Hyperbolic Representation Learning: Revisiting and Advancing

Menglin Yang [1] Min Zhou [2] Rex Ying [3] Yankai Chen [1] Irwin King [1]

Abstract

The non-Euclidean geometry of hyperbolic spaces has recently garnered considerable attention in the realm of representation learning. Current endeavors in hyperbolic representation largely presuppose that the underlying hierarchies can be automatically inferred and preserved through the adaptive optimization process. This assumption, however, is questionable and requires further validation. In this work, we first introduce a position-tracking mechanism to scrutinize existing prevalent hyperbolic models, revealing that the learned representations are sub-optimal and unsatisfactory. To address this, we propose a simple yet effective method, hyperbolic informed embedding (HIE), by incorporating cost-free hierarchical information deduced from the hyperbolic distance of the node to origin (i.e., induced hyperbolic norm) to advance existing hyperbolic models. The proposed method HIE is both task-agnostic and model-agnostic, enabling its seamless integration with a broad spectrum of models and tasks. Extensive experiments across various models and different tasks demonstrate the versatility and adaptability of the proposed method. Remarkably, our method achieves a remarkable improvement of up to 21.4% compared to the competing baselines.

1. Introduction

Hyperbolic space, considered as the continuous analogue of discrete trees (Krioukov et al., 2010), exhibits a natural advantage for modeling data with implicit or explicit tree-like layouts, such as hierarchical structures and power-law distributed data (Adcock et al., 2013; Zhou et al., 2022a). The superiority of hyperbolic space in representation learning, e.g., low distortion (Sarkar, 2011), small generalization error (Suzuki et al., 2021a;b), and impressive performance (Nickel & Kiela, 2017; Chami et al., 2019; Yang et al., 2022a), has been extensively validated in recent works, spanning a plethora of research topics and downstream applications (Peng et al., 2021; Yang et al., 2021a; Mettes et al., 2023), including graph learning, image, and text understanding.

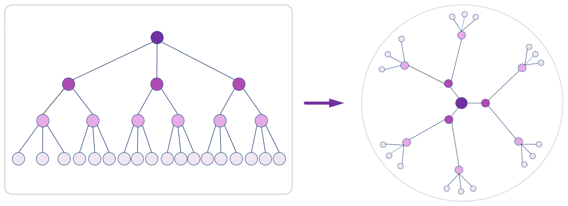


Figure 1. Illustration of an optimal embedding in hyperbolic space that preserves the local dependencies while maintaining a hierarchical relationships of the entities. Left: A tree-like graph. Right: Hyperbolic embedding of the tree-like graph.

An essential implicit and unspoken objective in utilizing hyperbolic space is to extract intrinsic hierarchical information within data. The expected learning objective, as illustrated in Figure 1, involves optimizing parent nodes above their respective descendants, or from a global perspective, to push the root closer to the hyperbolic origin while simultaneously situating leaf nodes at a greater distance from the hyperbolic origin. This can be understood from the following two aspects. Intuitively, in original data, the parent node occupies a higher hierarchical level than its offspring. Consequently, in the learning space, a high-fidelity embedding necessitates the parent node being positioned relatively higher position, or equivalently in smaller distance to the hyperbolic origin. Mathematically, let us consider the Poincaré ball model as an illustrative example, which is the most frequently used hyperbolic geometry models[1]. When the root node of a tree-like data structure is placed at the hyperbolic origin, it manifests a relatively small distance to all other nodes, as its norm is equal to zero. Meanwhile, leaf nodes typically locate closer to the periphery of the ball, as the distance between points increases rapidly as the norm approaches one, thus achieving an optimal and low-distortion embedding that effectively preserves the underlying hierarchical

[1]Department of Computer Sciences and Engineering, The Chinese University of Hong Kong [2]Huawei Technologies Co., Ltd. [3]Yale University. Correspondence to: Menglin Yang <mlyang.cuhk@outlook.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]The term "hyperbolic geometry models" denotes the mathematical formations for building hyperbolic geometry and creating hyperbolic space, e.g., Poincaré ball model and Lorentz model.

structure (Nickel & Kiela, 2017; 2018).

The majority of prior studies (Nickel & Kiela, 2017; 2018; Ganea et al., 2018; Shimizu et al., 2020; Chami et al., 2019; Liu et al., 2019; Zhang et al., 2021a) make the assumption that hyperbolic models[2] are capable of inferring continuous hierarchies from pairwise similarity measurements (Nickel & Kiela, 2017; 2018) or cross-entropy loss (Chami et al., 2019; Zhang et al., 2021b) in a self-optimization process. However, in the training process, the lack of prior geometric information, such as knowledge about the root, leaves, and hierarchical order, as well as the optimization with geometric-irrelevant objectives raise questions regarding the ability of hyperbolic models to arrange objects in a hierarchical manner and effectively.

To resolve this confusion, we initiate a thorough investigation into the prevalent hyperbolic models with the intent of ascertaining whether the inherent or latent hierarchical structure is adequately reflected in the resulting embeddings. We conduct an analysis based on a specially designed position-tracking approach, which is detailed in Section 3. The observations indicate that existing prevalent hyperbolic models fall short in effectively preserving hierarchies and capturing the underlying data structure. From a holistic perspective, the root and a significant proportion of leaf nodes are also optimized in unsatisfactory locations, lacking adequate separation in the hyperbolic space. To solve these problems, unlike previous works (Nickel & Kiela, 2017; 2018; Chami et al., 2019; Liu et al., 2019; Ganea et al., 2018; Shimizu et al., 2020; Zhang et al., 2021b) that solely projected objects into hyperbolic space without considering hierarchical knowledge and assuming that hierarchical dependencies can be automatically extracted in the learning process, we introduce cost-free geometric information to enhance the hierarchical learning process. More specifically, we harness the intrinsic geometric characteristics of hyperbolic space, namely hyperbolic distance to origin (HDO), to detect crucial geometric information, such as the root and hierarchical level, and subsequently align the root and ordering. The proposed method is both data and model-agnostic, enabling its facile application to a plethora of datasets and models.

To summarize, our work makes four key contributions:

- We propose a position-tracking strategy to investigate current prevalent hyperbolic models, revealing a significant discrepancy between the hyperbolic learning process and conventional understanding, shedding light on the process of hyperbolic representation learning.
- We introduce a novel method for inferring implicit hierarchy from hyperbolic embeddings. This approach is cost-free, scalable, and highly effective as it extracts hierarchical information directly from the embeddings themselves, eliminating the need for additional inputs or annotations.
- We present a simple yet effective approach that leverages the inferred hierarchy for advancing hyperbolic representation learning. Our method seamlessly integrates with existing hyperbolic models and does not introduce any additional model parameters, making it practical and easy to implement.
- We conduct extensive experiments on various hyperbolic models, demonstrating the effectiveness of our proposed method. The results show significant improvements over the baselines, with the largest improvement reaching 21.4%.

[2]The term "hyperbolic models" denotes deep or machine learning models that operate within hyperbolic space, including hyperbolic shallow models, hyperbolic neural networks, and hyperbolic graph neural networks.

2. Preliminary

2.1. Related Works

The field of neural networks has seen a growing interest in the incorporation of hyperbolic geometry (Peng et al., 2021; Yang et al., 2022b; Zhou et al., 2022b; Vyas et al., 2022; Xiong et al., 2022a), in areas like lexical entailment (Nickel & Kiela, 2017; Gulcehre et al., 2019; Sala et al., 2018), knowledge graphs (Chami et al., 2020; Bai et al., 2021; Sun et al., 2020; Xiong et al., 2022b), image understanding (Khrulkov et al., 2020; Zhang et al., 2020; Atigh et al., 2022; Hsu et al., 2021), and recommender systems (Vinh Tran et al., 2020; Chen et al., 2022; Sun et al., 2021a; Yang et al., 2022c;a). In the realm of graph learning (Gulcehre et al., 2019; Chami et al., 2019; Liu et al., 2019; 2022; Yang et al., 2022b), a significant amount of research works generalizing graph convolutions (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017; Yang et al., 2020; 2022d; Li et al., 2022; Zhang et al., 2019) in hyperbolic space for a better graph or temporal graph representation (Chami et al., 2019; Liu et al., 2019; Zhang et al., 2021b; Yang et al., 2021b; 2022b; Bai et al., 2023; 2022; Sun et al., 2021b), which has achieved impressive performance.

The choice of optimization targets or loss functions in hyperbolic learning models, is typically driven by specific downstream tasks or applications. For example, cross-entropy is utilized in node classification or graph classification tasks (Chami et al., 2019; Liu et al., 2019), while binary cross-entropy is commonly used for link prediction (Chami et al., 2019; Nickel & Kiela, 2018; Ganea et al., 2018), and ranking margin loss is employed for ranking process in recommender systems (Sun et al., 2021a; Vinh Tran et al., 2020; Yang et al., 2022a; Wang et al., 2021). However, many of these loss functions are task-specific and locally defined, without explicitly considering the underlying geometry of the hyperbolic space. This raises a concern regarding how hyperbolic learning approaches can effectively capture and learn the hierarchical structure of data without explicit or implicit guidance on hierarchical information.

2.2. Riemannian Geometry and Hyperbolic Space

Here are some preliminary definitions for Riemannian geometry and hyperbolic space. We refer the interested readers to (Spivak, 1973; Lee, 2006; 2013) for more details.

Riemannian Geometry. A Riemannian manifold $\mathcal{M}$ with d dimensions is a topological space that possesses a metric tensor g, denoted as $(\mathcal{M}, g)$. At any arbitrary point $\mathbf{x}$ in $\mathcal{M}$, the manifold can be locally approximated by a local linear space called the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$, which is isometric to $\mathbb{R}^d$. The shortest path between two points on the manifold is known as a geodesic, and its length is referred to as the induced distance $d_{\mathcal{M}}$. In particular, the distance of a node to the origin in hyperbolic space (HDO) can be construed as its corresponding induced hyperbolic norm. To project a tangent vector onto the manifold, the exponential map $\exp_{\mathbf{x}}$ is utilized, while its inverse function is known as the logarithmic map $\log_{\mathbf{x}}$. Another significant operation is parallel transport $P_{\mathbf{x}\to\mathbf{y}}$, which allows for the transportation of geometric data from $\mathbf{x}$ to $\mathbf{y}$ along the unique geodesics while preserving the metric tensors.

Hyperbolic Space. Hyperbolic space refers to a Riemannian manifold with a constant negative curvature, and its coordinates can be represented using various isometric models[3]. In the fields of machine learning and deep learning, the Poincaré ball model and the Lorentz model are widely studied. A d-dimensional *Poincaré ball model* with a constant negative curvature $\kappa(\kappa < 0)$ is defined as a Riemannian manifold $(\mathcal{B}^d_\kappa, g^\kappa_{\mathcal{B}})$, where $\mathcal{B}^d_\kappa = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\|^2 < -1/\kappa\}$ and its metric tensor $g^\kappa_{\mathcal{B}} = (\lambda^\kappa_{\mathbf{x}})^2 \mathbf{I}_d$ and $\mathbf{I}_d$ is d dimensional identity matrix. Here, $\mathcal{B}^d_\kappa$ is an d-dimensional open ball of radius $(-\kappa)^{-\frac{1}{2}}$, and $\lambda^\kappa_{\mathbf{x}} = 2(1+\kappa\|\mathbf{x}\|_2^2)^{-1}$ is the conformal factor. The Lorentz model of the hyperbolic space is another popular model. In this model, the space is visualized as one sheet of a two-sheeted hyperboloid in a (d+1)-dimensional Minkowski space. A d-dimensional *Lorentz model* (also called hyperboloid model) with a constant negative curvature $\kappa(\kappa < 0)$ is defined as a Riemannian manifold $(\mathcal{L}^d_\kappa, g^\kappa_{\mathcal{L}})$, where $\mathcal{L}^d_\kappa = \{\mathbf{x} \in \mathbb{R}^{d+1} \mid \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = 1/\kappa\}$ and its metric tensor $g^\kappa_{\mathcal{L}} = \mathrm{diag}([-1, 1, \cdots, 1])$. It is important to note that the study applies to both the Poincaré ball model and the Lorentz model, and for simplicity, a unified notation $\mathcal{H}$ is used. For the hyperbolic formula about exponential map $\exp^\kappa_{\mathbf{x}}$, logarithmic map $\log^\kappa_{\mathbf{x}}$, hyperbolic distance $d_{\mathcal{H}}$, and parallel transport $P^\kappa_{\mathbf{o}\to\mathbf{x}}$, and other related information, please refer to Appendix A.

[3] https://en.wikipedia.org/wiki/Hyperbolic_space

2.3. Hyperbolic Shallow Models

The hyperbolic shallow models encode entities into hyperbolic space and utilize the relative distance or similarity as a means to infer their ordering. For example, Nickel & Kiela (2017; 2018) proposed to embed data in Poincaré ball model as well as Lorentz model to learn embeddings by optimizing the following objective:

$$\max_{\mathbf{x}\in\mathcal{H}^d} \sum_{(i,j)\in E} \log \frac{\exp\left(-d_{\mathcal{H}}(\mathbf{x}_i, \mathbf{x}_j)\right)}{\sum_{j'\in Neg(i)} \exp\left(-d_{\mathcal{H}}(\mathbf{x}_i, \mathbf{x}_{j'})\right)}, \tag{1}$$

where E is a set consisting of linked or related object pairs, $Neg(i) = \{j|(i,j) \notin E\} \cup \{i\}$ is the set of negative examples for i, $\mathbf{x} \in \mathcal{H}^d$ is trainable node embedding ($\mathbf{x}_i$ indicates the specific object i), and $d_{\mathcal{H}}$ is the hyperbolic distance. The optimizing target encourages connected or related object pairs in the original structured data to have a small hyperbolic distance while pushing unconnected or unpaired nodes far apart.

