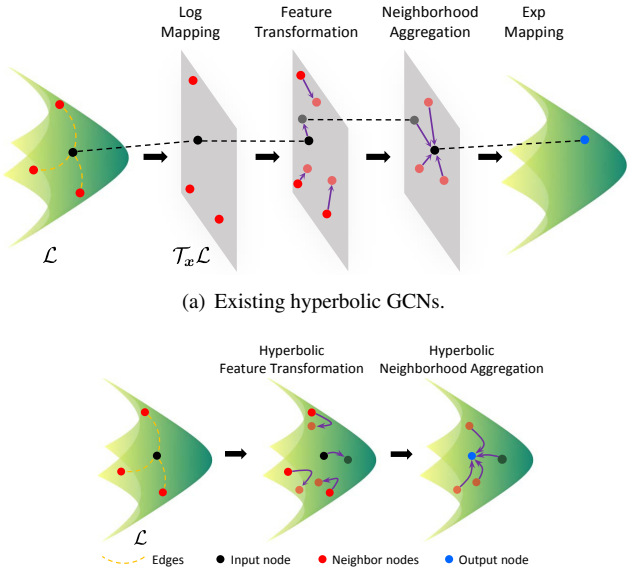
# A Hyperbolic-to-Hyperbolic Graph Convolutional Network

Jindou Dai, Yuwei Wu*, Zhi Gao, and Yunde Jia
Laboratory of Intelligent Information Technology, School of Computer Science,
Beijing Institute of Technology (BIT), Beijing, 100081, China.
{daijindou, wuyuwei, gaozhi_2017, jiayunde}@bit.edu.cn

## Abstract

*Hyperbolic graph convolutional networks (GCNs) demonstrate powerful representation ability to model graphs with hierarchical structure. Existing hyperbolic GCNs resort to tangent spaces to realize graph convolution on hyperbolic manifolds, which is inferior because tangent space is only a local approximation of a manifold. In this paper, we propose a hyperbolic-to-hyperbolic graph convolutional network (H2H-GCN) that directly works on hyperbolic manifolds. Specifically, we developed a manifold-preserving graph convolution that consists of a hyperbolic feature transformation and a hyperbolic neighborhood aggregation. The hyperbolic feature transformation works as linear transformation on hyperbolic manifolds. It ensures the transformed node representations still lie on the hyperbolic manifold by imposing the orthogonal constraint on the transformation sub-matrix. The hyperbolic neighborhood aggregation updates each node representation via the Einstein midpoint. The H2H-GCN avoids the distortion caused by tangent space approximations and keeps the global hyperbolic structure. Extensive experiments show that the H2H-GCN achieves substantial improvements on the link prediction, node classification, and graph classification tasks.*




(a) Existing hyperbolic GCNs.

(b) The proposed H2H-GCN.

Figure 1. Comparisons of exisiting hyperbolic GCNs and the proposed H2H-GCN. At the $\ell$-th layer, (a) existing hyperbolic GCNs performs Euclidean graph convolutional opeartions, *e.g.*, feature transformation and neighborhood aggregation, in the tangent space $\mathcal{T}_x\mathcal{L}$ that is a local approximation of the hyperbolic manifold $\mathcal{L}$; (b) H2H-GCN directly performs a hyperbolic feature transformation and a hyperbolic neighborhood aggregation on the hyperbolic manifold to learn node representations, keeping the global hyperbolic structure.

## 1. Introduction

Graph convolutional networks (GCNs) have attracted increasing attention for graph representation learning, where nodes in a graph are typically embedded into Euclidean spaces [25, 48, 46, 42, 20, 21]. Several works reveal that many graphs, such as social networks and citation networks, exhibit a highly hierarchical structure [11, 26, 32]. Recent studies have shown that hyperbolic spaces can well capture such hierarchical structure compared to Euclidean spaces [30, 31, 12]. Different from Euclidean spaces with zero curvature, hyperbolic spaces possess a constant negative curvature, which allows for an exponential growth of space volume with radius. This property of hyperbolic spaces pretty meets the requirements of hierarchical data (*e.g.*, trees) that need an exponential amount of space for branching, and encourages the development of GCNs in hyperbolic spaces to capture the hierarchical structure underlying graphs.

Existing hyperbolic GCNs [27, 10, 49] resort to tangent spaces to realize graph convolution in hyperbolic spaces. Since the hyperbolic space is a Riemannian manifold rather than a vector space, basic operations (such as matrix-vector multiplication and vector addition) well defined in Eu-

*Corresponding author

clidean spaces are not applicable in hyperbolic space. To generalize graph convolution to the hyperbolic space, the works in [27, 10, 49] first flatten a hyperbolic manifold, and then apply Euclidean graph convolutional operations in the tangent space. The results are projected back to the hyperbolic manifold. The procedures follow a manifold-tangent-manifold scheme, as shown in Figure 1(a). These methods has promoted the development of GCNs in hyperbolic spaces and achieved good performance. However, the mapping between the manifold and the tangent space is only locally diffeomorphic, which may distort the global structure of the hyperbolic manifold, especially frequently using tangent space approximations [23, 39].

In this paper, we propose to design a hyperbolic GCN that directly works on the hyperbolic manifold to keep global hyperbolic structure, rather than relying on the tangent space. This requires that each step of graph convolution, *e.g*., feature transformation and neighborhood aggregation, satisfies a manifold-to-manifold principle. To this end, we present a hyperbolic-to-hyperbolic graph convolutional network (H2H-GCN), where graph convolutional operations are directly conducted on the hyperbolic manifold.

Specifically, we developed a manifold-preserving graph convolution consisting of a hyperbolic feature transformation and a hyperbolic neighborhood aggregation. The hyperbolic feature transformation plays the role of linear transformation on hyperbolic manifolds, which requires multiplication of node representations by a transformation matrix. We constrain the transformation matrix to be a block diagonal matrix composed of a scalar 1 and an orthogonal matrix to ensure the transformed node representations still reside on the hyperbolic manifold. For hyperbolic neighborhood aggregation, we adopt the Einstein midpoint as the weighted message of neighbor nodes to update a node representation. Figure 1(b) depicts that H2H-GCN directly carries out the two steps on hyperbolic manifolds. In contrast to existing hyperbolic GCNs, the proposed H2H-GCN can avoid the distortion caused by tangent space approximations and keep the global hyperbolic structure underlying graphs. We summarize the contributions of this paper as follows.

- We propose a hyperbolic-to-hyperbolic graph convolutional network that directly performs graph convolution on hyperbolic manifolds, keeping the global hyperbolic structure underlying graphs. To the best of our knowledge, this is the first hyperbolic GCN without relying on tangent spaces.

- We developed a hyperbolic feature transformation that is a linear transformation on hyperbolic manifolds. The manifold constraint on the transformed hyperbolic representations is ensured by imposing the orthogonal constraint on the transformation sub-matrix.

## 2. Related Work

GCNs generalize classical convolutional neural networks to graph domains. To realize the convolution on graphs, there are two types of GCNs. Spectral-based GCNs [7, 13, 25, 22] are based on the convolution theorem to perform convolution by transforming graph signals into the spectral domain via the graph Fourier transform. Spatial-based GCNs [20, 21, 41, 33, 42, 46] update node representations by aggregating the message from its neighbor nodes, just like applying convolutional kernel on a local image patch. Despite a solid theoretical foundation of spectral-based GCNs, spatial-based GCNs have shown more superiorities due to efficiency, generality and flexibility.

