Hyperbolic Graph Attention Network

Yiding Zhang, Xiao Wang, Chuan Shi, *Member, IEEE,* Xunqiang Jiang, and Yanfang Ye

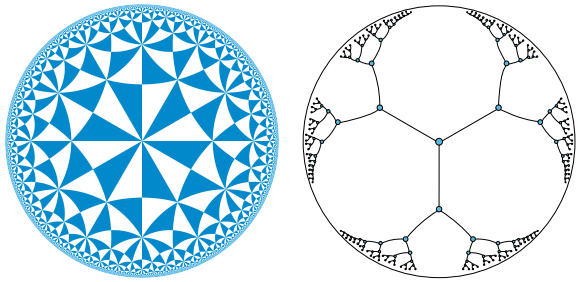
Abstract—Graph neural network (GNN) has shown superior performance in dealing with structured graphs, which has attracted considerable research attention recently. Most of the existing GNNs are designed in Euclidean spaces; however, real-world spatial structured data can be non-Euclidean surfaces (e.g., hyperbolic spaces). For example, biologists may inspect the geometric shape of a protein surface to determine its interaction with other biomolecules for drug discovery. Although there is growing research on generalizing GNNs to non-Euclidean surfaces, the works in these fields are still scarce. In this paper, we exploit the graph attention network to learn robust node representations of graphs in hyperbolic spaces. As the gyrovector space framework provides an elegant algebraic formalism for hyperbolic geometry, we utilize this framework to learn the graph representations in hyperbolic spaces. Specifically, we first use the operations defined in the framework to transform the features in a graph; and we exploit the proximity in the product of hyperbolic spaces to model the multi-head attention mechanism in the non-Euclidean setting; afterward, we further devise a parallel strategy using logarithmic and exponential maps to improve the efficiency of our proposed model. The comprehensive experimental results demonstrate the effectiveness of the proposed model, compared with state-of-the-art methods.

Index Terms—Deep Learning, Hyperbolic Space, Representation Learning, Graph neural network.

1 Introduction

The real-world data usually come together with the graph structure, such as social networks, citation networks, biology networks. Graph neural network (GNN) [1], [2], as a powerful deep representation learning method for such graph data, has shown superior performance on network analysis and aroused considerable research interest. There have been many studies using neural network to handle the graph data. For example, [1], [2] leveraged deep neural network to learn node representations based on node features and the graph structure; [3], [4], [5] proposed the graph convolutional networks by generalizing the convolutional operation to graph; [6] designed a novel convolution-style graph neural network by employing the attention mechanism in GNN. These proposed GNNs have been widely used to solve many real-world application problems. [7], [8], [9] proposed recommender systems based on GNNs. [10], [11] leveraged graph convolution to solve disease prediction problem.

Essentially, most of the existing GNN models are primarily designed for the graphs in Euclidean spaces. The main reason is that Euclidean space is the natural generalization of our intuition-friendly and visible three-dimensional space. However, real-world spatial structured data can be non-Euclidean surfaces (e.g., hyperbolic spaces) [12], [13]. For example, biologists may inspect the geometric shape of a protein surface to determine its interaction with other biomolecules for drug discovery; physicists find some statistical mechanics (e.g., heterogeneous degree distributions) of complex networks can be naturally discovered in hyperbolic geometry [14]; In such cases, existing models that assume spatial structure to be on a Euclidean plane may fail to achieve satisfied performance. On the other hand, several works [14], [15] have demonstrated that the hyperbolic spaces could be the latent spaces of graph data, as the hyperbolic space may reflect some properties of graph naturally, e.g., hierarchical and scale-free structure [14], [16]. Inspired by this insight, the study of graph data in hyperbolic spaces has received increasing attentions, such as hyperbolic graph embedding [15], [17], [18], [19].



(a) A triangular hyperbolic tiling (b) A tree with branching factor 3

Fig. 1. Two examples of hyperbolic spaces (Poincaré disk model).

