 193–201.

[30] C. Xu and M. Wu, "Learning feature interactions with lorentzian factorization machine," in *AAAI*, vol. 34, no. 04, 2020, pp. 6470–6477.

[31] M. Nickel and D. Kiela, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," in *ICML*. PMLR, 2018, pp. 3779–3788.

[32] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," *NeurIPS*, vol. 31, 2018.

[33] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *ICDE*, 2019, pp. 1358–1369.

[34] H. Zhang, X. Zhang, Q. Jiang, B. Zheng, Z. Sun, W. Sun, and C. Wang, "Trajectory similarity learning with auxiliary supervision and optimal matching," in *IJCAI*, 2020, pp. 3209–3215.

[35] Z. Fang, Y. Du, X. Zhu, D. Hu, L. Chen, Y. Gao, and C. S. Jensen, "Spatio-temporal trajectory similarity learning in road networks," in *SIGKDD*, 2022, pp. 347–356.

[36] DiDi Research, "Didi gaia open dataset," https://outreach.didichuxing.com/, 2022.

[37] Kaggle, "Porto taxi trajectory dataset," https://www.kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i/data, 2015.

[38] Microsoft, "T-drive trajectory data sample," https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/, 2011.

[39] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *TOIS*, pp. 422–446, 2002.

[40] L. Xiao, Y. Zheng, W. Tang, G. Yao, and L. Ruan, "Parallelizing dynamic time warping algorithm using prefix computations on gpu," in *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*. IEEE, 2013, pp. 294–299.

[41] L. Chen and R. T. Ng, "On the marriage of lp-norms and edit distance," in *VLDB*, 2004, pp. 792–803.

[42] T. Eiter and H. Mannila, "Computing discrete fréchet distance," 1994.

[43] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 2018, pp. 617–628.

[44] M. Gromov, "Hyperbolic groups," in *Essays in group theory*. Springer, 1987, pp. 75–263.

[45] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney, "Tree-like structure in large social and information networks," in *ICDM*. IEEE, 2013, pp. 1–10.

[46] E. Ravasz, A.-L. Barabási, and Z. Oltvai, "Hierarchical organization of complex networks," in *APS March Meeting*, vol. 2004, 2004, pp. V18–001.

[47] A. Ermolov, L. Mirvakhabova, V. Khrulkov, N. Sebe, and I. Oseledets, "Hyperbolic vision transformers: Combining improvements in metric learning," in *CVPR*, 2022, pp. 7409–7419.

[48] J. Yan, L. Luo, C. Deng, and H. Huang, "Unsupervised hyperbolic metric learning," in *CVPR*, 2021, pp. 12 465–12 474.

[49] V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, and V. Lempitsky, "Hyperbolic image embeddings," in *CVPR*, 2020, pp. 6418–6428.