Researchers discovered that many graphs, *e.g*., social networks and biological networks, usually exhibit a highly hierarchical structure [26, 32]. Krioukov *et al*. [26] pointed that the properties of strong clustering and power-law degree distribution in such graphs can be explained as a hidden hierarchy. Recent works have demonstrated powerful representation ability of hyperbolic spaces to model hierarchies that underlie taxonomies [30, 31], knowledge graphs [38, 2], images [24], semantic classes [28], actions [29], *etc* [37, 9, 44], achieving promising performance. Liu *et al*. [27] and Chami *et al*. [10] proposed hyperbolic GCNs that extend GCNs to hyperbolic spaces to capture the hierarchy underlying graphs. The main difference with our work is that they perform Euclidean graph convolutional operations in the tangent space, following a manifold-tangent-manifold scheme. The proposed H2H-GCN developed a hyperbolic graph convolution in the hyperbolic space without relying on tangent spaces. We designed a Lorentz linear transformation for feature transformation on the Lorentz model, and adopted Einstein midpoint to calculate manifold statistics [17] as aggregation function. We claim that such a manifold-to-manifold learning principle can avoid the distortion caused by tangent space approximations and keep the global hyperbolic structure, that is beneficial to graph representation learning.

## 3. Preliminaries

### 3.1. Hyperbolic Spaces

A Riemannian manifold $(\mathcal{M}, g)$ is a differentiable manifold $\mathcal{M}$ equipped with a metric tensor $g$. It can be locally approximated to a linear Euclidean space at an arbitrary point $\boldsymbol{x} \in \mathcal{M}$, and the approximated space is termed as a tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. Hyperbolic spaces are smooth Riemannian manifolds with a constant negative curvature [4]. There are five isometric models of hyperbolic spaces: the Lorentz model (a.k.a the hyperbolid model), the Klein model, the Jemisphere model, the Poincaré ball model, and the Poincaré half-space model [8]. In this paper, we choose the Lorentz model due to its numerical stability [31].

Formally, the Lorentz model of an $n$-dimensional hyperbolic space is defined by the manifold $\mathcal{L} = \{\boldsymbol{x} = [x_0, x_1, \cdots, x_n] \in \mathbb{R}^{n+1} : \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = -1, \boldsymbol{x}_0 > 0\}$ endowed with the metric tensor $g = \mathrm{diag}([-1, \mathbf{1}_n^\top])$ where $\mathrm{diag}(\cdot)$ function transforms a vector to a diagonal matrix. The Lorentz inner product induced by $g$ is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = \boldsymbol{x}^\top g \boldsymbol{y} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i. \tag{1}$$

In the following, we describe necessary operations.

**Distance.** The distance on a manifold is termed as a geodesic that is commonly a curve representing the shortest path between two nodes. For $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$, the distance between them is given by

$$d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{arcosh}(-\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}), \tag{2}$$

where $\mathrm{arcosh}(\cdot)$ is the inverse hyperbolic cosine function.

**Exponential and logarithmic maps.** An exponential map $\exp_{\boldsymbol{x}}(\boldsymbol{v})$ is the function projecting a tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ onto $\mathcal{M}$. A logarithmic map projects vectors on the manifold back to the tangent space satisfying $\log_{\boldsymbol{x}}(\exp_{\boldsymbol{x}}(\boldsymbol{v})) = \boldsymbol{v}$. For $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$, and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{L}$ the exponential map $\exp_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{L} \to \mathcal{L}$ and the logarithmic map $\log_{\boldsymbol{x}} : \mathcal{L} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{L}$ are given by

$$\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \cosh(\|\boldsymbol{v}\|_{\mathcal{L}})\boldsymbol{x} + \sinh(\|\boldsymbol{v}\|_{\mathcal{L}})\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_{\mathcal{L}}}, \tag{3}$$

$$\log_{\boldsymbol{x}}(\boldsymbol{y}) = \frac{\mathrm{arcosh}(-\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}})}{\sqrt{\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}^2 - 1}}(\boldsymbol{y} + \langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} \boldsymbol{x}), \tag{4}$$

where $\|\boldsymbol{v}\|_{\mathcal{L}} = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{L}}}$ is the norm of $\boldsymbol{v}$.

**Isometric isomorphism.** The Poincaré ball model $\mathcal{B}$ and the Klein model $\mathcal{K}$ are two other models of hyperbolic spaces. The bijections between a node $\boldsymbol{x} = [x_0, x_1, \cdots, x_n] \in \mathcal{L}$ and its unique corresponding node $\boldsymbol{b} = [b_0, b_1, \cdots, b_{n-1}] \in \mathcal{B}$ are given by

$$p_{\mathcal{L}\to\mathcal{B}}(\boldsymbol{x}) = \frac{[x_1, \cdots, x_n]}{x_0 + 1}, \; p_{\mathcal{B}\to\mathcal{L}}(\boldsymbol{b}) = \frac{[1 + \|\boldsymbol{b}\|^2, 2\boldsymbol{b}]}{1 - \|\boldsymbol{b}\|^2}. \tag{5}$$

The bijections between $\boldsymbol{x} = [x_0, x_1, \cdots, x_n] \in \mathcal{L}$ and its unique corresponding node $\boldsymbol{k} = [k_0, k_1, \cdots, k_{n-1}] \in \mathcal{K}$ are given by

$$p_{\mathcal{L}\to\mathcal{K}}(\boldsymbol{x}) = \frac{[x_1, \cdots, x_n]}{x_0}, \; p_{\mathcal{K}\to\mathcal{L}}(\boldsymbol{k}) = \frac{1}{\sqrt{1 - \|\boldsymbol{k}\|^2}}[1, \boldsymbol{k}]. \tag{6}$$

Geometric relationships among the Lorentz model $\mathcal{L}$, the Poincaré ball model $\mathcal{B}$ and the Klein model $\mathcal{K}$ are presented in Figure 2.

### 3.2. Graph Convolutional Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$, and $\{\boldsymbol{x}_i^E\}_{i\in\mathcal{V}}$ be node features where $^E$ denotes Euclidean representations. At the $\ell$-th layer of GCNs, the graph convolution can be formulated into two steps.

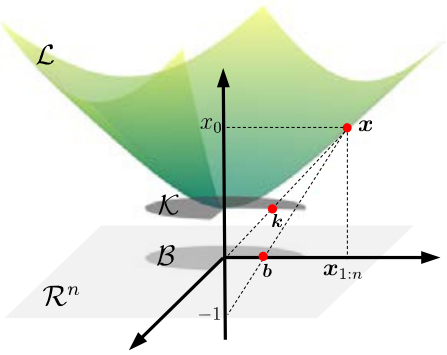



Figure 2. Geometric relationships among $\mathcal{L}$, $\mathcal{B}$ and $\mathcal{K}$.