Compared with Euclidean spaces, one key property of hyperbolic spaces is that they expand faster, because Euclidean spaces expand polynomially while hyperbolic spaces expand exponentially. For instance, each tile in Fig. 1(a) is of equal area in hyperbolic space but diminishes towards zero in Euclidean space towards the boundary. As the tiles grow exponentially, there is sufficient room to embed these tiles. So that we have shrunk the tiles in this Euclidean diagram for visualization. With these properties, hyperbolic spaces can be thought of as "continue tree". As shown in Fig. 1(b), considering a tree with branching factor b, the number of nodes at level l or no more than l hops from the root are $(b+1)b^{l-1}$ and $[(b+1)b^l-2]/(b-1)$ respectively. The number of nodes grows exponentially with their distance to

- *Y. Zhang, X. Wang, C. Shi and X. Jiang are with the Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: {zyd, xiaowang, shichuan, skd621}@bupt.edu.cn.*
- *Y. Ye is with Department of Computer and Data Sciences, Case Western Reserve University, USA. E-mail: yanfang.ye@case.edu.*
- *Chuan Shi is a corresponding author. E-mail: shichuan@bupt.edu.cn.*

the root of the tree, which is similar to hyperbolic spaces as they expand exponentially. Also, the graphs with tree structure can be embedded into 2-dimensional hyperbolic spaces naturally. Given a node at level l, the node can be placed on a sphere in hyperbolic spaces with distance $d \propto l$ to the origin of the sphere, and the branching factor b can be modeled by the constant curvature of hyperbolic spaces as $K = -\ln^2 b$. Therefore, there is a strong correlation between tree-likeness graph and hyperbolic spaces [14], [15]. With this property, hyperbolic spaces have been considered to model complex network recently [14], [16]. These researches discover that graphs with hierarchical structure and power-law distribution are suitable to be modeled in hyperbolic spaces. Meanwhile, graph data with these properties exist widely, such as social networks, network community structures, citation graphs and biology networks [14], [20], which motivates us to study the GNN in hyperbolic spaces.

Despite the powerful modeling ability on graph data of hyperbolic spaces, there are two key challenges in designing the graph attention network in hyperbolic spaces: (1) One is that there are many different procedures in GNNs, e.g., the projection step, the attention mechanism, and the propagation step. However, different from Euclidean spaces, hyperbolic spaces are not vector spaces, so the vector operations (e.g., including vector addition, subtraction and scalar multiplication) cannot be carried in hyperbolic spaces. Although these are some attempts in designing some hyperbolic graph operations, e.g., feature transformation [21], [22], it is unclear how to design the multi-head attention mechanism in the hyperbolic setting, which is a key step in GAT (graph attention network) [6]. Also, as hyperbolic spaces have negative curvatures, choosing a proper curvature is needed to our model. *How can we effectively implement those hyperbolic graph operations of GNN, especially for multi-head attention, in an elegant way?* (2) Another challenge is that mathematical operations in hyperbolic spaces could be more complex than those in Euclidean spaces. Some basic properties of mathematical operations, such as the commutative or associative of "vector addition", are not satisfied anymore in hyperbolic spaces. *How can we assure the learning efficiency in the proposed model?*

To address the above challenges, in this paper, we propose a novel Hyperbolic graph ATtention network (denoted as HAT). Specifically, inspired by [23], we use the framework of gyrovector spaces to build the graph attentional layer in hyperbolic spaces. Gyrovector spaces are the mathematical concepts proposed by Ungar [24], [25], [26], which study hyperbolic geometry in an analogy vector spaces way. In other words, just like vector spaces form algebraic formalism for Euclidean geometry, the framework of gyrovector spaces provides an elegant algebraic formalism for hyperbolic geometry. Therefore, we use the gyrovector operations in hyperbolic spaces to transform the features of the graph and exploit the proximity in hyperbolic spaces to model the multi-head attention mechanism. To improve the learning efficiency, we further propose a logarithmic map and exponential map based method to parallel our model, in the premise of preserving the character in hyperbolic spaces. In sum, the major contributions of this work can be summarized as follows:

- We propose a novel hyperbolic graph attention network, named HAT, which is the first to propose the hyperbolic multi-head attention mechanism. We employ the framework of gyrovector spaces to implement the hyperbolic graph processing.
- We design a method to accelerate our model while preserving the property in the hyperbolic spaces by using the logarithmic map and exponential map, which assures the efficiency of our proposed HAT model.
- We conduct extensive experiments to evaluate the performance of HAT on four datasets. The results show the superiority of HAT in node classification task compared with the state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work. The node distribution in graphs and the concept of gyrovector spaces are introduced in Section 3. Section 4 presents the proposed model HAT as well as the acceleration strategy. Experiments on three real-world datasets are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

In this section, we introduce GNNs, as well as the hyperbolic representation learning methods.