2.4. Hyperbolic Neural Networks

Hyperbolic neural networks (HNNs) (Ganea et al., 2018; Shimizu et al., 2020) represent a generalization of traditional neural networks in hyperbolic space. The fundamental architecture of the HNN encompasses several components, including multinomial logistic regression (MLR), fully-connected (FC) layers, and recurrent neural networks (RNNs). For brevity, we mainly focus on the core module, namely FC layer, in this study. The corresponding hyperbolic FC is composed of a linear transformation and non-linear activation. Hyperbolic linear transformation begins with matrix multiplication, followed by bias translation. In general, this process can be achieved in the tangent space of origin (Ganea et al., 2018; Liu et al., 2019; Chami et al., 2019; Zhang et al., 2021b). Given a hyperbolic point $\mathbf{x} \in \mathcal{H}^d$, it is first projected to the tangent space of origin $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$ using the logarithmic map and multiplied by the weight matrix, $\mathbf{W} \in \mathbb{R}^{d'\times d}$. Hyperbolic matrix multiplication is defined as, $\mathbf{W} \otimes^\kappa \mathbf{x} := \exp^\kappa_{\mathbf{o}}(\mathbf{W} \log^\kappa_{\mathbf{o}}(\mathbf{x}))$. For bias addition, let us use a bias $\mathbf{b}$ located at $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$, parallel transport it to the hyperbolic point of interest, and map it to the manifold, $\mathbf{x} \oplus^\kappa \mathbf{b} := \exp^\kappa_{\mathbf{x}}(P^\kappa_{\mathbf{o}\to\mathbf{x}}(\mathbf{b}))$. The non-linear activation is also achieved at the tangent space, and for simplicity, we use hyperbolic origin, and it is given as, $\sigma^{\kappa_1,\kappa_2}_{\mathcal{H}}(\mathbf{x}) := \exp^{\kappa_1}_{\mathbf{o}}\left(\sigma(\log^{\kappa_2}_{\mathbf{o}}(\mathbf{x}))\right)$. Finally, given $\mathbf{x} \in \mathcal{H}^d$, the ℓ-th hyperbolic neural network layer is formulated as:

$$\mathbf{x}^\ell = \sigma^{\kappa_\ell,\kappa_{\ell-1}}_{\mathcal{H}}\big(\mathbf{W} \otimes^{\kappa_{\ell-1}} \mathbf{x}^{\ell-1}\big) \oplus^{\kappa_{\ell-1}} \mathbf{b}\big), \tag{2}$$

where κ_ℓ and $\kappa_{\ell-1}$ is the curvature at layer ℓ and $\ell-1$, respectively.

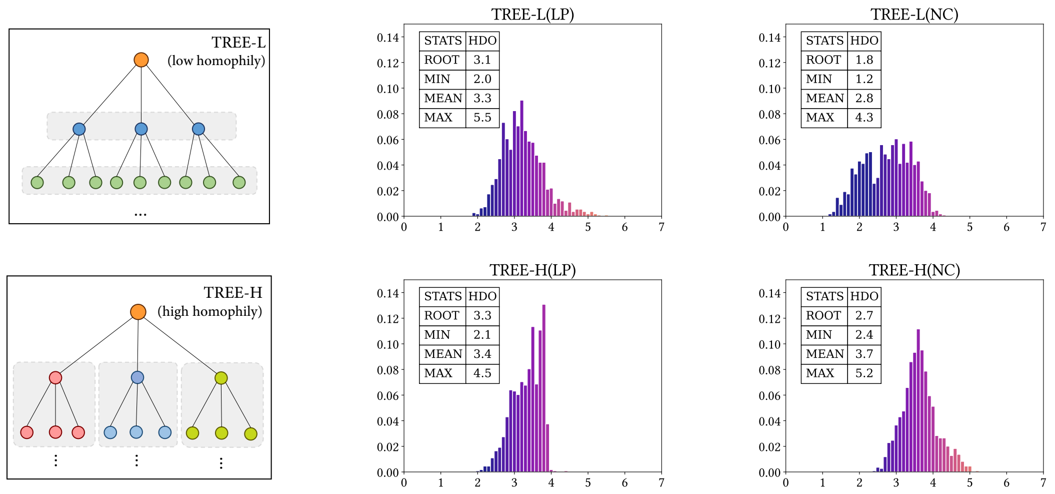


Figure 2. Illustration of the structure of synthetic TREE-L/H (first column), and the corresponding distribution of HDO by HGCN on link prediction (LP) task (second column) and node classification (NC) (third column).

2.5. Hyperbolic Graph Convolutional Neural Networks

The cornerstone of Hyperbolic Graph Convolutional Neural Networks (HGCNs) is the execution of graph convolutions within the bounds of hyperbolic space. HGNNs employ the message passing scheme, including linear transformation, neighbors aggregation, and non-linear activation. The research presented in this paper is applicable to both of the aforementioned cases. Compared with the HNNs given by Equation (2), the HGCN layer has one neighborhood aggregation more. Let $j \in \mathcal{N}(i)$ be the neighbors of node i with self-loop, the hyperbolic neighborhood aggregation is given as, $\textsc{Agg}^{\kappa}(\mathbf{x}_i) := \exp^{\kappa}_{\mathbf{x}_r}\left(\sum_{j\in\mathcal{N}(i)} \tilde{a}_{ij} \log^{\kappa}_{\mathbf{x}_r}(\mathbf{x}_j)\right)$, where $\mathbf{x}_r$ is a reference point in hyperbolic space, which is set as the origin point for simplicity in this study, and $\tilde{a}_{ij}$ is the normalized edge weights, which is computed either by an attention-based method (Chami et al., 2019), i.e., $\tilde{a}_{ij} = \text{Softmax}_{j\in\mathcal{N}(i)}(\text{MLP}(\log^{\kappa}_{\mathbf{o}}(\mathbf{x}_i)||\log^{\kappa}_{\mathbf{o}}(\mathbf{x}_j)))$ or by the symmetric normalized Laplacian matrix (Kipf & Welling, 2017; Liu et al., 2019), i.e., $\tilde{a}_{ij} = 1/\sqrt{\tilde{d}_i\tilde{d}_j}$, where the $\tilde{d}_v$ denotes the degree of node v including the self-loop. Then, one HGNN layer is formulated as:

$$\begin{aligned} \mathbf{x}_i^{\ell} &= \sigma_{\mathcal{H}}^{\kappa_{\ell-1},\kappa_\ell}(\tilde{\mathbf{h}}_i^{\ell}), \\ \tilde{\mathbf{h}}_i^{\ell} &= \textsc{Agg}^{\kappa_{\ell-1}}(\mathbf{h}_i^{\ell}), \\ \mathbf{h}_i^{\ell} &= (\mathbf{W}^{\ell} \otimes^{\kappa_{\ell-1}} \mathbf{x}_i^{\ell-1}) \oplus^{\kappa_{\ell-1}} \mathbf{b}^{\ell}. \end{aligned} \tag{3}$$

3. Investigation

3.1. Hyperbolic Distance to Origin (HDO)

Before delving into the analysis of the tracking of embedding trajectory, it is imperative to introduce an essential tool for the investigation.

The hyperbolic distance from a point $\mathbf{x}$ to the origin $\mathbf{o}$ (HDO, also referred to as the induced hyperbolic norm), $d_{\mathcal{H}}(\mathbf{x}, \mathbf{o})$, has found considerable usage as an interpretative post-hoc indicator, particularly for explicating the intricacies of embedding levels across diverse domains. For instance, in WordNet[4] embedding, Nickel & Kiela (2017) employed the difference of embedding norms to define a score function for evaluating the effectiveness of Poincaré embedding in representing semantic relationships. In image embedding, Khrulkov et al. (2020) incorporated HDO as an uncertainty quantifier and found that hard-to-classify images are embedded near the hyperbolic origin, whereas easy-to-classify images are placed around the boundary of the Poincaré. Hard-to-classify images often encompass patterns from various categories, which are associated with high-level positions in the hierarchy. Conversely, the easy-to-classify images exhibit specific patterns, which are related to low-level positions in the hierarchy. A subsequent study by Sun et al. (2021a) leveraged HDO to analyze the alignment of hyperbolic user-item embeddings with popularity. Their analysis concluded that smaller HDO values were indicative of higher popularity levels. A higher popularity of a item means that the item is favored by a substantial users, with the most popular item functioning as a network hub or a tree root.

In a k-regular tree, the number of nodes at level r proliferates exponentially, specifically, $(k+1)k^{r-1}$, and the number of nodes at levels less than or equal to r also follows a similar trend, specifically, $((k+1)k^r - 2)/(k-1)$. This implies that the number of nodes escalates with the distance from the root of the tree. Analogously, in hyperbolic space, the area of a disc also expands exponentially with distance from the hyperbolic origin. Therefore, strategically placing

[4] https://wordnet.princeton.edu/

Algorithm 1 Hyperbolic Informed Embedding (HIE)

```
1:  Input: (i) Input data $\mathcal{X}$ which can be either $n$ general input samples $\mathcal{X} = \{\mathbf{x}_1, \cdots \mathbf{x}_n\}$ or $n$ nodes in a graph with adjacency matrix, i.e., $\mathcal{X} = \{\mathbf{x}_1, \cdots \mathbf{x}_n, \mathbf{A}\}$; $\mathbf{x}_i$ can be randomly sampled and trainable or pre-defined. (ii) Learning model $\mathcal{F}$ which is optionally parameterized by $\mathbf{W}$; (iii) Hyperparameter $\lambda > 0$.
2:  Output: Embeddings of $n$ objects and optimal $\mathbf{W}$.
3:  Initialize input data $\mathcal{X}$ and learnable weight $\mathbf{W}$ (optional) of $\mathcal{F}$;
4:  Compute embedding $\mathbf{Z} \leftarrow \mathcal{F}([\mathbf{W}], \mathcal{X})$;
5:  if partial root-alignment then
6:      Calculate task-specific loss: $L_{\text{task}}$;
7:      Calculate root aligned embedding $\bar{\mathbf{Z}}$;
8:      Calculate hyperbolic distance to origin loss:$L_{\text{hyp}}$;
9:      Optimize with overall loss function: $L_{\text{task}} + \lambda L_{\text{hyp}}$;
10:     Return $\mathbf{Z}$,$\mathbf{W}$
11: end if
12: if whole root-alignment then
13:     Calculate root aligned embedding$\bar{\mathbf{Z}}$;
14:     Calculate hyperbolic distance to origin loss:$L_{\text{hyp}}$;
15:     Calculate task-specific loss: $L_{\text{task}}$;
16:     Optimize with overall loss function: $L_{\text{task}} + \lambda L_{\text{hyp}}$;
17:     Return $\bar{\mathbf{Z}}$,$\mathbf{W}$
18: end if
```

the nodes of level r of a tree at a distance R (where R is proportionate to r) (Krioukov et al., 2010; 2009) from the hyperbolic origin results in a hierarchical embedding that captures the underlying tree-like structure. Building on this insight, in this study, we design a position tracking analysis by HDO. Besides, we also utilize it as a tool to guide representation learning in hyperbolic space.

3.2. HDO Investigation

Due to the lack of crucial geometric information, such as the root label, level information, and leaf nodes, in real-world graph datasets, we synthesized two pure-tree datasets. To ensure our findings do not lose generality, we created two types of trees with varying degrees of homophily[5]: TREE-H (homophily rate of 0.998) and TREE-L (homophily rate of 0.018). Detailed information regarding the construction of TREE-L and TREE-H can be found in Appendix B. We conduct an experimental evaluation of the HGCN on node classification and link prediction tasks and depict the distribution of the HDO for the node states in the final embedding layer as illustrated in Figure 2. It is noteworthy that other hyperbolic models have demonstrated comparable results. Furthermore, we provide essential statistics regarding HDO, such as the minimum (MIN), maximum (MAX), mean (MEAN), and HDO of the root node (ROOT), in the corresponding subfigures for the convenience of readers.

It can be observed from the experimental results that in both cases, root nodes exhibit a large disparity with the minimum HDO (*c.f.*, MIN and ROOT values). For instance, in the link prediction task, the ROOTs of TREE-L and TREE-H are 3.1 and 3.3, respectively, while the MINs are 2.0 and 2.1. Moreover, the distribution of HDO exhibits a more normal pattern. However, in trees, leaf nodes constitute the majority and are expected to be located farther from the origin, following a power-law distribution. Moreover, the nodes are not fully spreading, which can be inferred from the shape of the distribution and the gap between min, mean, and max HDO. It is worth noting that the hyperbolic space expands exponentially, with the area far from the origin being significantly more spacious and accommodating.

Overall, the aforementioned investigation reveals the following: (1) The root node is not optimized to occupy or close to the highest level, potentially resulting in an inaccurate hierarchical structure; (2) The HDO of the learned embeddings is normally distributed, which inadequately captures the tree-like structure; (3) The overall embeddings are not maximally scattered, indicating that the model fails to fully leverage the expansive nature of the hyperbolic space. These limitations motivate us to reshape the optimization objective by explicitly incorporating root alignment and hierarchical stretching.