**Feature Transformation:**

$$\bar{\boldsymbol{h}}_i^{\ell,E} = \boldsymbol{W}^\ell \boldsymbol{h}_i^{\ell-1,E}. \tag{7}$$

where $\boldsymbol{h}_i^{\ell-1,E}$ denotes the $i$-th node's representation at the ($\ell$-1)-th layer and $\boldsymbol{h}_i^{0,E} = \boldsymbol{x}_i^E$. $\boldsymbol{W}^\ell$ is the learnable transformation matrix at the $\ell$-th layer. $\bar{\boldsymbol{h}}_i^{\ell,E}$ is the intermediate representation of the $i$-th node, ready for the next step.

**Neighborhood Aggregation:**

$$\begin{cases} \boldsymbol{m}_i^{\ell,E} = (\bar{\boldsymbol{h}}_i^{\ell,E} + \sum\limits_{j\in\mathcal{N}(i)} w_{ij}\bar{\boldsymbol{h}}_j^{\ell,E}) \\ \boldsymbol{h}_i^{\ell,E} = \sigma(\boldsymbol{m}_i^{\ell,E}) \end{cases}, \tag{8}$$

where $\mathcal{N}(i)$ denotes the set of neighbor nodes of the $i$-th node, and $w_{ij}$ is the aggregation weight. $\boldsymbol{m}_i^{\ell,E}$ is the aggregated message, that is sent to a non-linear activation function $\sigma(\cdot)$ to output the node representation $\boldsymbol{h}_i^{\ell,E}$ at the $\ell$-th layer.

By stacking multiple graph convolutional layers, the feature transformation enables GCNs to learn desirable node embeddings for a target task, *e.g.*, more discriminative for classifications. The neighborhood aggregation enables GCNs to exploit graph topology structures.

## 4. Hyperbolic-to-Hyperbolic GCN

We present H2H-GCN that directly performs graph convolution on hyperbolic manifolds to keep global hyperbolic structure. First, we explain how to generate hyperbolic node representations as input node features are usually Euclidean. Then, we elaborate the developed hyperbolic feature transformation and hyperbolic neighborhood aggregation. Next, we construct the H2H-GCN architecture used for link prediction, node classification and graph classification. Finally, we describe how to optimize parameters in the H2H-GCN.

### 4.1. Hyperbolic Node Representations

Let $\{\boldsymbol{x}_i^E\}_{i\in\mathcal{V}}$ be input Euclidean node features, and $\boldsymbol{o} \coloneqq [1, 0, \cdots, 0]$ denote the origin on the manifold $\mathcal{L}$ of

the Lorentz model. There is $\langle \boldsymbol{o}, [0, \boldsymbol{x}_i^E] \rangle_{\mathcal{L}} = 0$, where $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ denotes the Lorentz inner product defined in Eq. (1). We can reasonably regard $[0, \boldsymbol{x}_i^E]$ as a node on the tangent space at the origin $\boldsymbol{o}$. H2H-GCN uses the exponential map defined in Eq. (3) to generate hyperbolic node representations on the Lorentz model:

$$\begin{aligned} \boldsymbol{x}_i^{\mathcal{L}} &= \exp_{\boldsymbol{o}}([0, \boldsymbol{x}_i^E]) \\ &= \left[\cosh(\|\boldsymbol{x}_i^E\|_2), \sinh(\|\boldsymbol{x}_i^E\|_2) \frac{\boldsymbol{x}_i^E}{\|\boldsymbol{x}_i^E\|_2}\right]. \end{aligned} \tag{9}$$

### 4.2. Hyperbolic Feature Transformation

The feature transformation in (Euclidean) GCNs defined in Eq. (7) is a linear transformation realized via a matrix-vector multiplication. Nevertheless, it will break the hyperbolic manifold constraint while applying matrix-vector multiplication to hyperbolic node representations, making the transformed nodes not lie on hyperbolic manifolds. We developed a Lorentz linear transformation to tackle this problem.

**Definition 1** *(The Lorentz linear transformation). For any $\boldsymbol{x} \in \mathcal{L}$, the Lorentz linear transformation is defined as*

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{W}\boldsymbol{x} \\ \text{s.t. } \boldsymbol{W} &= \begin{bmatrix} 1 & \boldsymbol{0}^\top \\ \boldsymbol{0} & \widehat{\boldsymbol{W}} \end{bmatrix}, \widehat{\boldsymbol{W}}^\top \widehat{\boldsymbol{W}} = \boldsymbol{I}, \end{aligned} \tag{10}$$

*where $\boldsymbol{W}$ is a transformation matrix, and $\widehat{\boldsymbol{W}}$ is called a transformation sub-matrix. $\boldsymbol{0}$ is a column vector of zeros, and $\boldsymbol{I}$ is an identity matrix.*

**Proposition 1** *The Lorentz linear transformation defined in Definition. 1 is manifold-preserving. It ensures that the output $\boldsymbol{y}$ still lies on the manifold $\mathcal{L}$ of the Lorentz model.*

*Proof.* *For any $\boldsymbol{x} = [x_0, x_1, \cdots, x_n] \in \mathcal{L}$, we have*

$$-x_0^2 + \boldsymbol{x}_{1:n}^\top \boldsymbol{x}_{1:n} = -1, \text{ and } x_0 > 0,$$

*where $\boldsymbol{x}_{1:n} = [x_1, x_2, \cdots, x_n]$. After applying the Lorentz linear transformation to $\boldsymbol{x}$, we have*

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} = \left[x_0, \widehat{\boldsymbol{W}}\boldsymbol{x}_{1:n}\right],$$

*satisfying*

$$y_0 = x_0 > 0,$$

*and*

$$\begin{aligned} \langle \boldsymbol{y}, \boldsymbol{y} \rangle_{\mathcal{L}} &= -x_0^2 + (\widehat{\boldsymbol{W}}\boldsymbol{x}_{1:n})^\top \widehat{\boldsymbol{W}}\boldsymbol{x}_{1:n} \\ &= -x_0^2 + \boldsymbol{x}_{1:n}^\top (\widehat{\boldsymbol{W}}^\top \widehat{\boldsymbol{W}}) \boldsymbol{x}_{1:n} \\ &= -x_0^2 + \boldsymbol{x}_{1:n}^\top \boldsymbol{x}_{1:n} \\ &= -1. \end{aligned}$$

*Thus, $\boldsymbol{y}$ lies on the manifold $\mathcal{L}$ of the Lorentz model.* □

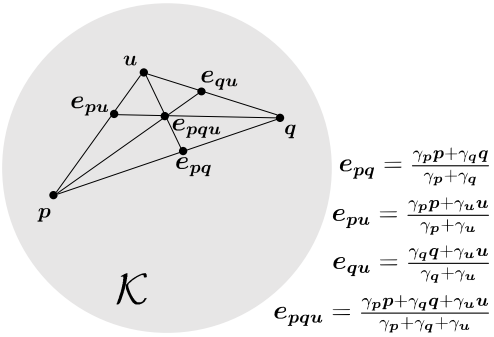



Figure 3. Einstein midpoint on the Klein model $\mathcal{K}$, taking three nodes $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q} \in \mathcal{K}$ for example.