2.1 Graph Neural Network

GNN aims to extend the deep neural network to deal with arbitrary graph-structured data [1], [2]. Recently, there is a surge of generalizing convolutions to the graph domain. This direction can be categorized as spectral approaches and non-spectral approaches. Spectral approaches work with a spectral representation of the graphs. [27] extended convolution to general graph by finding the corresponding Fourier basis. [3] utilized K-order Chebyshev polynomials to approximate smooth filters in the spectral domain. [4] leveraged a localized first-order approximation of spectral graph convolutions to learn the node representations. [28] normalized each node and its neighbors, which served as the receptive field for the convolutional operation. Non-spectral approaches define convolutions directly on the graph and operate on groups of spatially close neighbors. [5] proposed a framework to generate representations by sampling and aggregating features from the local neighborhoods of a node. [28] extracted and normalized each node and its neighborhoods, and the normalized neighborhood served as the receptive field for the convolutional operation. Besides, some researchers proposed other propagation methods of *Message Passing* process of GNN. [29] proposed the gated graph neural network which used the Gate Recurrent Units in the propagation step. [30] proposed the Sentence LSTM for improving text encoding which converted text into a graph and utilized the Graph LSTM to learn the representation. [6] studied the attention mechanism in GNN which incorporated the attention mechanism into the propagation step. Based on [6], [31] proposed a graph attention network with node-level attention as well as semantic-level attention on heterogeneous graphs. [32] uses the relationship between GCN and PageRank to derive an improved propagation scheme based on personalized PageRank. There are some GNNs applied to recommender systems. [7] developed an

efficient GCN algorithm for recommendation which combined efficient random walks and graph convolutions to generate embeddings of nodes. [8] proposed a recommender system which modeled the social influence with a graph attention network and user behaviors with a RNN. A comprehensive review can be found in [33]. To sum up, most GNNs model graph in Euclidean spaces.

2.2 Representation Learning in Hyperbolic Spaces

Recently, representation learning in hyperbolic spaces has received increasing attention, and has been widely applied to many fields, such as graph embedding, natural language processing and recommender systems. For graph embedding, [15] embedded graph into hyperbolic spaces to learn the hierarchical feature representation. [17] focused on discovering pairwise hierarchical relations between concepts via embedding graph in hyperbolic spaces. [18] proposed a novel combinatorial embedding approach as well as a approach to multi-dimensional scaling in hyperbolic spaces. [34] considered the power-law distribution in heterogeneous information network and learnt the embedding in hyperbolic spaces. For natural language processing, Inspired by [15], [35] embedded text in Poincaré model to learn the words and sentences in hyperbolic spaces. [36] focused on question answering problem, modeling the relationship between question and answer representations in hyperbolic spaces. [37] modeled the words in a hyperboloid in Minkowski space to learn the word representations. [38] proposed a novel word embedding method in a Cartesian product of hyperbolic spaces. For recommender systems, [39] designs a recommendation algorithm based on Bayesian personalized ranking in hyperbolic spaces. [40] presents a large scale recommender system in hyperbolic spaces by using Einstein midpoint. Besides, some researchers began to study the deep learning in hyperbolic spaces. [23] generalized deep neural models in hyperbolic spaces, such as recurrent neural networks and gated recurrent unit. [41] proposes attention mechanism in hyperbolic spaces to perform better representation learning. [42] learns graph representations based on adversarial learning via training an autoencoder. Recently, some hyperbolic GCNs are proposed to learn node representations. [21] proposes graph neural networks in hyperbolic spaces which focuses on graph classification problem. [22] leverages hyperbolic graph convolution to learn the node representation in the hyperboloid model. It worth noting that our model is different from these hyperbolic GNNs, because our model focus on node classification task and design our model in Poincaré ball model. A detailed comparison of the hyperbolic GNNs can be found in Section 4.6.