4. Method

To address the above problem, we proposed the HIE algorithm which is demonstrated in Algorithm 1. HIE can function both as a novel learning paradigm incorporating `whole root-alignment` settings, and as a flexible plug-in module by `partial root-alignment`. The core of HIE is highlighted in blue in Algorithm 1, and we provide further details in the following.

As sketched in Figure 3, the intuition of our idea starts with identifying the root node of the overall data and aligning it with the origin of the hyperbolic space. Then we optimize each node away from the origin according to its level information. This not only facilitates the formation of the correct hierarchies but also makes full use of the hyperbolic space. However, there are two challenges to achieving this target: **(i) Defining and locating the root node:** Numerous indicators, such as centrality[6], can describe a node's role in the graph. However, these methods solely rely on the graph topology and overlook the node's features. Additionally, some of these indicators are computationally expensive when dealing with large-scale graphs. Moreover, tree-like datasets often contain multiple subtrees or roots. **(ii) Effi-**

[5]Homophily is a metric used to differentiate assortative and disassortative graphs, as defined by (Pei et al., 2020).

[6]https://en.wikipedia.org/wiki/Centrality

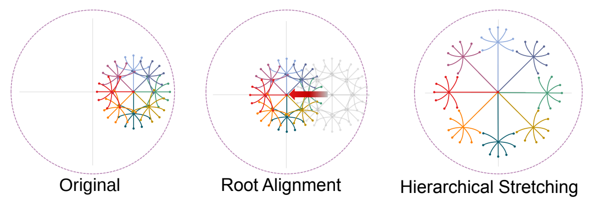


Figure 3. Illustration of the basic idea of HIE, which can be decomposed into two critical steps: root alignment and level-aware stretching.

ciently accessing level information and guiding hierarchical learning: This hierarchical information is rarely on hand in real-world scenarios and is labor-consuming to label in the large-scale dataset.

(1) To address the first challenge, we propose utilizing the hyperbolic embedding center (HC) as the root node and making an alignment with the hyperbolic origin. HC is computed through a manifold-specific method and serves as the optimal solution for minimizing the sum of squared distances with all nodes, as outlined by Theorem 4.1. This conforms to the characteristics of the root node in a regular tree, that is, the node with the smallest sum of distances to other nodes.

Remark. Utilizing the HC as the root node presents several key advantages: (i) Computation of the HC is relatively straightforward, eliminating the need for extensive topology search. The computational cost is $O(|V|)$, or even $O(1)$ when utilizing parallel computing with a GPU, which is significantly more efficient than other centrality computations; (ii) The HC is derived from hyperbolic embeddings, which encode both structural and feature information. It effectively handles scenarios with multiple root nodes since the HC always represents a unique point that can be seen as a supernode connecting all subtrees; (iii) HC can be adjusted adaptively according to downstream tasks during the learning process.

Let $\mathbf{Z}$ be the final layer embedding matrix consisting of n hyperbolic node vectors $(\mathbf{z}_1, \cdots, \mathbf{z}_i, \cdots, \mathbf{z}_n)$ where $\mathbf{z}_i \in \mathcal{H}^d_\kappa$ and let v_i denote the weight assigned to $\mathbf{z}_i$. The function f_c is used for computing the weighted center, and $\mathbf{z}_c$ represents this center, which is computed as: $\mathbf{z}_c = f_c(\{\mathbf{z}_i, v_i | i \in V\})$, where f_c is detailed in Appendix C. Following this, we employ the *root alignment* strategy as defined in Equation (4).

(2) To address the second challenge, we propose a hierarchical stretching of the nodes by leveraging the HDO, which serves as a means of encapsulating implicit hierarchical information, as discussed in Section (3.1). Specifically, by aligning the hyperbolic embedding center (i.e., the root node) with the origin of the hyperbolic space, the HDO more accurately reflects the relative distance of a node to the root node, thus indicating its hierarchical level, which is given by Equation (5). For implicitly *hierarchical stretching*, it is incorporated into the loss function (6).

Table 1. Comparisons of shallow models for link prediction task on DISEASE. The first and second rows correspond to the AUC and AP metrics, respectively.

Link	75%Training Links			25%Training Links		
Dim	8	64	256	8	64	256
Euclidean	61.1 ± 2.4	72.2 ± 0.5	73.5 ± 0.4	53.5 ± 0.2	54.3 ± 0.3	54.1 ± 0.2
Hyperbolic	65.2 ± 3.2	74.9 ± 0.4	77.0 ± 0.3	52.9 ± 0.5	53.7 ± 0.5	55.0 ± 0.4
Ours	**72.0** ± 0.9	**80.7** ± 0.4	**82.5** ± 0.2	**58.1** ± 1.1	**63.9** ± 0.4	**66.8** ± 0.4
$\Delta(\%)$	+10.4	+7.7	+7.1	+8.6	+17.7	+21.4
Euclidean	58.7 ± 1.3	69.7 ± 0.5	71.4 ± 0.7	52.2 ± 0.2	52.8 ± 0.6	52.3 ± 0.1
Hyperbolic	62.3 ± 2.7	71.3 ± 0.7	72.7 ± 0.4	51.9 ± 0.7	52.4 ± 0.8	54.1 ± 0.7
Ours	**66.6** ± 1.0	**75.0** ± 0.4	**76.4** ± 0.3	**56.1** ± 1.0	**62.6** ± 0.6	**65.2** ± 0.6
$\Delta(\%)$	+6.9	+5.2	+5.0	+7.4	+18.7	+20.4

$$\bar{\mathbf{z}} = \mathbf{z} \oplus_\kappa (-\mathbf{z}_c), \qquad \text{(root alignment)} \quad (4)$$

$$z_{\text{hdo}} = \frac{1}{|V|} \sum_{i \in V} w_i d_{\mathcal{H}}(\bar{\mathbf{z}}_i, \mathbf{o}), \qquad \text{(HDO computation)} \quad (5)$$

$$L_{\text{hyp}} = \sigma(-z_{\text{hdo}}), \qquad \text{(stretching)} \quad (6)$$

where in Equation (4), $\oplus_\kappa$ denotes the hyperbolic addition operation and its mathematical expressions is provided in Appendix C; In Equaiton (5), w_i indicates the node level in hyperbolic space which is obtained from the HDO ($w_i := f(d_{\mathcal{H}}(\bar{\mathbf{z}}_i, \mathbf{o}))$) and f is monotonically increasing function, such as linear function, tanh, sigmoid, etc, we use identity function for simplicity; In Equation (6), the stretching is achieved by minimizing L_{hyp} with monotonically increasing function σ (linear function, tanh, exp, etc).

Remark. By minimizing the above loss function, the high-level nodes that are close to the origin will be placed with small weights so that they will not push away; the low-level nodes that are far away from the origin will be given large weights so that they can arrive at a correct position. In this approach, we optimize by promoting nodes to move away from the origin rather than close to the origin, primarily because the capacity of hyperbolic space increases exponentially, and regions further from the origin bend more, thereby providing more embedding space.

In the field of hyperbolic learning, a common approach is to model data in the tangent space at the origin of the hyperbolic space. Thus, we introduce an extension to the HIE model, referred to as the tangent-version HIE[7]. Suppose we have the tangential embedding $\mathbf{z}^{\mathcal{T}}$, then we have the Equation (7)(8)(9).

$$\bar{\mathbf{z}}^{\mathcal{T}} = \mathbf{z}^{\mathcal{T}} - \mathbf{z}^{\mathcal{T}}_c \qquad \text{(root alignment)} \quad (7)$$

$$z^{\mathcal{T}}_{\text{hdo}} := \frac{1}{|V|} \sum_{i \in V} w_i d(\bar{\mathbf{z}}^{\mathcal{T}}_i, \mathbf{o}^{\mathcal{T}}) \qquad \text{(HDO computation)} \quad (8)$$

$$L_{\text{hyp}} = \sigma(-z^{\mathcal{T}}_{\text{hdo}}) \qquad \text{(stretching)} \quad (9)$$

[7]Experimental studies demonstrate that both the tangent version and hyperbolic version exhibit comparable outcomes.

Table 2. Comparisons of the different neural networks on node classification tasks. $\delta_{(\text{DISEASE})}$ is 1.0 and $\delta_{(\text{CITESEER})}$ is 2.5.

Dataset	Dim	MLP	HNN	HNN++	Ours	$\Delta(\%)$
	8	23.5 ± 23.7	46.6 ± 12.6	59.1 ± 9.5	**65.6** ± 6.6	$+11.0$
DISEASE	64	63.3 ± 7.8	68.4 ± 4.4	67.4 ± 2.0	**78.4** ± 0.7	$+14.6$
	256	70.9 ± 3.5	75.1 ± 2.8	76.1 ± 5.7	**77.3** ± 0.9	$+1.60$
	8	53.6 ± 1.8	52.5 ± 1.6	48.0 ± 3.9	**63.7** ± 2.1	$+18.8$
CITESEER	64	59.0 ± 0.7	57.2 ± 0.7	59.1 ± 1.1	**67.0** ± 1.0	$+13.4$
	256	59.1 ± 0.7	58.2 ± 0.9	60.2 ± 1.0	**68.2** ± 0.5	$+13.3$

where $\mathbf{z}_c^{\mathcal{T}} := \frac{1}{|V|}\sum_{i\in V}\mathbf{z}_i^{\mathcal{T}}$ denotes the center of the tangential embedding, and w_i derived from $d(\bar{\mathbf{z}}_i, \mathbf{o})$ and we use identity function for simplicity, which signifies the Euclidean distance. The symbol σ represents a monotonically increasing function such as linear function, tanh, sigmoid, etc. Similarly, the hyperbolic tangent embedding center is the optimal solution for minimizing the weighted sum of squared distance with all nodes in the tangent spaces, as described in Theorem 4.2.

Theorem 4.1. *Given the node embedding $\mathbf{z} \in \mathcal{H}^d$ of a graph $G = \{V, E\}$, the hyperbolic embedding center $\mathbf{z}_c \in \mathcal{H}^d$ is a solution of the minimization problem of the sum of weighted squared hyperbolic distance with all nodes in graph G, which is expressed as follows:*

$$\mathbf{z}_c = \min_{\mathbf{z}_a\in\mathcal{H}^{d,\kappa}} \sum_{i\in V} v_i d_{\mathcal{H}}^2(\mathbf{z}_i, \mathbf{z}_a), \tag{10}$$

where $\mathbf{z}_a$ is any point in the manifold, and the embedding center includes Möbius gyromidpoint in Poincaré ball model, weighted centroid in the Lorentz model.

Theorem 4.2. *Given the node embedding $\mathbf{z} \in \mathcal{H}^d$ of a graph $G = \{V, E\}$, the hyperbolic tangent embedding center $\mathbf{z}_c^{\mathcal{T}} \in \mathcal{T}_{\mathbf{o}}\mathcal{H}^d$ is a solution of the minimization problem of the sum of weighted squared distance with all nodes in graph G at the tangent space $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$, which is expressed as follows:*

$$\mathbf{z}_c^{\mathcal{T}} = \min_{\mathbf{z}_a^{\mathcal{T}}\in\mathcal{T}_{\mathbf{o}}\mathcal{H}^{d,\kappa}} \sum_{i\in V} v_i d^2(\mathbf{z}_i^{\mathcal{T}}, \mathbf{z}_a^{\mathcal{T}}), \tag{11}$$

where $\mathbf{z}_a^{\mathcal{T}}$ is any point in the tangent space of the manifold.

Note that the proposed method can be implemented in two ways: one is a hard operation, which involves replacing the all original embeddings, and the other is generating the desired update gradients by implementing it partially in the L_{hyp} loss function as illustrated in Algorithm 1. After conducting extensive experiments, we have found that both approaches yield comparable results. In the following section, for a better visualization comparison, we mainly report the results obtained using the partially root-alignment setting.

5. Experiments

5.1. Experimental Settings

Datasets. In terms of experimental datasets, we perform evaluations on four public available datasets, namely DISEASE, AIRPORT, CORA, CITESEER and. For more details about these datasets, please refer to Appendix F.1. *Experimental Setups.* To ensure fairness, we employed the same data split for all comparison models in each experiment. The averaged results and standard deviation presented in these tables were computed from 12 runs, with the maximum and minimum values removed. The experiments were executed with the PyTorch (Paszke et al., 2017) on the NVIDIA GPUs using Adam (Kingma & Ba, 2014) or Riemannian Adam (Bécigneul & Ganea, 2018) optimizer. Due to the page limitation, we only list key findings here. Additional details regarding the experimental settings and results can be found in Appendix F.2, G and H.1.

5.2. Experiments on Shallow Models

The shallow models optimize node embeddings as learnable parameters. In line with prior research, we mainly evaluate their generalization capability on tree-structured graphs through link prediction tasks. To comprehensively ascertain the effectiveness of the proposed method, we conduct a thorough experimental evaluation across a diverse range of training settings. Specifically, the training edge ratio is varied at 75% and 25%. Intuitively, as the amount of available structure information is reduced, abstracting the intrinsic hierarchical structure becomes increasingly challenging.

We conduct a comparative evaluation between the Euclidean shallow model and the hyperbolic shallow model (based on the Poincaré ball). The experimental results on AUC and AP metrics are presented in Table 1. The findings from the results are as follows: The hyperbolic shallow models (including Poincaré and ours) demonstrate a distinct advantage over their Euclidean counterparts in most cases. However, as the amount of supervision information decreases, i.e., by reducing the training link ratio to 25%, the performance of the conventional hyperbolic model deteriorates dramatically and even falls below that of the Euclidean model. This is understandable, as the model faces a more arduous task in perceiving the overall structure of the tree-like data when the number of known links is limited. On the contrary, our proposed method, with implicit hierarchical learning, effectively addresses this challenge and produces a substantial improvement, with the maximum enhancement reaching 21.4%.