We utilize the Lorentz linear transformation as the hyperbolic feature transformation in H2H-GCN. At the $\ell$-th layer, we take the node representation from the previous layer $\boldsymbol{h}_i^{\ell-1,\mathcal{L}}$ and the transformation matrix $\boldsymbol{W}^\ell$ as input. The $i$-th node's intermediate representation is calculated by

$$\begin{aligned} \bar{\boldsymbol{h}}_i^{\ell,\mathcal{L}} &= \boldsymbol{W}^\ell \boldsymbol{h}_i^{\ell-1,\mathcal{L}} \\ \text{s.t. } \boldsymbol{W}^\ell &= \begin{bmatrix} 1 & \boldsymbol{0}^\top \\ \boldsymbol{0} & \widehat{\boldsymbol{W}}^\ell \end{bmatrix}, \widehat{\boldsymbol{W}}^{\ell\top} \widehat{\boldsymbol{W}} = \boldsymbol{I}, \end{aligned} \tag{11}$$

where $\boldsymbol{h}_i^{0,\mathcal{L}} = \boldsymbol{x}_i^{\mathcal{L}}$. The intermediate node representation $\bar{\boldsymbol{h}}_i^{\ell,\mathcal{L}}$ is ready for hyperbolic neighborhood aggregation in Section 4.3. We describe an effective way to learn transformation matrix $\boldsymbol{W}^\ell$, a constrained parameter, via optimization on a Stiefel manifold in Section 4.5.

### 4.3. Hyperbolic Neighborhood Aggregation

The neighborhood aggregation in GCNs defined in Eq.(8) updates a node representation by aggregating the message from its neighbor node set, enabling GCNs to capture graph topological structure. A generalization of Euclidean mean aggregation in hyperbolic spaces is Fréchet mean [18]. However, Fréchet mean is difficult to apply because it does not have a closed-form solution. We adopt the Einstein midpoint [40] as the hyperbolic neighborhood aggregation in H2H-GCN. In this case, our hyperbolic neighborhood aggregation possesses two desirable properties: translation invariance and rotation invariance. The aggregated hyperbolic average is invariant to translating the input node set by a same distance in a common direction, and invariant to rotating the input node set by a same angle around the origin.

The Einstein midpoint takes the form in the Klein model, illustrated in Figure 3. We first project the intermediate node representations from the Lorentz model to the Klein model, and then calculate the hyperbolic average via the Einstein midpoint. The aggregated hyperbolic average on the Klein model is projected back to the Lorentz model. Formally, given the intermediate representation of a node $\bar{\boldsymbol{h}}_i^{\ell,\mathcal{L}}$ and the intermediate representations of its neighbor

nodes $\{\bar{\boldsymbol{h}}_j^{\ell,\mathcal{L}}\}_{j\in\mathcal{N}(i)}$, the hyperbolic neighborhood aggregation on the Lorentz model is given by

$$\begin{cases} \bar{\boldsymbol{h}}_j^{\ell,\mathcal{K}} = p_{\mathcal{L}\to\mathcal{K}}(\bar{\boldsymbol{h}}_j^{\ell,\mathcal{L}}) \\ \boldsymbol{m}_i^{\ell,\mathcal{K}} = \sum_{j\in\widehat{\mathcal{N}}(i)} \gamma_j \bar{\boldsymbol{h}}_j^{\ell,\mathcal{K}} / \sum_{j\in\widehat{\mathcal{N}}(i)} \gamma_j \\ \boldsymbol{m}_i^{\ell,\mathcal{L}} = p_{\mathcal{K}\to\mathcal{L}}(\boldsymbol{m}_i^{\ell,\mathcal{K}}) \end{cases}, \tag{12}$$

where $\bar{\boldsymbol{h}}_j^{\ell,\mathcal{K}}$ is the $j$-th node's intermediate representation on the Klein model. $p_{\mathcal{L}\to\mathcal{K}}(\cdot)$ and $p_{\mathcal{K}\to\mathcal{L}}(\cdot)$ are the isometric and isomorphic bijections between the Lorentz model and the Klein model as defined in Eq. (6). $\widehat{\mathcal{N}}(i)$ is a node set consisting of the $i$-th node and its neighbor nodes. $\gamma_j = \frac{1}{\sqrt{1-\|\bar{\boldsymbol{h}}_j^{\mathcal{K}}\|^2}}$ denotes the Lorentz factor. $\boldsymbol{m}_i^{\ell,\mathcal{K}}$ is the hyperbolic average on the Klein model that aggregates the message from $\widehat{\mathcal{N}}(i)$ via the Einstein midpoint. We get the hyperbolic average on the Lorentz model $\boldsymbol{m}_i^{\ell,\mathcal{L}}$ by projecting $\boldsymbol{m}_i^{\ell,\mathcal{K}}$ to $\mathcal{L}$.

The non-linear activation plays an important role in GCNs, which prevents a multi-layer network from collapsing into a single layer network. However, applying commonly-used non-linear activation functions (*e.g.*, ReLU) on the Lorentz representation will break the manifold constraint of the Lorentz model. We notice that the non-linear activation on the Poincaré ball model $\mathcal{B}$ is manifold-preserving: for any $\boldsymbol{b} \in \mathcal{B}$, we have $\sigma(\boldsymbol{b}) \in \mathcal{B}$. Inspired by this, we project hyperbolic average $\boldsymbol{m}_i^{\ell,\mathcal{L}}$ to the Poincaré ball model to apply non-linear activation, and then project the result back to the Lorentz model, given by

$$\boldsymbol{h}_i^{\ell,\mathcal{L}} = p_{\mathcal{B}\to\mathcal{L}}\Big(\sigma\big(p_{\mathcal{L}\to\mathcal{B}}(\boldsymbol{m}_i^{\ell,\mathcal{L}})\big)\Big), \tag{13}$$

where $p_{\mathcal{B}\to\mathcal{L}}(\cdot)$ and $p_{\mathcal{L}\to\mathcal{B}}(\cdot)$ are the isometric and isomorphic bijections between the Lorentz model and the Poincaré ball model as defined in Eq. (5). After Eq.(13), H2H-GCN obtains the output of the $\ell$-th layer: the $i$-th node's representation $\boldsymbol{h}_i^{\ell,\mathcal{L}}$ on the Lorentz model.

### 4.4. H2H-GCN Architecture

We summarize the H2H-GCN embedding generation algorithm as shown in Algorithm 1. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$, H2H-GCN first maps input Euclidean node features $\{\boldsymbol{x}_i^E\}_{i\in\mathcal{V}}$ to hyperbolic space via Eq. (9). The obtained hyperbolic node representations $\{\boldsymbol{h}_i^{0,\mathcal{L}}\}_{i\in\mathcal{V}}$ are sent to a multi-layer H2H-GCN. At the $\ell$-th layer, the input node representation $\boldsymbol{h}_i^{\ell-1,\mathcal{L}}$ from previous layer is passed through the hyperbolic feature transformation in Eq. (11), and is updated via the hyperbolic neighborhood aggregation in Eq. (12) and the non-linear activation in Eq.(13). After $L$ layers, we obtain the H2H-GCN node embeddings $\{\boldsymbol{h}_i^{L,\mathcal{L}}\}_{i\in\mathcal{V}}$.

**Algorithm 1:** H2H-GCN embedding generation.

**Input:** Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; node features $\{\boldsymbol{x}_i^E\}_{i\in\mathcal{V}}$; number of layers $L$; transformation matrices $\{\boldsymbol{W}^\ell\}_{\ell=1}^L$; non-linearity activation function $\sigma(\cdot)$.
**Output:** H2H-GCN node embeddings $\{\boldsymbol{h}_i^{L,\mathcal{L}}\}_{i\in\mathcal{V}}$.