3 Preliminaries

3.1 Hyperbolic Spaces and Graphs

We provide some detail reasons for modeling graphs with hyperbolic geometry. As mentioned in Section 1, one key property of hyperbolic spaces is that they expand faster than Euclidean spaces. Specifically, considering a disk in a 2-dimensional hyperbolic space with constant curvature $\beta = -1$, the perimeter and area of the disk of hyperbolic radius r are given as $2\pi \sinh r$ and $2\pi(\cosh r - 1)$, respectively, and both of them grow as e^r with r. [1] In a 2-dimensional Euclidean space, the length of a circle and the area of a disk of Euclidean radius r are given as $2\pi r$ and πr^2, growing only linearly and quadratically with regard to r. With this property, some researches discover that hyperbolic spaces may be the inherent spaces for graphs with hierarchal structure and power-law distribution [14], [16]. Hence, many real graphs with hierarchical structure and power-law distribution are suitable to be modeled in hyperbolic spaces [43], [44]. Moreover, it is well known that hierarchical structure is a universal phenomenon for real-world graphs [45], including citation networks, social networks, biology networks [14], [20]. Therefore, the conclusion has been made that many real world graphs are suitable to be modeled in hyperbolic spaces.

TABLE 1
Notations and Explanations.

Notation	Explanation
$\mathbb{R}^d$	d-dimension Euclidean space
$\mathbb{D}_c^d$	d-dimension open ball with parameter c
$1/\sqrt{c}$	the radius of open ball $\mathbb{D}^d$
β	the curvature of a hyperbolic space
$T_{\mathbf{x}}\mathbb{D}_c^d$	the tangent space at point $\mathbf{x}$
$\lambda_{\mathbf{x}}^c$	conformal factor
$\mathbf{f}_i$	input feature of node i
$\mathbf{p}_i$	the feature of node i after exponential map
$\mathbf{h}_i$	the feature of node i after linear transformation
$\mathbf{h}_i'$	the final aggregated feature of node i
H_i	the feature of node i in product hyperbolic space
$\mathbf{M}$	weight matrix
$\exp(\cdot)$	exponential map
$\log(\cdot)$	logarithmic map
$d_c(\cdot,\cdot)$	the distance in gyrovector spaces
α	the attention coefficient
w	the normalized attention coefficient
N_i	the neighborhoods of node i

3.2 Differential Geometry

To study graph attention network in hyperbolic spaces, differential geometry should be used. Here, we will briefly introduce some basic related concepts, and more elaborate details of differential geometry can be found in [46], [47].

Manifold. A n-dimensional manifold $\mathcal{M}$ is a topological space that can be locally approximated by a n-dimensional Euclidean space $\mathbb{R}^n$.

Riemannian metric. A Riemannian metric on $\mathcal{M}$ is a collection of inner products: $T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$, for every $x \in \mathcal{M}$.

Riemannian manifold. A smooth manifold equipped with a Riemannian metric is called a Riemannian manifold. The hyperbolic space can be described by some different models, which are defined by different Riemannian manifolds.

Tangent space. The tangent space of $\mathcal{M}$ at a point $x \in \mathcal{M}$ is defined as the n-dimensional vector space approximating $\mathcal{M}$ around x at a first order and denoted by $T_x\mathcal{M}$. Also, tangent vectors are the elements of $T_x\mathcal{M}$. The tangent space is useful when we need to conduct graph operations in hyperbolic spaces.

Exponential map. The exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ is a map from a subset of a tangent space $T_x\mathcal{M}$ of a

1. Because of $\sinh r = \frac{1}{2}(e^r - e^{-r})$, $\cosh r = \frac{1}{2}(e^r + e^{-r})$

Riemannian manifold $\mathcal{M}$ to $\mathcal{M}$ itself. In Euclidean space, the exponential map have the formulation as $\exp_x(s) = x + s$.