5.3. Experiments on HNNs

HNNs utilize the feature of nodes instead of the topology information for graph learning, so here we carry out the

Table 3. Comparisons on graph neural networks in Euclidean and hyperbolic space

Dataset	DISEASE ($\delta = 0.0$)			AIRPORT ($\delta = 1.0$)			CITESEER ($\delta = 2.5$)			CORA ($\delta = 11.0$)		
Dim	8	64	256	8	64	256	8	64	256	8	64	256
SGC	70.6 ± 6.3	77.4 ± 0.5	78.7 ± 0.4	71.9 ± 1.5	83.3 ± 1.4	81.9 ± 0.7	70.1 ± 0.6	72.2 ± 0.4	72.2 ± 0.4	80.2 ± 0.8	82.2 ± 0.5	81.9 ± 0.5
SAGE	66.5 ± 8.2	71.3 ± 5.3	72.0 ± 4.1	75.4 ± 1.1	87.1 ± 1.6	85.2 ± 1.7	71.0 ± 0.8	70.0 ± 0.8	70.9 ± 0.8	81.0 ± 0.8	80.5 ± 0.6	80.4 ± 0.6
GCN	76.5 ± 2.9	77.7 ± 1.3	78.8 ± 1.5	71.2 ± 1.8	85.2 ± 0.5	84.1 ± 0.6	69.6 ± 0.3	70.9 ± 0.5	71.0 ± 0.5	81.4 ± 0.6	81.9 ± 0.3	82.0 ± 0.2
GAT	76.8 ± 1.2	78.7 ± 1.5	80.2 ± 2.0	73.7 ± 1.5	87.5 ± 0.4	90.4 ± 1.2	69.8 ± 0.3	72.0 ± 0.4	71.7 ± 0.5	81.5 ± 0.4	82.9 ± 0.3	$\mathbf{83.0} \pm 0.3$
LGCN	86.6 ± 0.7	87.1 ± 0.8	88.2 ± 0.5	88.2 ± 1.7	91.8 ± 1.3	92.4 ± 0.3	66.8 ± 0.7	69.3 ± 0.8	70.5 ± 0.5	78.0 ± 2.0	81.2 ± 1.0	81.3 ± 0.7
HGCN	86.0 ± 3.2	90.6 ± 1.2	92.2 ± 1.8	89.8 ± 1.2	94.0 ± 0.6	94.1 ± 0.5	65.6 ± 1.5	67.6 ± 0.4	67.6 ± 0.7	78.2 ± 0.6	78.5 ± 0.6	79.1 ± 0.4
Ours	$\mathbf{94.4} \pm 0.6$	$\mathbf{95.0} \pm 0.8$	$\mathbf{95.4} \pm 0.8$	$\mathbf{89.8} \pm 1.1$	$\mathbf{94.1} \pm 0.8$	$\mathbf{94.7} \pm 0.4$	$\mathbf{72.5} \pm 1.1$	$\mathbf{74.1} \pm 0.3$	$\mathbf{74.2} \pm 0.3$	$\mathbf{81.8} \pm 0.6$	$\mathbf{83.0} \pm 0.3$	$\mathbf{83.0} \pm 0.2$

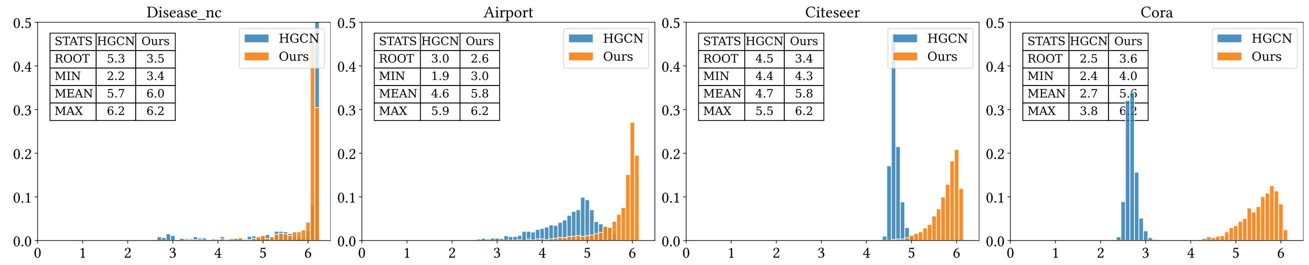


Figure 4. Illustration HDO distribution on DISEASE, AIRPORT, CITESEER and CORA where x-axis denotes the value of HDO and y-axis is the corresponding ratio. For a complete comparison, please refer to Appendix H.1. Here the Root denotes HC.

task of node classification. For model comparisons, we include Euclidean MLP, HNN (Ganea et al., 2018), and HNN++ (Shimizu et al., 2020). Compared with HNN, HNN++ reduces the number of parameters, so it is more lightweight. Our work is built upon HNN other than HNN++ with the following consideration: without any additions or reductions to the original model, the effectiveness of the proposed method could be more clear. But note that the proposed method is also applicable to HNN++. In this section, we assess the performance of our proposed model on two datasets: DISEASE with low hyperbolicity and CITESEER with high hyperbolicity. Hyperbolicity, a metric from graph theory, quantifies the tree-like structure of a network; lower values indicate a more tree-like structure. By analyzing the performance across datasets of varying hyperbolicity, we establish the robustness and generalizability of our model.

As shown in Table 2, we notice that hyperbolic models are more prominent in low-hyperbolicity tree-like data, i.e., DISEASE, but is poorer in general graph data, like CITESEER and even perform lower than the Euclidean model. Although the modified HNN++ improves the performance of the HNN model to a certain extent, its gains are modest. In contrast, our method has a significant improvement over the original HNN model, regardless of the low or high hyperbolicity graphs. In addition to being able to extract the underlying hierarchical structure of the data, the hyperbolic space has a larger room for embedding than the Euclidean space. The improvement of the proposed method on a high-hyperbolicity graph is mainly from the stretching operation where the nodes are fully expanded in the hyperbolic space and have nice separability.

5.4. Experiments on HGNNs

For model comparisons, we compare our proposed method against (1) four Euclidean GCN models, i.e., GCN (Kipf & Welling, 2017), Graph Attention Networks (GAT) (Veličković et al., 2018), SGC (Wu et al., 2019) and SAGE (Hamilton et al., 2017); and (2) two prominent hyperbolic GCN model, i.e., HGCN (Chami et al., 2019) and LGCN (Zhang et al., 2021b), we take two different aggregation methods as described in Section 2.5 and report the best results of them. Our method is built upon the original model, HGCN. To ensure fairness, we adhered to the parameters and experimental strategies outlined in the original papers of all baselines and conducted extensive experiments in a consistent environment. Following the literature, we test on node classification task and report the F1-score for DISEASE and AIRPORT datasets and accuracy for the others.

The experimental results are reported in Table 3. As observed, the proposed method *consistently* outperforms strong hyperbolic baselines HGCN and LGCN, which demonstrates the effectiveness of the proposed method. We further dig into the details and have the following observations: (1) The performance of baselines varies along embedding dimensions. With the proposed HIE, the hyperbolic model outperforms the baselines, especially achieving remarkable gains on CITESEER, and DISEASE. (2) As noted, the improvement on AIRPORT is relatively small. A possible explanation for this is that AIRPORT is well hierarchically organized. The original hyperbolic models have already successfully learned the proper embeddings, and the improvement is limited. Nonetheless, the proposal pushes the accuracy to a new level and obtains state-of-the-art performance.

Table 4. Ablation Study of HIE

Dataset	HGCN	w/ Stretching	w/ Alignment	Ours
DISEASE	90.6 ± 1.2	89.4 ± 2.5	94.5 ± 0.6	$\mathbf{95.0} \pm 0.8$
AIRPORT	94.0 ± 0.6	93.6 ± 0.9	93.6 ± 0.3	$\mathbf{94.1} \pm 0.8$
CITESEER	67.6 ± 0.4	68.5 ± 0.6	66.7 ± 0.9	$\mathbf{74.1} \pm 0.3$
CORA	78.5 ± 0.6	81.5 ± 0.5	77.5 ± 1.0	$\mathbf{83.0} \pm 0.3$

5.5. Analysis and Discussion

Visualization of HDO Distribution. We conduct a thorough analysis of the changes in HDO between the original HGCN and our proposed method. To provide a clear comparison with the baselines, we implement root centering in the loss function without altering the original node embeddings directly. The results presented in Figure 4 reveal that our method results in an increase in the mean HDO value, indicating that the node spreads more in the hyperbolic space. Additionally, it is observed that the shape of the HDO distribution has undergone a transformation, with a notable increase in the proportion of tail nodes, resulting in a more long-tailed distribution that better captures the tree-like or scale-free structure of the data. Furthermore, the root node (i.e., HC) is optimized to close the highest position (i.e., min HDO), demonstrating higher preservation of hierarchy.

Ablation Study of HIE. HIE contains two steps: root alignment and level-aware stretching. In the following, we perform the decoupling analysis by decomposing HIE into the corresponding two components. The experimental results are summarized in Table 4. Unsurprisingly, the performance decreases by different degrees if we remove each component. With only level-aware stretching, the performance of HIE barely satisfies and even degrades on low hyperbolicity datasets such as DISEASE and AIRPORT. It is consistent with the expectation that merely stretching without aligning the root to the origin will lead to an incorrect hierarchy and large distortions for a highly tree-like dataset. Equipped with only alignment, on the other hand, the HIE has witnessed apparent improvement on DISEASE but some declines on CORA or CITESEER. It is because alignment will arrange embeddings to the area near the origin, which is much more cramped than that close to the boundary.

Effectiveness of HC. Previously, we proposed the utilization of HC as the root node. For graph data, there exist numerous indicators to define the significance of nodes, such as degree centrality (DC), betweenness centrality (BC), and closeness centrality (CC). The definitions and computation equations of these metrics can be found in Appendix E. We implement these centralities HGCN and evaluate their performance in the context of node classification. As shown in Table 5, the experimental results reveal that alternative metrics do not exhibit the same effectiveness as HC. This is probably because these metrics only take into account topological information to determine the weight of the most prominent nodes while neglecting the role of node features. Moreover, the computational complexity of calculating these centralities is high, whereas HC is relatively computationally efficient.

Table 5. Comparisons with different centralities

Dataset	BC	CC	DC	HC(Ours)
DISEASE	85.0 ± 2.9	81.6 ± 3.2	85.0 ± 2.9	$\mathbf{95.0} \pm 0.8$
AIRPORT	93.4 ± 0.6	94.0 ± 0.5	93.3 ± 0.5	$\mathbf{94.1} \pm 0.8$
CITESEER	66.8 ± 0.7	66.5 ± 0.4	67.3 ± 0.5	$\mathbf{74.1} \pm 0.3$
CORA	81.3 ± 0.4	81.3 ± 0.4	81.3 ± 0.4	$\mathbf{83.0} \pm 0.3$

Discussion with Non-Hyperbolic Methods. When addressing the modeling of trees, it is important to note that the scope is not limited to hyperbolic models, such as the ones presented in the works by Sonthalia et al. (Sonthalia & Gilbert, 2020), Abraham et al. (Abraham et al., 2007), Chepoi et al. (Chepoi et al., 2008), and Saitou et al. (Saitou & Nei, 1987). One cannot ignore the fact that they primarily focus on the topological structure of the graph, while neglecting the features that the objects carry. Despite this limitation, these methods can serve as an excellent start point for enhancing the learning of hyperbolic models. Furthermore, they can be seamlessly integrated into the proposed method, thus leveraging their strengths while addressing the need to consider the object features.

6. Conclusion

In this work, we first investigated the hierarchical representation ability of the currently popular hyperbolic models, including hyperbolic shallow models, HNNs, and HGNNs. The results indicate that the current hyperbolic models fail to obtain desired hierarchical embeddings. With the proposed method, HIE, the hyperbolic representation ability has been substantially improved. More importantly, the proposed method does not introduce additional model parameters or change the original architecture. Considering the fact that our proposed method is task- and model-agnostic to be readily applied in various scenarios, we believe that our research holds significant ramifications for the field of hyperbolic representation learning. In future work, we will investigate the embedding in more expressive manifolds (Gu et al., 2019; Xiong et al., 2022b; 2023; 2021; Bachmann et al., 2020) and explore contrastive learning (Zhang et al., 2022a;b; Liu et al., 2022) to enhance the hierarchical learning.

Acknowledgements

We express gratitude to the anonymous reviewers and area chairs for their valuable comments and suggestions. This work was partly supported by grants from the National Key Research and Development Program of China (No. 2018AAA0100204) and the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14222922, RGC GRF, No. 2151185).

References

Abraham, I., Balakrishnan, M., Kuhn, F., Malkhi, D., Ramasubramanian, V., and Talwar, K. Reconstructing approximate tree metrics. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pp. 43–52, 2007.

Adcock, A. B., Sullivan, B. D., and Mahoney, M. W. Tree-like structure in large social and information networks. In *IEEE International Conference on Data Mining*, pp. 1–10. IEEE, 2013.

Anderson, R. M., Anderson, B., and May, R. M. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

Atigh, M. G., Schoep, J., Acar, E., van Noord, N., and Mettes, P. Hyperbolic image segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4453–4462, 2022.