Orthogonally initialize transformation sub-matrices $\{\widehat{\boldsymbol{W}}^\ell\}_{\ell=1}^L$;
Construct $\boldsymbol{W}^\ell = \begin{bmatrix} 1 & \boldsymbol{0}^\top \\ \boldsymbol{0} & \widehat{\boldsymbol{W}}^\ell \end{bmatrix}, \forall \ell \in \{1, \cdots, L\}$;
Generate hyperbolic node representations $\{\boldsymbol{x}_i^{\mathcal{L}}\}_{i\in\mathcal{V}}$ via Eq. (9);
$\boldsymbol{h}_i^{0,\mathcal{L}} = \boldsymbol{x}_i^{\mathcal{L}}, \forall i \in \mathcal{V}, ;$
**for** $\ell = 1$ *to* $L$ **do**
  Generate intermediate node representations $\{\bar{\boldsymbol{h}}_i^{\ell,\mathcal{L}}\}_{i\in\mathcal{V}}$ via the hyperbolic feature transformation in Eq. (11);
  **for** $i \in \mathcal{V}$ **do**
    Generate hyperbolic average $\boldsymbol{m}_i^{\ell,\mathcal{L}}$ by aggregating message from $\{\bar{\boldsymbol{h}}_j^{\ell,\mathcal{L}}\}_{j\in\widehat{\mathcal{N}}(i)}$ via the hyperbolic heighborhood aggregation in Eq. (12);
    Generate the node representation $\boldsymbol{h}_i^{\ell,\mathcal{L}}$ via the non-linear activation on $\boldsymbol{m}_i^{\ell,\mathcal{L}}$ in Eq. (13);
  **end**
**end**
**return** H2H-GCN node embeddings $\{\boldsymbol{h}_i^{L,\mathcal{L}}\}_{i\in\mathcal{V}}$.

For link prediction, we use the Fermi-Dirac decoder [26, 30] to calculate probability scores for the edge between the $i$-th node and the $j$-th node, given by

$$p\big((i,j)\in\mathcal{E}|\boldsymbol{h}_i^{L,\mathcal{L}}, \boldsymbol{h}_j^{L,\mathcal{L}}\big) = \big[e^{(d_{\mathcal{L}}(\boldsymbol{h}_i^{L,\mathcal{L}}, \boldsymbol{h}_j^{L,\mathcal{L}})^2 - r)/t} + 1\big]^{-1}, \tag{14}$$

where $d_{\mathcal{L}}(\cdot,\cdot)$ is the hyperbolic distance function defined in Eq. (2), and $r$ and $t$ are hyper-parameters. Following [10], we use the negative sampling strategy and the cross entropy loss for training.

For node classification and graph classification, we exploit a centroid-based classification method studied in [27]. Specifically, we introduce a set of centroids $C = \{\boldsymbol{c}_1^{\mathcal{L}}, \boldsymbol{c}_2^{\mathcal{L}}, \cdots, \boldsymbol{c}_{|C|}^{\mathcal{L}}\}$ lying on the Lorentz model, then calculate a distance matrix $\boldsymbol{D} \in \mathbb{R}^{|\mathcal{V}|\times|C|}$ whose element $\boldsymbol{D}_{i,j} = d_{\mathcal{L}}(\boldsymbol{h}_i^{L,\mathcal{L}}, \boldsymbol{c}_j^{\mathcal{L}})$ represents the distance between the $i$-th node embedding $\boldsymbol{h}_i^{L,\mathcal{L}}$ and the $j$-th centroid $\boldsymbol{c}_j^{\mathcal{L}}$. For node classification, we send $\boldsymbol{D}_i$, the $i$-th row of distance matrix that contains the distance information between the $i$-th node and all centroids, to a classifier to predict the category of the $i$-th node. For graph classification, we apply average pooling to $\{\boldsymbol{D}_i\}_{i=1}^{|\mathcal{V}|}$ as a readout operation to yield a graph embedding $\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\boldsymbol{D}_i$, followed a classifier for prediction. On both classification tasks, we use softmax classifiers and

| | Datasets | DISEASE | | AIRPORT | | PUBMED | | CORA | |
|---|---|---|---|---|---|---|---|---|---|
| | **Methods** | LP | NC | LP | NC | LP | NC | LP | NC |
| Shallow | EUC | $59.8 \pm 2.0$ | $32.5 \pm 1.1$ | $92.0 \pm 0.0$ | $60.9 \pm 3.4$ | $83.3 \pm 0.1$ | $48.2 \pm 0.7$ | $82.5 \pm 0.3$ | $23.8 \pm 0.7$ |
| | HYP [30] | $63.5 \pm 0.6$ | $45.5 \pm 3.3$ | $94.5 \pm 0.0$ | $70.2 \pm 0.1$ | $87.5 \pm 0.1$ | $68.5 \pm 0.3$ | $87.6 \pm 0.2$ | $22.0 \pm 1.5$ |
| | EUC-MIXED | $49.6 \pm 1.1$ | $35.2 \pm 3.4$ | $91.5 \pm 0.1$ | $68.3 \pm 2.3$ | $86.0 \pm 1.3$ | $63.0 \pm 0.3$ | $84.4 \pm 0.2$ | $46.1 \pm 0.4$ |
| | HYP-MIXED | $55.1 \pm 1.3$ | $56.9 \pm 1.5$ | $93.3 \pm 0.0$ | $69.6 \pm 0.1$ | $83.8 \pm 0.3$ | $73.9 \pm 0.2$ | $85.6 \pm 0.5$ | $45.9 \pm 0.3$ |
| NNs | MLP | $72.6 \pm 0.6$ | $28.8 \pm 2.5$ | $89.8 \pm 0.5$ | $68.6 \pm 0.6$ | $84.1 \pm 0.9$ | $72.4 \pm 0.2$ | $83.1 \pm 0.5$ | $51.5 \pm 1.0$ |
| | HNN [19] | $75.1 \pm 0.3$ | $41.0 \pm 1.8$ | $90.8 \pm 0.2$ | $80.5 \pm 0.5$ | $94.9 \pm 0.1$ | $69.8 \pm 0.4$ | $89.0 \pm 0.1$ | $54.6 \pm 0.4$ |
| GCNs | GCN [25] | $64.7 \pm 0.5$ | $69.7 \pm 0.4$ | $89.3 \pm 0.4$ | $81.4 \pm 0.6$ | $91.1 \pm 0.5$ | $78.1 \pm 0.2$ | $90.4 \pm 0.2$ | $81.3 \pm 0.3$ |
| | GAT [41] | $69.8 \pm 0.3$ | $70.4 \pm 0.4$ | $90.5 \pm 0.3$ | $81.5 \pm 0.3$ | $91.2 \pm 0.1$ | $79.0 \pm 0.3$ | $93.7 \pm 0.1$ | $\mathbf{83.0} \pm 0.7$ |
| | GRAPHSAGE [21] | $65.9 \pm 0.3$ | $69.1 \pm 0.6$ | $90.4 \pm 0.5$ | $82.1 \pm 0.5$ | $86.2 \pm 1.0$ | $77.4 \pm 2.2$ | $85.5 \pm 0.6$ | $77.9 \pm 2.4$ |
| | SGC [45] | $65.1 \pm 0.2$ | $69.5 \pm 0.2$ | $89.8 \pm 0.3$ | $80.6 \pm 0.1$ | $94.1 \pm 0.0$ | $78.9 \pm 0.0$ | $91.5 \pm 0.1$ | $81.0 \pm 0.1$ |
| HYP GCNs | HGCN [10] | $90.8 \pm 0.3$ | $74.5 \pm 0.9$ | $\mathbf{96.4} \pm 0.1$ | $\mathbf{90.6} \pm 0.2$ | $96.3 \pm 0.0$ | $\mathbf{80.3} \pm 0.3$ | $92.9 \pm 0.1$ | $79.9 \pm 0.2$ |
| | H2H-GCN (Ours) | $\mathbf{97.0} \pm 0.3$ | $\mathbf{88.6} \pm 1.7$ | $\mathbf{96.4} \pm 0.1$ | $89.3 \pm 0.5$ | $\mathbf{96.9} \pm 0.0$ | $79.9 \pm 0.5$ | $\mathbf{95.0} \pm 0.0$ | $82.8 \pm 0.4$ |