3.3 The Poincaré Ball and Gyrovector Spaces

Vector spaces form the algebraic formalism in Euclidean spaces, so we can use vector operations such as vector addition, subtraction and scalar multiplication in Euclidean spaces. These operations are widely used to design algorithms in Euclidean space. However, they cannot be carried in hyperbolic spaces. Fortunately, the framework of gyrovector spaces provides an algebraic formalism for hyperbolic geometry [24], [25], [26]. The gyrovector spaces are defined in the Poincaré ball, which is a isometric model of hyperbolic geometry [48] and enable some operations, such as vector addition and scalar multiplication, to be carried in hyperbolic spaces. We can use gyrovector operations to design the algorithms in hyperbolic spaces. Therefore, we briefly introduce the Poincaré ball model and gyrovector spaces here.

Gyrovector spaces are the tools for the formulation of special relativity, allowing to add speed vectors belonging to the Poincaré ball of radius c (i.e., the speed of light), so that the speed vectors remain in the ball and not exceeding the speed of light [23], [26]. The operations in gyrovector spaces are very useful in design hyperbolic deep learning methods, and the gyrovector spaces are building upon an open d-dimensional Poincaré ball:

$$\mathbb{D}_c^d := \{\mathbf{x} \in \mathbb{R}^d : c\|\mathbf{x}\|^2 < 1\},$$

with the Riemannian metric: $g_{\mathbf{x}} = (\lambda_{\mathbf{x}}^c)^2 g^{\mathbb{R}}$, where $\lambda_{\mathbf{x}}^c = 2/(1 - c\|\mathbf{x}\|^2)$, $g^{\mathbb{R}} = \mathbf{I}_d$, and $c \geq 0$ is corresponding to the radius of the ball. If $c = 0$, then the ball equals to the Euclidean space, i.e., $\mathbb{D}_c^d = \mathbb{R}^d$; if $c > 0$, then $\mathbb{D}_c^d$ is the open ball of radius $\frac{1}{\sqrt{c}}$; if $c = 1$, then we recover the usual ball $\mathbb{D}^d$. Also, the curvature of hyperbolic space is denoted as β and $\beta = -c$. The notations we will use throughout the article are summarized in Table 1.

4 The Proposed Model

In this section, we present our hyperbolic graph attention network model, named HAT, whose framework is shown in Fig. 2. Hence, our model can be summarized as two procedures: (1) **The hyperbolic feature projection**. Given the original input node feature, this procedure projects it into a hyperbolic space through the exponential map and the hyperbolic linear transformation, so as to obtain the latent representation of the node in hyperbolic space. (2) **The hyperbolic attention mechanism**. This procedure designs an attention mechanism based on the hyperbolic proximity to aggregate the latent representations. Also, we devise an acceleration strategy to speed up the proposed model by using logarithmic and exponential mapping, since operations in hyperbolic spaces are usually more complex and time consuming than that in Euclidean spaces. Moreover, we leverage the product of hyperbolic spaces to achieve the multi-head attention. Finally, we feed the aggregated representations to a loss function for the downstream task. Here we mainly describe a single graph attentional layer, as the sole layer is used throughout all of our proposed HAT architectures in our experiments.

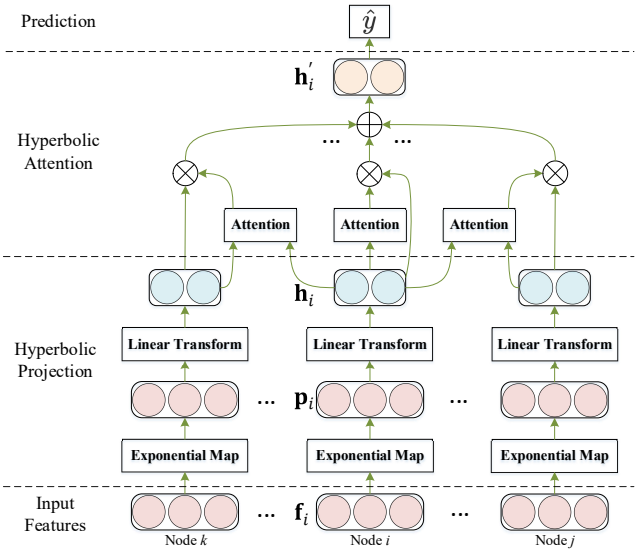


Fig. 2. The framework of HAT model.