Bachmann, G., Bécigneul, G., and Ganea, O. Constant curvature graph convolutional networks. In *International Conference on Machine Learning*, pp. 486–496. PMLR, 2020.

Bai, Q., Guo, J., Zhang, H., Nie, C., Zhang, L., and Yuan, X. H^2tne: Temporal heterogeneous information network embedding in hyperbolic spaces. In *International Semantic Web Conference*, pp. 179–195, 2022.

Bai, Q., Nie, C., Zhang, H., Zhao, D., and Yuan, X. Hgwavenet: A hyperbolic graph neural network for temporal link prediction. *arXiv preprint arXiv:2304.07302*, 2023.

Bai, Y., Ying, Z., Ren, H., and Leskovec, J. Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. *Advances in Neural Information Processing Systems*, 34:12316–12327, 2021.

Bécigneul, G. and Ganea, O.-E. Riemannian adaptive optimization methods. *arXiv preprint arXiv:1810.00760*, 2018.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4868–4879, 2019.

Chami, I., Wolf, A., Juan, D.-C., Sala, F., Ravi, S., and Ré, C. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6901–6914, 2020.

Chen, Y., Yang, M., Zhang, Y., Zhao, M., Meng, Z., Hao, J., and King, I. Modeling scale-free graphs with hyperbolic geometry for knowledge-aware recommendation. In *ACM International Conference on Web Search and Data Mining*, pp. 94–102, 2022.

Chepoi, V., Dragan, F., Estellon, B., Habib, M., and Vaxès, Y. Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pp. 59–68, 2008.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 2022.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, pp. 5345–5355, 2018.

Gu, A., Sala, F., Gunel, B., and Ré, C. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., et al. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1025–1035, 2017.

Hsu, J., Gu, J., Wu, G., Chiu, W., and Yeung, S. Capturing implicit hierarchical structure in 3d biomedical images with self-supervised hyperbolic representations. *Advances in Neural Information Processing Systems*, 34: 5112–5123, 2021.

Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., and Lempitsky, V. Hyperbolic image embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6418–6428, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Krioukov, D., Papadopoulos, F., Vahdat, A., and Boguná, M. Curvature and temperature of complex networks. *Physical Review E*, 80(3):035101, 2009.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguná, M. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Law, M., Liao, R., Snell, J., and Zemel, R. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pp. 3672–3681. PMLR, 2019.

Lee, J. M. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.

Lee, J. M. Smooth manifolds. In *Introduction to Smooth Manifolds*, pp. 1–31. Springer, 2013.

Li, B., Zhou, M., Zhang, S., Yang, M., Lian, D., and Huang, Z. BSAL: A framework of bi-component structure and attribute learning for link prediction. *arXiv preprint arXiv:2204.09508*, 2022.

Liu, J., Yang, M., Zhou, M., Feng, S., and Fournier-Viger, P. Enhancing hyperbolic graph embeddings via contrastive learning. *arXiv preprint arXiv:2201.08554*, 2022.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 8230–8241, 2019.

Mettes, P., Atigh, M. G., Keller-Ressel, M., Gu, J., and Yeung, S. Hyperbolic deep learning in computer vision: A survey. *arXiv preprint arXiv:2305.06611*, 2023.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, pp. 6338–6347, 2017.

Nickel, M. and Kiela, D. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pp. 3779–3788, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Peng, W., Varanka, T., Mostafa, A., Shi, H., and Zhao, G. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Saberi, M., Khosrowabadi, R., Khatibi, A., Misic, B., and Jafari, G. Topological impact of negative links on the stability of resting-state brain network. *Scientific reports*, 11(1):1–14, 2021.

Saitou, N. and Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

Sala, F., De Sa, C., Gu, A., and Re, C. Representation tradeoffs for hyperbolic embeddings. In *International Conference on Machine Learning*, pp. 4460–4469, 2018.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pp. 355–366. Springer, 2011.

Shimizu, R., Mukuta, Y., and Harada, T. Hyperbolic neural networks++. *arXiv preprint arXiv:2006.08210*, 2020.

Sonthalia, R. and Gilbert, A. Tree! i am no tree! i am a low dimensional hyperbolic embedding. *Advances in Neural Information Processing Systems*, 33:845–856, 2020.

Spivak, M. A comprehensive introduction to differential geometry. *Bull. Amer. Math. Soc*, 79:303–306, 1973.

Sun, J., Cheng, Z., Zuberi, S., Pérez, F., and Volkovs, M. Hgcf: Hyperbolic graph convolution networks for collaborative filtering. In *Proceedings of the Web Conference*, pp. 593–601, 2021a.

Sun, L., Zhang, Z., Zhang, J., Wang, F., Peng, H., Su, S., and Philip, S. Y. Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4375–4383, 2021b.

Sun, Z., Chen, M., Hu, W., Wang, C., Dai, J., and Zhang, W. Knowledge association with hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2010.02162*, 2020.

Suzuki, A., Nitanda, A., Wang, J., Xu, L., Yamanishi, K., and Cavazza, M. Generalization error bound for hyperbolic ordinal embedding. In *International Conference on Machine Learning*, pp. 10011–10021. PMLR, 2021a.

Suzuki, A., Nitanda, A., Xu, L., Yamanishi, K., Cavazza, M., et al. Generalization bounds for graph embedding using negative sampling: Linear vs hyperbolic. *Advances in Neural Information Processing Systems*, 34:1243–1255, 2021b.

Ungar, A. A. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

Ungar, A. A. et al. The hyperbolic square and mobius transformations. *Banach Journal of Mathematical Analysis*, 1 (1):101–116, 2007.

van den Heuvel, M. P. and Sporns, O. Network hubs in the human brain. *Trends in cognitive sciences*, 17(12): 683–696, 2013.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Vinh Tran, L., Tay, Y., Zhang, S., Cong, G., and Li, X. HyperML: A boosting metric learning approach in hyperbolic space for recommender systems. In *ACM International Conference on Web Search and Data Mining*, pp. 609–617, New York, NY, USA, 2020.

Vyas, A., Choudhary, N., Khatir, M., and Reddy, C. K. Graphzoo: A development toolkit for graph neural networks with hyperbolic geometries. In *Proceedings of the Companion Proceedings of the Web Conference, Lyon, France*, pp. 25–29, 2022.

Wang, H., Lian, D., Tong, H., Liu, Q., Huang, Z., and Chen, E. Hypersorec: Exploiting hyperbolic user and item representations with multiple aspects for social-aware recommendation. *TOIS*, pp. 1–28, 2021.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019.

Xiong, B., Zhu, S., Potyka, N., Pan, S., Zhou, C., and Staab, S. Pseudo-riemannian graph convolutional networks. *arXiv preprint arXiv:2106.03134*, 2021.

Xiong, B., Cochez, M., Nayyeri, M., and Staab, S. Hyperbolic embedding inference for structured multi-label prediction. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, 2022a.

Xiong, B., Zhu, S., Nayyeri, M., Xu, C., Pan, S., Zhou, C., and Staab, S. Ultrahyperbolic knowledge graph embeddings. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2130–2139, 2022b.

Xiong, B., Nayyeri, M., Jin, M., He, Y., Cochez, M., Pan, S., and Staab, S. Geometric relational embeddings: A survey. *arXiv preprint arXiv:2304.11949*, 2023.

Yang, H., Chen, H., Li, L., Yu, P. S., and Xu, G. Hyper meta-path contrastive learning for multi-behavior recommendation. *arXiv preprint arXiv:2109.02859*, 2021a.

Yang, M., Meng, Z., and King, I. Featurenorm: L2 feature normalization for dynamic graph embedding. In *2020 IEEE International Conference on Data Mining*, pp. 731–740. IEEE, 2020.

Yang, M., Zhou, M., Kalander, M., Huang, Z., and King, I. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1975–1985, 2021b.

Yang, M., Li, Z., Zhou, M., Liu, J., and King, I. HICF: Hyperbolic informative collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2212–2221, 2022a.

Yang, M., Zhou, M., Li, Z., Liu, J., Pan, L., Xiong, H., and King, I. Hyperbolic graph neural networks: A review of methods and applications. *arXiv preprint arXiv:2202.13852*, 2022b.

Yang, M., Zhou, M., Liu, J., Lian, D., and King, I. HRCF: Enhancing collaborative filtering via hyperbolic geometric regularization. In *Proceedings of the Web Conference*, 2022c.

Yang, M., Zhou, M., Pan, L., and King, I. Hyperbolic curvature graph neural network. *arXiv preprint arXiv:2212.01793*, 2022d.

Zhang, J., Shi, X., Zhao, S., and King, I. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. *arXiv preprint arXiv:1905.13129*, 2019.

Zhang, R., Khan, A. A., and Grossman, R. L. Evaluation of hyperbolic attention in histopathology images. In *BIBE*, pp. 773–776, 2020. doi: 10.1109/BIBE50027.2020.00131.

Zhang, Y., Wang, X., Shi, C., Jiang, X., and Ye, Y. F. Hyperbolic graph attention network. *IEEE Transactions on Big Data*, 2021a.

Zhang, Y., Wang, X., Shi, C., Liu, N., and Song, G. Lorentzian graph convolutional networks. In *Proceedings of the Web Conference*, pp. 1249–1261, 2021b.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. COSTA: Covariance-preserving feature augmentation for graph contrastive learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2524–2534, 2022a.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. Spectral feature augmentation for graph contrastive learning and beyond. *arXiv preprint arXiv:2212.01026*, 2022b.

Zhou, M., Li, B., Yang, M., and Pan, L. Telegraph: A benchmark dataset for hierarchical link prediction. *arXiv preprint arXiv:2204.07703*, 2022a.

Zhou, M., Yang, M., Pan, L., and King, I. Hyperbolic graph representation learning: A tutorial. *arXiv preprint arXiv:2211.04050*, 2022b.

Appendix

A. Riemannian Geometry

In this section, we present the details about the concepts mentioned in the work, i.e., geodesics, distance function, maps and parallel transport of a Riemannian geometry $(\mathcal{M}, g)$ and summarize the formula of hyperbolic space used in the study.

Geodesics and Distance Function. For a curve $\gamma : [\alpha, \beta] \to \mathcal{M}$, the length of γ, called geodesics, is defined as $L(\gamma) = \int_\alpha^\beta \|\gamma'(t)\|_g dt$. Then the distance of $\mathbf{u}, \mathbf{v} \in \mathcal{M}$ is given by $d_\mathcal{M}(\mathbf{u}, \mathbf{v}) = \inf L(\gamma)$ where γ is a curve that $\gamma(\alpha) = \mathbf{u}, \gamma(\beta) = \mathbf{v}$.

Maps and Parallel Transport. With assuming the manifolds are smooth, i.e. the maps are diffeomorphic, the map defines the projection between the manifold and the tangent space. For a point $\mathbf{x} \in \mathcal{M}$ and a vector $\mathbf{v} \in \mathcal{T}_\mathbf{x}\mathcal{M}$, there exists a unique geodesic $\gamma : [0, 1] \to \mathcal{M}$ where $\gamma(0) = \mathbf{x}, \gamma'(0) = \mathbf{v}$. The exponential map $\exp_\mathbf{x} : \mathcal{T}_\mathbf{x}\mathcal{M} \to \mathcal{M}$ is defined as $\exp_\mathbf{x}(\mathbf{v}) = \gamma(1)$ and logarithmic map $\log_\mathbf{x} : \mathcal{M} \to \mathcal{T}_\mathbf{x}\mathcal{M}$ is the inverse of $\exp_\mathbf{x}$. The parallel transport $PT_{\mathbf{x}\to\mathbf{y}} : \mathcal{T}_\mathbf{x}\mathcal{M} \to \mathcal{T}_\mathbf{y}\mathcal{M}$ achieves the transportation from point $\mathbf{x}$ to $\mathbf{y}$ that preserves the metric tensors along the unique geodesic.

Hyperbolic Models. Riemannian manifolds with different curvatures define different geometries: elliptic geometry (positive curvature), Euclidean geometry (zero curvature), and hyperbolic geometry (negative curvature). Here, we focus on the negative curvature space, i.e., hyperbolic geometry. Hyperbolic geometry is a Riemannian manifold with a constant negative sectional curvature. There exist multiple equivalent hyperbolic models which show different characteristics but are mathematically equivalent. We here mainly consider two widely studied hyperbolic models: the Poincaré ball model (Nickel & Kiela, 2017) and the Lorentz model (also known as the hyperboloid model) (Nickel & Kiela, 2018).

Let $\|.\|$ be the Euclidean norm and $\langle ., . \rangle_\mathcal{L}$ represent the Minkowski inner product. The formulas and operations associated with distance, maps, and parallel transport, hyperbolic origin point are summarized in Table 6, where $\oplus_\kappa$ and $\text{gyr}[., .]v$ are Möbius addition and gyration operator (Ungar et al., 2007), respectively, which are given in the following.

Möbius addition: For $\mathbf{x}, \mathbf{y} \in \mathcal{B}_\kappa^n$, the Möbius addition (Ungar et al., 2007) is

$$\mathbf{x} \oplus_\kappa \mathbf{y} = \frac{\left(1 - 2\kappa\langle\mathbf{x}, \mathbf{y}\rangle_2 - \kappa\|\mathbf{y}\|_2^2\right)\mathbf{x} + \left(1 + \kappa\|\mathbf{x}\|_2^2\right)\mathbf{y}}{1 - 2\kappa\langle\mathbf{x}, \mathbf{y}\rangle_2 + \kappa^2\|\mathbf{x}\|_2^2\|\mathbf{y}\|_2^2}. \tag{12}$$

Then the induced Möbius substraction $\ominus_\kappa$ is given by $\mathbf{x} \ominus_\kappa \mathbf{y} = \mathbf{x} \oplus_\kappa (-\mathbf{y})$.