Table 1. ROC AUC for Link Prediction (LP), and F1 score (DISEASE, binary class) and accuracy (the others, multi-class) for Node Classification (NC) tasks. We set the embedding dimensionality to 16 for fair comparisons. The results of EUC, EUC-MIXED, HYP-MIXED, and MLP are reported from [10].

cross entropy loss functions.

### 4.5. Optimization

We explain how to learn parameters in H2H-GCN, especially the transformation matrix $\boldsymbol{W}$ (omitting the layer number $^{\ell}$ for convenience) in the hyperbolic feature transformation Eq. (11), that is an optimization problem with the orthogonal constraint. Other parameters can be learned by a standard gradient descent optimizer straightforwardly.

The transformation matrix $\boldsymbol{W} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \widehat{\boldsymbol{W}} \end{bmatrix}$ is a block diagonal matrix that consists of a scalar 1 and an orthogonal matrix $\widehat{\boldsymbol{W}} \in \mathrm{St}(n', n)$ which resides on the Stiefel manifold.

**Definition 2** *(The Stiefel manifold). The set of $(n' \times n)$-dimensional matrices, $n \leq n'$, with orthonornal columns forms a compact Riemannian manifold called the Stiefel manifold* $\mathrm{St}(\mathrm{n}', \mathrm{n})$ *[5].*

$$\mathrm{St}(\mathrm{n}', \mathrm{n}) \triangleq \{\boldsymbol{M} \in \mathbb{R}^{n' \times n} : \boldsymbol{M}^\top \boldsymbol{M} = \boldsymbol{I}\}. \tag{15}$$

While updating $\boldsymbol{W}$, we keep 1 unchanged and introduce a Riemannian stochastic gradient descent optimizer to update $\widehat{\boldsymbol{W}}$. Formally, let $J$ be the loss function, *e.g.*, cross entropy loss for classifications. $\widehat{\boldsymbol{W}}$ is updated by

$$\begin{cases} \boldsymbol{P}^{(t)} = \eta \pi_{\widehat{\boldsymbol{W}}^{(t)}}(\nabla^{(t)}_{\widehat{\boldsymbol{W}}}) \\ \widehat{\boldsymbol{W}}^{(t+1)} = r_{\widehat{\boldsymbol{W}}^{(t)}}(-\boldsymbol{P}^{(t)}) \end{cases}, \tag{16}$$

where $\eta$ is the learning rate. $\nabla^{(t)}_{\widehat{\boldsymbol{W}}} = \mathrm{d}J/\widehat{\boldsymbol{W}}^{(t)}$ denotes the Euclidean gradient of the loss function $J$ with respect to $\widehat{\boldsymbol{W}}^{(t)}$ calculated at time $t$. $\pi_{\widehat{\boldsymbol{W}}^{(t)}}(\cdot)$ is an orthogonal projection that transforms the Euclidean gradient to the Riemannian gradient

$$\pi_{\widehat{\boldsymbol{W}}^{(t)}}(\nabla^{(t)}_{\widehat{\boldsymbol{W}}}) = \nabla^{(t)}_{\widehat{\boldsymbol{W}}} - \frac{1}{2}\widehat{\boldsymbol{W}}^{(t)}\Big(\widehat{\boldsymbol{W}}^{(t)\top}\nabla^{(t)}_{\widehat{\boldsymbol{W}}} + \nabla^{(t)\top}_{\widehat{\boldsymbol{W}}}\widehat{\boldsymbol{W}}^{(t)}\Big). \tag{17}$$

$r_{\widehat{\boldsymbol{W}}^{(t)}}(\cdot)$ is a retraction operation, defined as

$$r_{\widehat{\boldsymbol{W}}^{(t)}}(-\boldsymbol{P}^{(t)}) = \mathrm{qf}(\widehat{\boldsymbol{W}}^{(t)} - \boldsymbol{P}^{(t)}), \tag{18}$$

where $\mathrm{qf}(\cdot)$ extracts the orthogonal factor in the QR decomposition. The retraction operation prevents the updated $\widehat{\boldsymbol{W}}^{(t+1)}$ from falling off the Stiefel manifold.

## 5. Experiments

We evaluate the proposed H2H-GCN on the link prediction, node classification and graph classification tasks, and comprehensively compare H2H-GCN with a variety of state-of-the-art Euclidean GCNs and hyperbolic GCNs.

### 5.1. Link Prediction and Node Classification

The link prediction (LP) task is to predict the existence of links among nodes in a graph, and the node classification (NC) task is to predict labels of nodes in a graph. They have many applications such as predicting friendships among users in a social network and predicting research directions of papers in a citation network. For link prediction, we report area under the ROC curve (AUC), and for node classification, we report F1 score for binary-class datasets and accuracy for multi-class datasets. Following HGCN [10], a link prediction regularization objective is added in the node classification task.

**Datasets.** DISEASE [10] is constructed by simulating the SIR disease spreading model [1], where the label of a node indicates whether the node was infected or not, and the feature of a node indicates the susceptibility of the node to

| Methods | Dimensionality | | | | |
|---|---|---|---|---|---|
| | 3 | 5 | 10 | 20 | 256 |
| Euclidean | $77.2 \pm 0.12$ | $90.0 \pm 0.21$ | $90.6 \pm 0.17$ | $94.8 \pm 0.25$ | $95.3 \pm 0.17$ |
| HGNN [27] | $94.1 \pm 0.03$ | $95.6 \pm 0.14$ | $96.4 \pm 0.23$ | $96.6 \pm 0.22$ | $95.3 \pm 0.28$ |
| H2H-GCN (Ours) | **95.4** $\pm 0.26$ | **96.7** $\pm 0.12$ | **96.8** $\pm 0.04$ | **97.0** $\pm 0.05$ | **97.2** $\pm 0.03$ |

Table 2. Results on synthetic graph classification where F1 (macro) score and standard deviation are reported. The results of Euclidean and HGNN are reported from [27].

the disease. CORA and PUBMED [35] are citation network datasets where nodes are scientific papers and edges represent citation links. CORA contains 7 classes of machine learning papers, and there are $2,708$ nodes, $5,429$ edges and $1,433$ features per node. PUBMED contains 3 classes of medicine publications, and there are $19,717$ nodes, $44,338$ edges and $500$ features per node. AIRPORT [10] is a flight network dataset where nodes are airports and edges represent the airline routes. There are $2,236$ nodes in total and the label of a node is the population of the country where the airport (node) belongs to.

**Baselines.** We consider four types of baseline methods: shallow methods, neural networks (NNs) methods, GCNs, and hyperbolic GCNs (HYP GCNS). Shallow methods optimize to minimize a reconstruction loss, and the parameters in models act as an embedding look-up table. We consider Euclidean embeddings (EUC) and its hyperbolic extension (HYP) [30]. As the two embeddings fail to leverage node features, EUC-MIXED and HYP-MIXED concatenate the shallow embeddings with node features for a fair comparison with other methods using node features. NNs methods only utilize the node features but does not consider graph structures. Compared NNs methods include Euclidean multi-layer perceptron (MLP) and its hyperbolic extension: hyperbolic neural networks (HNN) [19]. For GCNs, we compare H2H-GCN with several Euclidean state-of-the-art GCNs models: GCN [25], GRAPHSAGE [21], graph attention networks (GAT) [41], and simplified graph convolution (SGC) [45]. For HYP GCNS, we consider HGCN [10] that performs graph convolutional operations in tangent spaces.

**Results.** The comparisons are presented in Table 1. We notice that HNN, a generalization of MLP to hyperbolic spaces, outperforms MLP on most tasks. It indicates that hyperbolic spaces are more suitable for modeling graphs compared than Euclidean spaces. A similar conclusion can be drawn while comparing Euclidean GNNs with hyperbolic GCNs. HGCN works better than Euclidean GCNs in most cases. H2H-GCN performs competitively or even exceeds many graph networks on both tasks. We take the DISEASE dataset as an example to analyze the effectiveness of H2H-GCN. The DISEASE dataset is a tree network that possesses a strong hierarchical structure. As hyperbolic spaces can be viewed as smooth versions of trees, hyperbolic GCNs should work better than Euclidean GCNs. The results are in line with expectation that both HGCN and H2H-GCN show significant improvement on the LP and NC tasks compared with Euclidean methods. In particular, H2H-GCN achieves an average of $21.9\%$ (LP) and $18.2\%$ (NC) performance gains than HNN and GAT, and $6.2\%$ (LP) and $14.1\%$ (NC) performance gains than HGCN. It demonstrates that our H2H-GCN is superior to hyperbolic GCNs which rely on tangent spaces. We owe it to our developed hyperbolic graph convolution that directly works on the hyperbolic manifold. Both the hyperbolic feature transformation and the hyperbolic neighborhood aggregation are manifold-preserving. It can avoid the distortion caused by tangent space approximations and keep the global hyperbolic geometry underlying graphs. For NC on the AIRPORT and PUBMED datasets, our method achieve comparable results with HGCN.

## 5.2. Graph Classification

### 5.2.1 Synthetic Graphs

Following [27], we take three graph generation algorithms: Erdős-Rényi [15], Barabási-Albert [3] and Watts-Strogatz [43] to construct a synthetic graph dataset. For each graph generation algorithm, we generate $6,000$ graphs and divide them into 3 equal parts for the training, validation, and test (see [27] for more generation details). Typical properties, such as small-world property of graphs generated by Watts-Strogatz algorithm and scale-free property of graphs generated by Barabási-Albert algorithm, can be explained by an underlying hyperbolic geometry [26], thus it is more suitable for modeling such graphs in hyperbolic spaces than in Euclidean spaces.

We compare the proposed H2H-GCN with Euclidean embeddings and HGNN [27]. The F1 scores of different embedding dimensionalities are presented in Table 2. The performance of HGNN and our method surpasses Euclidean embeddings by a large margin when embedding dimensionality is low. It is because hyperbolic spaces can well capture the hyperbolic geometry underlying these synthetic graphs. As the embedding dimensionality increases (*e.g.*, 256), HGNN tends to be comparable with Euclidean alternative $95.3\%$, while our method achieves $97.2\%$, $1.9\%$

higher than HGNN. The reason is that the distortion caused by tangent space approximations in HGNN becomes significant with the increase of dimensionality of embedding spaces, leading to an inferior performance. H2H-GCN tackles this problem by directly learning node embeddings in the hyperbolic space. It shows the best performance from 3-dimensional embeddings to 256-dimensional embeddings.

#### 5.2.2 Molecular Graphs

We evaluate our method on several chemical datasets to predict the function of molecular graphs. D&D [14] has $1,178$ graphs in total, and 2 classes indicting the molecular is an enzyme or not. PROTEINS [6] has $1,113$ graphs, and 3 classes of graphs representing helix, sheet or turn. ENZYMES [34] contains 600 graphs and 6 classes in total.

We compare our method with several state-of-the-art Euclidean GCNs, including DGCNN [48], DIFFPOOL [47], ECC [36], GIN [46] and GRAPHSAGE [21], and a hyperbolic GCN, *i.e.*, HGNN [27] that performs Euclidean graph convolutional operations in tangent spaces. In general, researchers adopt tenfold cross validation for model evaluation. However, as pointed out by [16], the data splits are different and the experimental procedures are often ambiguous of different works, which results in unfair comparisons. To solve this problem, the work in [16] provides a uniform and rigorous benchmarking of state-of-the-art models. In this part, we follow the same experimental procedures and use the same data splits as [16] for fair comparisons.

| **Methods** | D&D | PROTEINS | ENZYMES |
|---|---|---|---|
| DGCNN [48] | $76.6 \pm 4.3$ | $72.9 \pm 3.5$ | $38.9 \pm 5.7$ |
| DIFFPOOL [47] | $75.0 \pm 3.5$ | $73.7 \pm 3.5$ | $59.5 \pm 5.6$ |
| ECC [36] | $72.6 \pm 4.1$ | $72.3 \pm 3.4$ | $29.5 \pm 8.2$ |
| GIN [46] | $75.3 \pm 2.9$ | $73.3 \pm 4.0$ | $59.6 \pm 4.5$ |
| GRAPHSAGE [21] | $72.9 \pm 2.0$ | $73.0 \pm 4.5$ | $58.2 \pm 6.0$ |
| HGNN [27] | $75.8 \pm 3.3$ | $73.7 \pm 2.3$ | $51.3 \pm 6.1$ |
| H2H-GCN (Ours) | **78.2** $\pm 3.3$ | **74.4** $\pm 3.0$ | **61.3** $\pm 4.9$ |

Table 3. Results on chemical graph classification where mean accuracy and standard deviation are reported.