Gyration operator. In the theory of gyrogroups, the notion of the gyration operator (Ungar, 2008) is given by

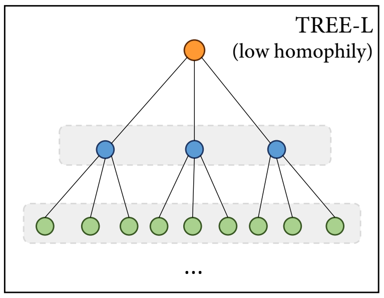
$$\text{gyr}[\mathbf{x}, \mathbf{y}]\mathbf{v} = \ominus_\kappa (\mathbf{x} \oplus_\kappa \mathbf{y}) \oplus_\kappa (\mathbf{x} \oplus_\kappa (\mathbf{y} \oplus_\kappa \mathbf{v})).$$

Table 6. Summary of operations in the Poincaré ball model and the Lorentz model ($\kappa < 0$)

	Poincaré Ball Model	**Lorentz Model (Hyperboloid Model)**
Manifold	$\mathcal{B}_\kappa^n = \left\{\mathbf{x} \in \mathbb{R}^n : \langle\mathbf{x}, \mathbf{x}\rangle_2 < -\frac{1}{\kappa}\right\}$	$\mathcal{L}_\kappa^n = \left\{\mathbf{x} \in \mathbb{R}^{n+1} : \langle\mathbf{x}, \mathbf{x}\rangle_\mathcal{L} = \frac{1}{\kappa}\right\}$
Metric	$g_\mathbf{x}^{\mathcal{B}_\kappa} = (\lambda_\mathbf{x}^\kappa)^2 \mathbf{I}_n$ where $\lambda_\mathbf{x}^\kappa = \frac{2}{1+\kappa\|\mathbf{x}\|_2^2}$	$g_\mathbf{x}^{\mathcal{L}_\kappa} = \eta,$ where η is I except $\eta_{0,0} = -1$
Distance	$d_\mathcal{B}^\kappa(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{\lvert\kappa\rvert}} \cosh^{-1}\left(1 - \frac{2\kappa\|\mathbf{x}-\mathbf{y}\|_2^2}{\left(1+\kappa\|\mathbf{x}\|_2^2\right)\left(1+\kappa\|\mathbf{y}\|_2^2\right)}\right)$	$d_\mathcal{L}^\kappa(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{\lvert\kappa\rvert}} \cosh^{-1}(\kappa\langle\mathbf{x}, \mathbf{y}\rangle_\mathcal{L})$
Logarithmic Map	$\log_\mathbf{x}^\kappa(\mathbf{y}) = \frac{2}{\sqrt{\lvert\kappa\rvert\lambda^\kappa}} \tanh^{-1}\left(\sqrt{\lvert\kappa\rvert}\,\|-\mathbf{x} \oplus_\kappa \mathbf{y}\|_2\right) \frac{-\mathbf{x}\oplus_\kappa\mathbf{y}}{\|-\mathbf{x}\oplus_\kappa\mathbf{y}\|_2}$	$\log_\mathbf{x}^\kappa(\mathbf{y}) = \frac{\cosh^{-1}(\kappa\langle\mathbf{x},\mathbf{y}\rangle_\mathcal{L})}{\sinh\left(\cosh^{-1}(\kappa\langle\mathbf{x},\mathbf{y}\rangle_\mathcal{L})\right)}(\mathbf{y} - \kappa\langle\mathbf{x}, \mathbf{y}\rangle_\mathcal{L}\mathbf{x})$
Exponential Map	$\exp_\mathbf{x}^\kappa(\mathbf{v}) = \mathbf{x} \oplus_\kappa \left(\tanh\left(\sqrt{\lvert\kappa\rvert}\frac{\lambda_\mathbf{x}^\kappa\|\mathbf{v}\|_2}{2}\right)\frac{\mathbf{v}}{\sqrt{\lvert\kappa\rvert}\|\mathbf{v}\|_2}\right)$	$\exp_\mathbf{x}^\kappa(\mathbf{v}) = \cosh\left(\sqrt{\lvert\kappa\rvert}\|\mathbf{v}\|_\mathcal{L}\right)\mathbf{x} + \mathbf{v}\frac{\sinh\left(\sqrt{\lvert\kappa\rvert}\|\mathbf{v}\|_\mathcal{L}\right)}{\sqrt{\lvert\kappa\rvert}\|\mathbf{v}\|_\mathcal{L}}$
Parallel Transport	$PT_{\mathbf{x}\to\mathbf{y}}^\kappa(\mathbf{v}) = \frac{\lambda_\mathbf{x}^\kappa}{\lambda_\mathbf{y}^\kappa}\text{gyr}[\mathbf{y}, -\mathbf{x}]v$	$PT_{\mathbf{x}\to\mathbf{y}}^\kappa(\mathbf{v}) = \mathbf{v} - \frac{\kappa\langle\mathbf{y},\mathbf{v}\rangle_\mathcal{L}}{1+\kappa\langle\mathbf{x},\mathbf{y}\rangle_\mathcal{L}}(\mathbf{x} + \mathbf{y})$
Origin Point	$\mathbf{0}_n$	$[\frac{1}{\sqrt{\lvert\kappa\rvert}}, \mathbf{0}_n]$

B. Dataset Construction about TREE-L and TREE-H

The geometric information of a real-world dataset is not explicitly given and the dataset is also not a pure-tree structure in general, so we synthesize two tree datasets, TREE-L and TREE-H, to track the position of node embedding in hyperbolic space. In particular, the two trees are with the same structures but with different homophily. All nodes except leaf nodes have three child nodes. In total, there are eight layers, 1092 edges, and 1093 nodes for each tree. On the tree with high homophily (i.e., TREE-H), nodes in each largest subtree (total, three largest subtrees) have the same node labels. We set the root node as another class, and then it is easy to know that there are four different classes. For the node features belonging to the same class, we generate a 32-dimension vector from a Gaussian distribution. The mean and variance of the four classes are (0.0, 1.0), (1.0, 1.0), (2.0, 1.0), (3.0, 1.0), respectively, where (0.0, 1.0) is for the class of root and the other three are for the rest. For TREE-L, on the other hand, nodes at the same level are with the same labels. We further set the first four layers to be the same class to ensure the total classes are four. Then for the four classes in TREE-L, we also use the same method, mean, and variance to generate the node features.



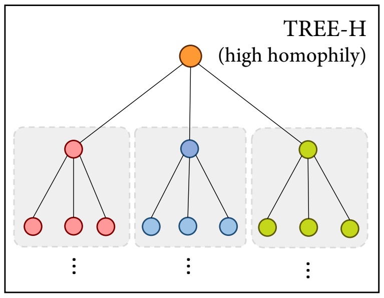


Figure 5. Illustration of the structure of synthetic TREE-L/H. In TREE-L, the node class in the same level is the same, while in TREE-H, the node class in each largest subtree is the same. For clarity, we use the same color to denote the same class in these two trees.

There is a metric proposed by Pei et al. (Pei et al., 2020) to measure the homophily rate H_m of a graph G, which is given by:

$$H_m(G) = \frac{1}{|V|}\sum_{v\in V}\frac{\text{\#Label}(v\text{'s neighbors})\text{==Label of } v}{\#v\text{'s neighbors}}. \tag{13}$$

Then, the computed homophily rates H_m of TREE-H and TREE-L are **0.998** and **0.018**, respectively.

C. Hyperbolic Midpoint Computation and Alignment

Let $\mathbf{Z}$ be the final layer embedding matrix consisting of n hyperbolic node vectors $(\mathbf{z}_1, \cdots, \mathbf{z}_i, \cdots, \mathbf{z}_n)$ where $\mathbf{z}_i \in \mathcal{H}^d_\kappa$, and let $v_i \in \mathbb{R}$ ($v_i \geq 0$ and $\sum_i v_i > 0$) be the weight for $\mathbf{z}_i$ and we set it as 1 in this study, the details on the computation of hyperbolic center $\mathbf{z}_c$ and alignment given as follows.

In the Poincaré ball model, the center, i.e., Möbius gyromidpoint (Ungar, 2008), is computed in gyrovector space, which is given by

$$\mathbf{z}_c := \frac{1}{2} \oplus_\kappa \left(\frac{\sum_{i=1}^n v_i \lambda^\kappa_{\mathbf{z}_i} \mathbf{z}_i}{\sum_{i=1}^n v_i \left(\lambda^\kappa_{\mathbf{z}_i} - 1\right)} \right), \tag{14}$$

where $\lambda^\kappa_{\mathbf{z}_i} = \frac{2}{1+\kappa\|\mathbf{z}_i\|_2^2}$. The root alignment operation is related to addition in Poincaré ball model which is computed by Equation (12).

In the Lorentz model (Law et al., 2019), the center (also called Lorentzian centroid) is computed by

$$\mathbf{z}_c := \frac{1}{\sqrt{|\kappa|}} \frac{\sum_{i=1}^n v_i \mathbf{z}_i}{|\, \|\sum_{i=1}^n v_i \mathbf{z}_i\|_{\mathcal{L}} \,|}, \tag{15}$$

The root alignment operation is related to addition operation in Lorentz model which is formulated as:

$$\mathbf{z}_i \oplus_\kappa (\mathbf{z}_c) := \exp^\kappa_{\bar{\mathbf{z}}}(PT_{\mathbf{o}\to\bar{\mathbf{z}}}(\log^\kappa_{\mathbf{o}}(\mathbf{z}_i))). \tag{16}$$

In the tangent space, the embedding $\mathbf{Z}$ is first projected to the tangent space at origin, that is $\mathbf{Z}^{\mathcal{T}} = \log^\kappa_{\mathbf{o}}(\mathbf{Z}^{\mathcal{T}})$. Then the center is defined by the following weighted manner,

$$\mathbf{z}^{\mathcal{T}}_c := \frac{\sum_{i=1}^n v_i \mathbf{z}^{\mathcal{T}}_i}{\sum_{i=1}^n v_i}, \tag{17}$$

The root alignment is given as,

$$\mathbf{z}_i^{\mathcal{T}} \oplus_\kappa (\mathbf{z}_c) := \mathbf{z}_i^{\mathcal{T}} - \mathbf{z}_c^{\mathcal{T}}. \tag{18}$$

D. Proof of Theorem

D.1. Proof of Theorem 4.1

The Möbius gyromidpoint for Poincaré ball is a solution of the minimization problem, which has been proved by Shimizu et al. (2020) and please check Theorem 2 in work (Shimizu et al., 2020). The Lorentzian centroid for Lorentz model also minimizes the weighted sum of the squared distance, which has been proved by (Law et al., 2019)(c.f.,Theorem 3.3)

D.2. Proof of Theorem 4.2

Proof. For the sake of clarity, we shall disregard the superscript $\mathcal{T}$ in the following proof. Let $\mathbf{z}_c = \frac{\sum_{i=1}^n v_i \mathbf{z}_i}{\sum_{i=1}^n v_i}$ denote the weighted center, where v_i represents the weight associated with $\mathbf{z}_i$, and we use v_s denote $\sum_{i=1}^n v_i$ for breity. Then we easily derive that,

$$\begin{aligned}
\mathbf{z}_c &= \frac{\sum_{i=1}^n v_i \mathbf{z}_i}{\sum_{i=1}^n v_i} \\
&\Rightarrow \mathbf{z}_c \cdot \sum_{i=1}^n v_i = \sum_{i=1}^n v_i \mathbf{z}_i \\
&\Rightarrow \mathbf{z}_c \cdot v_s = \sum_{i=1}^n v_i \mathbf{z}_i.
\end{aligned} \tag{19}$$

Furthermore, let $\mathbf{z}_a$ be a point in the tangent space. Then the minimization problem can be reformulated as:

$$\begin{aligned}
\min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i d^2(\mathbf{z}_i, \mathbf{z}_a) &= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i ||\mathbf{z}_i - \mathbf{z}_a||^2 \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \left(\|\mathbf{z}_i\|^2 + \|\mathbf{z}_a\|^2 - 2\mathbf{z}_i^T \mathbf{z}_a\right) \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 + \sum_{i=1}^n v_i \|\mathbf{z}_a\|^2 - \sum_{i=1}^n v_i \cdot 2\mathbf{z}_i^T \mathbf{z}_a \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 + v_s \|\mathbf{z}_a\|^2 - 2\sum_{i=1}^n v_i \cdot \mathbf{z}_i^T \mathbf{z}_a \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 + v_s \|\mathbf{z}_a\|^2 - 2v_s \cdot \mathbf{z}_c^T \mathbf{z}_a \quad \text{\%(refer to Equation (16))} \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 + v_s \left(\|\mathbf{z}_a\| - 2\mathbf{z}_c^T \mathbf{z}_a\right) \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 + v_s \left(\|\mathbf{z}_c - \mathbf{z}_a\|^2 - \|\mathbf{z}_c\|^2\right) \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 + v_s \|\mathbf{z}_c - \mathbf{z}_a\|^2 - v_s \|\mathbf{z}_c\|^2 \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \sum_{i=1}^n v_i \|\mathbf{z}_i\|^2 - v_s \|\mathbf{z}_c\|^2 + v_s \|\mathbf{z}_c - \mathbf{z}_a\|^2 \\
&= \min_{\mathbf{z}_a \in \mathcal{T}_{\mathbf{o}}\mathcal{H}_\kappa^d} \underbrace{\sum_{i=1}^n v_i \left(\|\mathbf{z}_i\|^2 - \|\mathbf{z}_c\|^2\right)}_{constants} + v_s \|\mathbf{z}_c - \mathbf{z}_a\|^2.
\end{aligned} \tag{20}$$

It is known that the first term in the equation are constant. Furthermore, it can be deduced that the last term, $v_s\|\mathbf{z}_c - \mathbf{z}_a\|^2$, is non-negative. It follows that the total distance sum to all other nodes is minimized when $\mathbf{z}_a = \mathbf{z}_c$. In other words, the embedding center, represented by $\mathbf{z}_c$, is the solution that minimizes the sum of distances to all other nodes.