The mean accuracy and standard deviation are reported in Table 3. We observe that HGNN are comparable with Euclidean methods, which illustrates two possible reasons: either hyperbolic GCNs are not suitable for the three datasets, or some factors in HGNN limit representation ability of hyperbolic GCNs. The performance of H2H-GCN may give the answer. It achieves the best performance on all datasets: $1.6\%$ higher than DGCNN on D&D, $0.7\%$ higher than DIFFPOOL on PROTEINS, and $1.7\%$ higher than GIN on ENZYMES. Compared with HGNN that does graph convolutional operations in the tangent space, the key difference is that the proposed H2H-GCN performs a hyperbolic graph convolution in the hyperbolic space. In this way, H2H-GCN preserves the global hyperbolic geometry, leading to a superior performance.

| | **Dimensionality** | | | | |
|---|---|---|---|---|---|
| **Methods** | 3 | 5 | 10 | 20 | 128 |
| Euclidean | $64.2 \pm 4.9$ | $71.2 \pm 3.4$ | $76.2 \pm 1.5$ | $78.1 \pm 2.1$ | $80.4 \pm 0.9$ |
| HGNN [27] | $65.3 \pm 3.6$ | $71.0 \pm 3.4$ | $76.1 \pm 1.5$ | $79.2 \pm 1.6$ | $80.1 \pm 0.9$ |
| HGCN [10] | $70.8 \pm 1.6$ | $75.4 \pm 1.7$ | $78.1 \pm 0.8$ | $79.7 \pm 1.4$ | $81.7 \pm 0.7$ |
| H2H-GCN (Ours) | **73.1** $\pm 2.5$ | **77.8** $\pm 0.6$ | **79.9** $\pm 0.9$ | **81.2** $\pm 0.9$ | **83.6** $\pm 0.8$ |

Table 4. Comparisons of embedding dimensionality for node classification on CORA where accuracy and standard deviation are reported. The results of Euclidean and HGNN are based on the official code of HGNN. The results of HGCN are based on its official code.

### 5.3. Dimensionality Comparisons

We test the effect of embedding dimensionality from 3 to 128 for node classification on CORA, and report the performance of Euclidean embedding, HGNN [27], HGCN [10] and the proposed H2H-GCN in Table 4. HGNN gets comparable results with Euclidean embeddings, while H2H-GCN shows pretty improvements. H2H-GCN outperforms HGCN, $2.3\%$ and $1.9\%$ higher than HGCN when embedding dimensionalities are 3 and 256, respectively. We claim that the distortion caused by tangent space approximations exists in both low and high embedding dimensionalities. Although increasing dimensionality can improve performance, it cannot solve this problem. H2H-GCN tackles it by proposing a hyperbolic graph convolution to directly work on the hyperbolic manifold. Such a manifold-to-manifold method achieves remarkable improvements.

## 6. Conclusion

In this paper, we have presented a hyperbolic-to-hyperbolic graph convolutional network (H2H-GCN) for embedding graph with hierarchical structure into hyperbolic spaces. The developed hyperbolic graph convolution which consists of a hyperbolic feature transformation and a hyperbolic neighborhood aggregation, can be directly conducted on hyperbolic manifolds. The both operations can ensure that the output still lies on the hyperbolic manifold. In constrast to existing hyperbolic GCNs relying on tangent spaces, H2H-GCN can avoid the distortion caused by tangent space approximations and keep the global hyperbolic geometry underlying graphs. Extensive experiments on link prediction, node classification and graph classification have showed that H2H-GCN achieves competitive results compared with state-of-the-art Euclidean GCNs and existing hyperbolic GCNs.

**Acknowledgments.** This work was supported by the Natural Science Foundation of China (NSFC) under Grants No. 61773062 and No. 62072041.

## References


[1] Roy M Anderson, B Anderson, and Robert M May. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992. 6

[2] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4463–4473, 2019. 2

[3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. 7

[4] Riccardo Benedetti and Carlo Petronio. *Lectures on hyperbolic geometry*. Springer Science & Business Media, 2012. 2

[5] William M Boothby. An introduction to differentiable manifolds and riemannian geometry. In *Academic press*, 1986. 6

[6] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005. 8

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*, pages 1–14, 2014. 2

[8] James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31:59–115, 1997. 2

[9] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 2

[10] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4868–4879, 2019. 1, 2, 5, 6, 7, 8

[11] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008. 1

[12] Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. Representation tradeoffs for hyperbolic embeddings. *Proceedings of Machine Learning Research*, 80:4460, 2018. 1

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016. 2

[14] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003. 8

[15] Paul Erdős and Alfréd Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959. 7

[16] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations (ICLR)*, 2020. 8

[17] P Thomas Fletcher. Geodesic regression and the theory of least squares on riemannian manifolds. *International Journal of Computer Vision (IJCV)*, 105(2):171–185, 2013. 2

[18] Maurice Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pages 215–310, 1948. 4

[19] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5345–5355, 2018. 6, 7

[20] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017. 1, 2

[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1024–1034, 2017. 1, 2, 6, 7, 8

[22] Ruiqi Hu, Shirui Pan, Guodong Long, Qinghua Lu, Liming Zhu, and Jing Jiang. Going deep: Graph convolutional ladder-shape networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2

[23] Zhiwu Huang, Ruiping Wang, Shiguang Shan, Luc Van Gool, and Xilin Chen. Cross euclidean-to-riemannian metric learning with application to face recognition from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 40(12):2827–2840, 2017. 2

[24] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6418–6428, 2020. 2

[25] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 6, 7

[26] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010. 1, 2, 5, 7

[27] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8230–8241, 2019. 1, 2, 5, 7, 8

[28] Shaoteng Liu, Jingjing Chen, Liangming Pan, Chong-Wah Ngo, Tat-Seng Chua, and Yu-Gang Jiang. Hyperbolic visual embedding learning for zero-shot recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9273–9281, 2020. 2

[29] Teng Long, Pascal Mettes, Heng Tao Shen, and Cees GM Snoek. Searching for actions on the hyperbole. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1150, 2020. 2

[30] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances*

*in Neural Information Processing Systems (NeurIPS)*, pages 6338–6347, 2017. 1, 2, 5, 6, 7

[31] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning (ICML)*, pages 3776–3785, 2018. 1, 2

[32] Fragkiskos Papadopoulos, Maksim Kitsak, M Ángeles Serrano, Marián Boguná, and Dmitri Krioukov. Popularity versus similarity in growing networks. *Nature*, 489(7417):537–540, 2012. 1, 2

[33] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *In International Conference on Learning Representations (ICLR)*, 2020. 2

[34] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004. 8

[35] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. 7

[36] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3693–3702, 2017. 8

[37] Rishi Sonthalia and Anna C Gilbert. Tree! i am no tree! i am a low dimensional hyperbolic embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[38] Zequn Sun, Muhao Chen, Wei Hu, Chengming Wang, Jian Dai, and Wei Zhang. Knowledge association with hyperbolic knowledge graph embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020. 2

[39] Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 30(10):1713–1727, 2008. 2

[40] Abraham A Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005. 4

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, pages 1–12, 2017. 2, 6, 7

[42] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[43] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440, 1998. 7

[44] Melanie Weber, Manzil Zaheer, Ankit Singh Rawat, Aditya Menon, and Sanjiv Kumar. Robust large-margin learning in hyperbolic space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[45] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, 2019. 6, 7

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, pages 1–17, 2019. 1, 2, 8

[47] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4800–4810, 2018. 8

[48] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 1, 8

[49] Yiding Zhang, Xiao Wang, Xunqiang Jiang, Chuan Shi, and Yanfang Ye. Hyperbolic graph attention network. *arXiv preprint arXiv:1912.03046*, 2019. 1, 2