□

E. Graph Centrality

In graph theory and network analysis, centrality metrics are utilized to assign numerical or ordinal values to nodes within a graph commensurate with their relative importance and prominence within the network. These metrics find wide-ranging applications, such as identifying key actors in a social network, critical infrastructure nodes in internet or urban networks, primary vectors of disease transmission, and salient nodes in brain networks (van den Heuvel & Sporns, 2013; Saberi et al., 2021). In this part, we present some important centrality in graphs for reference. Unless otherwise specified, the graph referred to here is assumed to be an undirected one.

Degree centrality. The degree centrality is a commonly employed metric in graph theory and network analysis, which quantifies the centrality of a node by calculating the number of edges incident upon it, commonly referred to as its degree, that is,

$$C_D(v) = \deg(v). \tag{21}$$

Computing degree centrality of all nodes takes $\Theta(|V|^2)$ in a dense adjacency matrix representation and $\Theta(|E|)$ in a sparse matrix representation. Nodes with a higher degree are considered more central in the network, and thus we select the node with the highest degree centrality as the root node for comparison.

Betweenness centrality. The betweenness centrality of a vertex is based on the number of shortest paths passing through it. This metric is computed by considering all pairs of vertices in a connected graph and identifying the number of shortest paths between them that pass through the given vertex, which is given by

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\delta st(v)}{\delta st}, \tag{22}$$

where δst represents the total number of shortest paths from vertex s to vertex t, and $\delta st(v)$ denotes the number of those paths that traverse vertex v. Computing betweenness centrality of all nodes involves the computation of shortest path between all pairs, which takes $\Theta(|V|^3)$ (Cormen et al., 2022) for general graphs, and $O(|V|^2 \log |V| + |V||E|)$ for sparse graphs. Similarly, we take the node with highest betweenness centrality for experiential comparison.

Closeness centrality. The closeness centrality quantifies how closely connected the node is to all other nodes in the network. The core concept of closeness centrality is a node is central if the average number of links needed to reach another node is small. It is computed by taking the reciprocal of the sum of the shortest path distances between the node and every other node in the graph, which can be formulated as,

$$C_C(v) = \frac{n-1}{\sum_{u=1}^{n-1} d(v,u)}, \tag{23}$$

where $d(v,u)$ is the shortest-path distance between v and u, and $n-1$ is the number of nodes reachable from v. The time complexity is similar to betweenness centrality since it requires to compute the shortest distance as well. Nodes with higher closeness centrality are considered more central as they are closer to all other nodes. Therefore, we treat the highest clustering centrality as the equivalent for comparisons.

F. Experimental Details

F.1. Statistics of Datasets

We list the statistics of the dataset used in our work in Table 7. In specific, DISEASE is simulated by the SIR disease spreading model (Anderson et al., 1992), where the label indicates whether the node was infected or not. The AIRPORT is a flight network where nodes are the airports, edges represent airline routes, and the label denotes the population of the country that the corresponding airport belongs to. CORA and CITESEER are standard benchmarks describing citation networks with nodes as scientific papers, edges as citations between them, and node labels as academic areas.

Table 7. Statistics of the datasets.

Dataset	Nodes	Edges	Classes	Node features
DISEASE (NC)	1,044	1,043	2	1,000
DISEASE (LP)	2,665	2,664	2	1000
AIRPORT	3,188	18,631	4	4
CITESEER	3,327	4,732	6	3,703
CORA	2,708	5,429	7	1,433
TREE-L	1,093	1,092	4	32
TREE-H	1,093	1,092	4	32

F.2. Training Details

(1) **Data split.** We conducted experiments on both node classification and link prediction tasks. For the link prediction task, we randomly split the edges in the DISEASE dataset into training (75%), validation (5%), and test (20%) sets for the shallow models. In the semi-supervised learning setting, we split the DISEASE dataset into training (25%), validation (5%), and test (70%) sets. For node classification, we split the nodes in the AIRPORT dataset into 70%, 15%, and 15%, and the nodes in the DISEASE dataset into 30%, 10%, and 60%. For the CORA and CITESEER datasets, we used 20 labeled examples per class for training. The above splits are the same as those used in previous works, except for the semi-supervised learning.

(2) **Implementation details.** For all models, we traverse the number of embedding dimensions from 8, 64, 256 and then perform a hyper-parameter search on a validation set over learning rate $\{0.01, 0.02, 0.005\}$, weight decay $\{1e-4, 5e-4, 5e-5\}$, dropout $\{0.1, 0.2, 0.5, 0.6\}$, and the number of layers $\{1, 2, 3, 4, 5\}$. We also adopt the early stopping strategies based on the validation set with patience in $\{100, 200, 500\}$.

(3) **Evaluation metric.** Following the literature, we report the F1-score for DISEASE and AIRPORT datasets and accuracy for the others in the node classification tasks. For the link predictions task, the Area Under Curve(AUC) is calculated. We report the results of 12 random experiments by removing the max and min values.

(4) **Loss functions.** In this study, we conducted two different experimental tasks, link prediction, and node classification. For link prediction, we used the following loss function as (Chami et al., 2019; Nickel & Kiela, 2017),

$$L_{\text{lp}} = \frac{1}{|E|}\sum_{(i,j)\in E} -\log p(\mathbf{z}_i, \mathbf{z}_j) + \frac{1}{|E|}\sum_{(i,j)\notin E'} \log p(\mathbf{z}_i, \mathbf{z}'_j), \tag{24}$$

where E is the edge set and $p(\cdot)$ is the Fermi-Dirac function, indicating the probability of two hyperbolic nodes $\mathbf{u}, \mathbf{v}$ have a link or not, which is defined as:

$$p(\mathbf{u}, \mathbf{v}) = \left[\exp\left(d_{\mathcal{H}}(\mathbf{u}, \mathbf{v})^2 - r\right)/t + 1\right]^{-1}, \tag{25}$$

The loss function is to maximize the probability of two nodes if they are linked in the training set while minimizing the probability of two nodes if they are not linked in the training set. We sample the same number of negative sampling for training.

For node classification, we follow the previous work (Chami et al., 2019), and classify the node by cross-entropy, which is formulated by,

$$L_{\text{ce}} = -\sum_{i\in|V|} q(\mathbf{z}_i)\log q(\mathbf{z}_i) \tag{26}$$

where

$$q(\mathbf{z}) = \text{Softmax}(\text{MLP}(\log^{\kappa}_{\mathbf{o}}(\mathbf{z}))) \tag{27}$$

Totally, our optimization target is to minimize the following loss function,

$$L = L_{\text{task}} + \lambda L_{\text{hyp}}, \tag{28}$$

where the task denotes the down-stream task, and λ is selected from $\{1, 0.1, 0.01, 0.001\}$

Table 8. Comparisons using stretching, alignment, opposite stretching and our proposed method HIE with dimension 64.

Dataset	HGCN	w/Stretching	w/Alignment	w/-Stretching	HIE (Ours)
DISEASE	90.6 ± 1.2	89.4 ± 2.5	95.5 ± 0.6	85.7 ± 4.8	**95.0** ± 0.8
AIRPORT	94.0 ± 0.6	93.6 ± 0.9	93.6 ± 0.3	92.1 ± 0.5	**94.1** ± 0.8
CITESEER	67.6 ± 0.4	68.5 ± 0.6	66.7 ± 0.9	18.1 ± 0.0	**74.1** ± 0.3
CORA	78.5 ± 0.6	81.5 ± 0.5	77.5 ± 1.0	31.9 ± 0.0	**83.0** ± 0.3

G. Experiments on Opposite Stretching

The capacity of hyperbolic space increases exponentially with the radius, in this work, we propose to push nodes away from the origin and thus introduce a origin-concerned geometric penalty. In this part, we supplement the opposite stretching operation, which pushes the node hidden state close to the origin, to further verify our motivation. Specifically, we use the following equation to achieve the opposite stretching:

$$L_{\text{hyp}}^{-} = \tanh\left(\frac{1}{|V|}\sum_{i \in V} w_i d_{\mathcal{H}}(\mathbf{z}_i^{\mathcal{H}}, \mathbf{o})\right). \tag{29}$$

We ran all experiments 12 times with the same settings as the previous, including the random seeds. The experimental results are shown in Table 8. To straightforwardly compare the performance of the opposite stretching and other related methods, we list the independent stretching (w/-Stretching), alignment (w/Alignment), and the proposed HIE together. The performance of using opposite stretching is shown in the ”w/Stretching” column.

It can be seen that the opposite stretching (w/-Stretching), or equivalently, encouraging the node embedding near the origin, is fatal to CORA and CITESEER where the performance drops to 31.9 and 18.1. At the same time, the performances on DISEASE and AIRPORT also drop dramatically. This is not difficult to understand since the embedding space near the origin is relatively small, and it is difficult for the model to obtain a better discriminative representation. This further verifies the effectiveness of our proposal.

H. More Results

H.1. HDO on Different Dimensions

Here we additionally supplement the HDO distribution on different dimensions, in Figure 6. It is easy to know that the overall HDO values become larger when applied with our HIE, which can be inferred from the shapes of HDO distribution or MEAN values. It is also observed that the number of the nodes whose embeddings are located near the boundary has increased significantly, as the shape of the HDO distribution looks more like a long-tailed distribution, which well matches the exponential increase in the capacity of the hyperbolic space.

We observe that the distribution of HDO of DISEASE is more concentrated in locations far from the origin in both our method and the original model, which mainly lies in the properties of the DISEASE dataset. The DISEASE dataset is pure tree-like, which has obvious hierarchies and can well match the hyperbolic space. Therefore, with hyperbolic embedding, the overall HDO is distributed far from the origin. This can explain the reason that only using centering for HGCN achieves considerable improvements on DISEASE, that is, the leaf nodes have already embedded near the boundary. However, it is also noted that the embedding position of ROOT (i.e., HC) is quite different: the ROOT in HGCN is far from the origin, while our proposal is at or near the origin. Our method is significantly better than the original HGCN. It further confirms that aligning the embedding center with the hyperbolic origin is of significance for the embedding quality and classification performance.

H.2. Hyperbolic Distance to HC

Besides, we further compute the hyperbolic distance from each node to the hyperbolic center in Figure 7. In the figure above, HC represents an absolute value indicating the distance from the center to the origin, while MIN, MEAN, and MAX represent relative values reflecting the distance to the HC. Our experiments have revealed that previous models are unable to capture the hierarchical relationships within the data from the HDC distribution effectively, as most of them are normally distributed. In contrast, our proposed approach consistently produced relatively larger values for the relative MIN, MEAN, and MAX, indicating that it can effectively fit the hyperbolic space.

H.3. Relative Hierarchies within Nodes

Table 9. Accuracy of relative hierarchies within nodes

Models	HGCN	Ours	HGCN	Ours
Dims	16	16	256	256
TREE-L	72.5%	75.4%	75.3%	77.6%
TREE-H	68.9%	80.0%	73.7%	83.5%

In the following, we compared hierarchies accuracy based on synthetic datasets, which provide easily accessible hierarchies, using both HGCN and our proposed method. We randomly selected 5000 pairs of nodes and computed their distance from the origin as a proxy for their respective hierarchical levels. We labeled a pairing as “1” if its hierarchical ordering matched the true relationship and “0” otherwise. Finally, we computed the accuracy of the sampled node pairs and presented the results in Table 9. The results demonstrate that our proposed method outperforms HGCN in ranking more pairs correctly, indicating its ability to effectively capture the hierarchical relationships within the data.

H.4. Similar Idea in Euclidean Space

The method proposed in this work is simple and effective. Can the same effect be achieved if the same idea is adopted in the Euclidean space? In the following, we conduct the same algorithm presented in Algorithm 1 within Euclidean space, termed as EIE, on the best Euclidean model GAT.

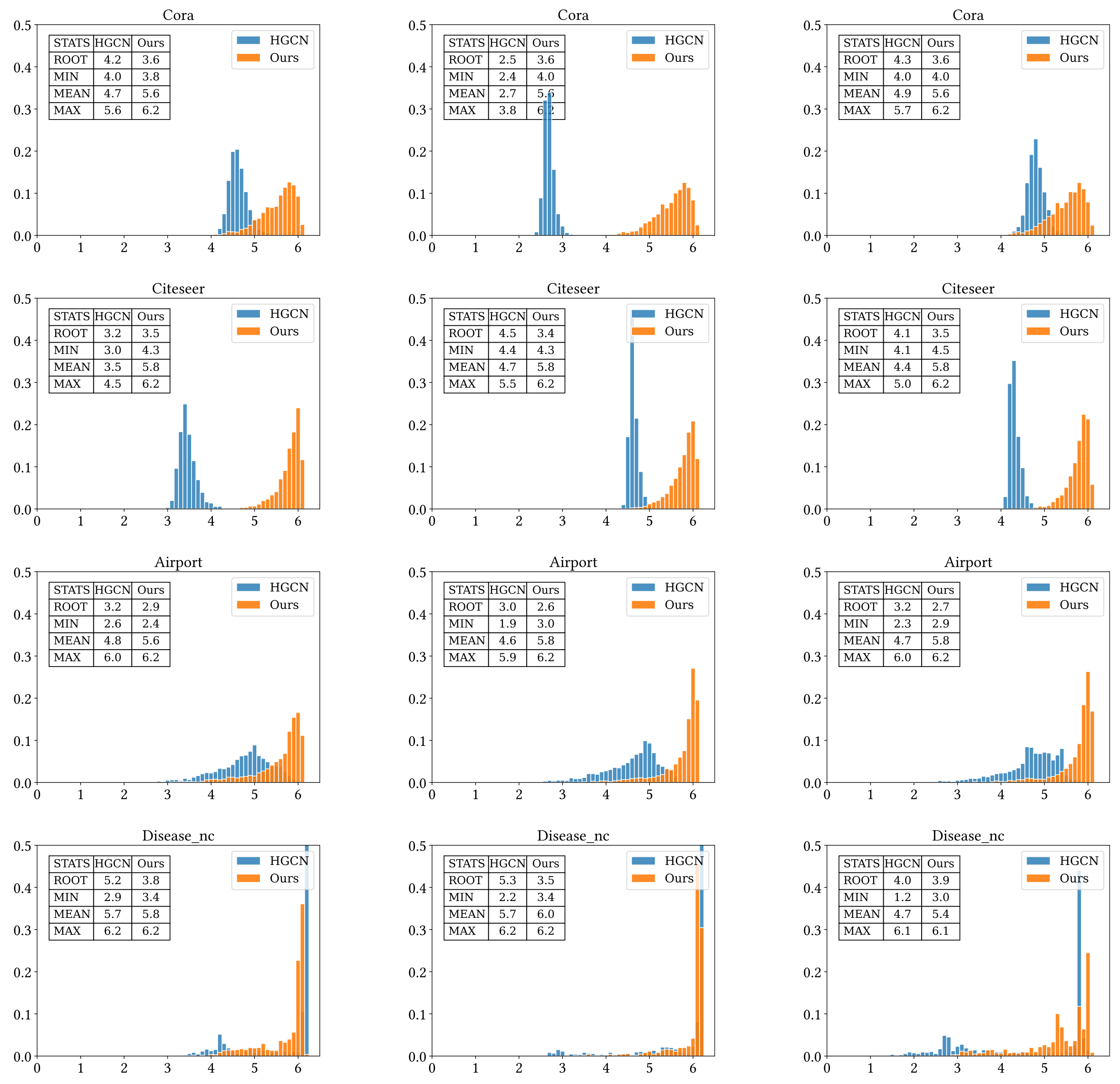
Table 10. Comparisons with Euclidean variants

Dataset	Disease	Disease	Airport	Airport	Citeseer	Citeseer	Cora	Cora
Dimension	16	256	16	256	16	256	16	256
GAT	74.4	80.2	78.2	90.4	70.9	71.7	82.5	83.0
GAT+EIL	79.1	81.5	78.6	90.8	71.2	72.2	82.8	83.0
HGAT	88.8	89.8	92.9	95.2	66.8	67.9	79.0	79.1
HGAT+HIL	93.7	94.0	93.0	95.7	73.4	74.4	83.0	83.4

The experiments are shown in Table 10 and the experiments revealed that penalizing the Euclidean-centered norm of nodes can prevent nodes from getting too close to each other, which can impact classifier performance or downstream tasks, resulting in minor improvements across most cases. However, the capacity limitations of Euclidean space lead to inferior outcomes compared to using HIE in hyperbolic space.

Moreover, we observed that while classification accuracy does not see significant improvement with increased dimensionality in Euclidean space, such an increase in hyperbolic space can enhance performance. These phenomena demonstrate important progress in hyperbolic space. First, HIE enhances the hyperbolic models’ embeddability, allowing them to fully utilize the spacious capacity of hyperbolic space and achieve superior performance across various datasets compared to Euclidean space. Second, as the embedding dimension increases, performance can still be improved, pushing the model to achieve better results, while their Euclidean counterparts show only minor improvements.

Overall, we believe that our study provides novel insights into developing innovative methodologies in hyperbolic space, and highlights the potential benefits of using hyperbolic geometry for machine learning tasks.



Cora (16)

STATS	HGCN	Ours
ROOT	4.2	3.6
MIN	4.0	3.8
MEAN	4.7	5.6
MAX	5.6	6.2

Cora (64)

STATS	HGCN	Ours
ROOT	2.5	3.6
MIN	2.4	4.0
MEAN	2.7	5.6
MAX	3.8	6.2

Cora (256)

STATS	HGCN	Ours
ROOT	4.3	3.6
MIN	4.0	4.0
MEAN	4.9	5.6
MAX	5.7	6.2

Citeseer (16)

STATS	HGCN	Ours
ROOT	3.2	3.5
MIN	3.0	4.3
MEAN	3.5	5.8
MAX	4.5	6.2

Citeseer (64)

STATS	HGCN	Ours
ROOT	4.5	3.4
MIN	4.4	4.3
MEAN	4.7	5.8
MAX	5.5	6.2

Citeseer (256)

STATS	HGCN	Ours
ROOT	4.1	3.5
MIN	4.1	4.5
MEAN	4.4	5.8
MAX	5.0	6.2

Airport (16)

STATS	HGCN	Ours
ROOT	3.2	2.9
MIN	2.6	2.4
MEAN	4.8	5.6
MAX	6.0	6.2

Airport (64)

STATS	HGCN	Ours
ROOT	3.0	2.6
MIN	1.9	3.0
MEAN	4.6	5.8
MAX	5.9	6.2

Airport (256)

STATS	HGCN	Ours
ROOT	3.2	2.7
MIN	2.3	2.9
MEAN	4.7	5.8
MAX	6.0	6.2

Disease_nc (16)

STATS	HGCN	Ours
ROOT	5.2	3.8
MIN	2.9	3.4
MEAN	5.7	5.8
MAX	6.2	6.2

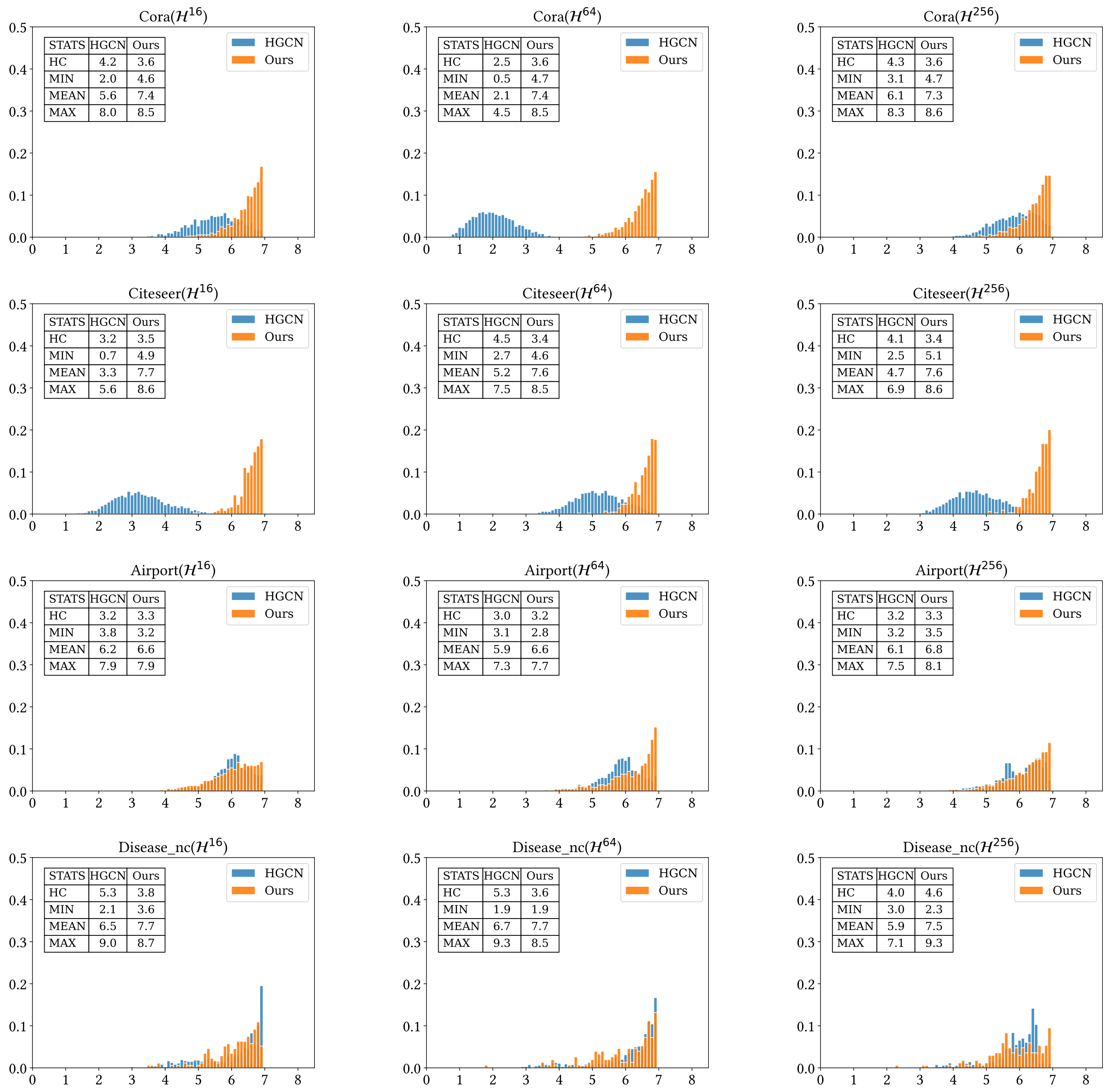
Disease_nc (64)

STATS	HGCN	Ours
ROOT	5.3	3.5
MIN	2.2	3.4
MEAN	5.7	6.0
MAX	6.2	6.2

Disease_nc (256)

STATS	HGCN	Ours
ROOT	4.0	3.9
MIN	1.2	3.0
MEAN	4.7	5.4
MAX	6.1	6.1

Figure 6. Illustration HDO distribution on CORA, CITESEER, AIRPORT, Disease where x-axis denotes the value of HDO and y-axis is the corresponding ratio. The figures in the first, second, and third column denotes the dimension 16, 64 and 256, respectively.



Cora($\mathcal{H}^{16}$)

STATS	HGCN	Ours
HC	4.2	3.6
MIN	2.0	4.6
MEAN	5.6	7.4
MAX	8.0	8.5

Cora($\mathcal{H}^{64}$)

STATS	HGCN	Ours
HC	2.5	3.6
MIN	0.5	4.7
MEAN	2.1	7.4
MAX	4.5	8.5

Cora($\mathcal{H}^{256}$)

STATS	HGCN	Ours
HC	4.3	3.6
MIN	3.1	4.7
MEAN	6.1	7.3
MAX	8.3	8.6

Citeseer($\mathcal{H}^{16}$)

STATS	HGCN	Ours
HC	3.2	3.5
MIN	0.7	4.9
MEAN	3.3	7.7
MAX	5.6	8.6

Citeseer($\mathcal{H}^{64}$)

STATS	HGCN	Ours
HC	4.5	3.4
MIN	2.7	4.6
MEAN	5.2	7.6
MAX	7.5	8.5

Citeseer($\mathcal{H}^{256}$)

STATS	HGCN	Ours
HC	4.1	3.4
MIN	2.5	5.1
MEAN	4.7	7.6
MAX	6.9	8.6

Airport($\mathcal{H}^{16}$)

STATS	HGCN	Ours
HC	3.2	3.3
MIN	3.8	3.2
MEAN	6.2	6.6
MAX	7.9	7.9

Airport($\mathcal{H}^{64}$)

STATS	HGCN	Ours
HC	3.0	3.2
MIN	3.1	2.8
MEAN	5.9	6.6
MAX	7.3	7.7

Airport($\mathcal{H}^{256}$)

STATS	HGCN	Ours
HC	3.2	3.3
MIN	3.2	3.5
MEAN	6.1	6.8
MAX	7.5	8.1

Disease_nc($\mathcal{H}^{16}$)

STATS	HGCN	Ours
HC	5.3	3.8
MIN	2.1	3.6
MEAN	6.5	7.7
MAX	9.0	8.7

Disease_nc($\mathcal{H}^{64}$)

STATS	HGCN	Ours
HC	5.3	3.6
MIN	1.9	1.9
MEAN	6.7	7.7
MAX	9.3	8.5

Disease_nc($\mathcal{H}^{256}$)

STATS	HGCN	Ours
HC	4.0	4.6
MIN	3.0	2.3
MEAN	5.9	7.5
MAX	7.1	9.3

Figure 7. Illustration hyperbolic distance to center (HDC) distribution on CORA, CITESEER, AIRPORT, Disease where x-axis denotes the value of HDC and y-axis is the corresponding ratio. The figures in the first, second, and third column denotes the dimension 16, 64 and 256, respectively.