4.1 The HAT Model

In this section, we present our HAT model and detail its main procedures.

The hyperbolic feature projection. *The procedure mainly focuses on dealing with an input node and projects it into the hyperbolic spaces as a latent representation.*

The input of GNN is the node feature, which usually lives in Euclidean spaces. To make the node feature available in hyperbolic spaces, we use the exponential map to project the feature into the hyperbolic spaces [21], [22]. Specifically, let $\mathbf{f}_i$ be the feature of node i, and then for $\mathbf{f}_i \in T_{\mathbf{x}}\mathbb{D}_c^d \backslash \{\mathbf{0}\}$ [2] , where $\mathbf{x}$ is a point in hyperbolic spaces and $T_{\mathbf{x}}\mathbb{D}_c^d$ is the tangent space at point $\mathbf{x}$, the exponential map $\exp_{\mathbf{x}}^c : T_{\mathbf{x}}\mathbb{D}_c^d \to \mathbb{D}_c^d$ is given for $\mathbf{x} \neq \mathbf{0}$ by:

$$\exp_x^c(\mathbf{f}_i) = \mathbf{x} \oplus_c \Big(\tanh(\sqrt{c}\frac{\lambda_{\mathbf{x}}^c \|\mathbf{f}_i\|}{2}) \frac{\mathbf{f}_i}{\sqrt{c}\|\mathbf{f}_i\|} \Big), \tag{1}$$

when $\mathbf{x} = \mathbf{0}$, the exponential map is defined as:

$$\exp_{\mathbf{0}}^c(\mathbf{f}_i) = \tanh(\sqrt{c}\|\mathbf{f}_i\|) \frac{\mathbf{f}_i}{\sqrt{c}\|\mathbf{f}_i\|}, \tag{2}$$

where $\lambda_{\mathbf{x}}^c := \frac{2}{(1-c\|\mathbf{x}\|^2)}$ is a conformal factor. The operation $\oplus_c$ is the Möbius addition and it will be interpreted in Eq. (6). Here we assume that the feature $\mathbf{f}_i$ lies in the tangent spaces at the point $\mathbf{x} = \mathbf{0}$, so we can get the new feature $\mathbf{p}_i \in \mathbb{D}_c^d$ in hyperbolic spaces via $\mathbf{p}_i = \exp_0^c(\mathbf{f}_i)$.

We then transform $\mathbf{p}_i$ into a higher-level latent representation $\mathbf{h}_i$ to obtain sufficient representation power. To this end, we use a shared linear transformation parametrized by a weight matrix $\mathbf{M} \in \mathbb{R}^{d' \times d}$ (where d' is the dimension of the final representation). The challenge is that we cannot simply use the Euclidean matrix-vector multiplication, and instead, here we employ the Möbius matrix-vector multiplication [23]. If $\mathbf{M}\mathbf{p}_i \neq \mathbf{0}$, we have:

$$\mathbf{h}_i = \mathbf{M} \otimes_c \mathbf{p}_i = \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\mathbf{M}\mathbf{p}_i\|}{\|\mathbf{p}_i\|} \tanh^{-1}(\sqrt{c}\|\mathbf{p}_i\|) \right) \frac{\mathbf{M}\mathbf{p}_i}{\|\mathbf{M}\mathbf{p}_i\|}, \tag{3}$$

2. $\backslash\{\mathbf{0}\}$ means the point $\mathbf{0}$ is not included.

and if $\mathbf{M}\mathbf{p}_i = \mathbf{0}$, $\mathbf{M} \otimes_c \mathbf{p}_i = \mathbf{0}$. This operation satisfies the matrix associativity: $(\mathbf{M}\mathbf{M}') \otimes_c \mathbf{p}_i = \mathbf{M} \otimes_c (\mathbf{M}' \otimes_c \mathbf{p}_i)$. Here $\mathbf{h}_i \in \mathbb{D}_c^{d'}$ is the representation of node i in the hyperbolic space, which can be considered as a latent representation in the hidden layer of HAT.

The hyperbolic attention mechanism. *The procedure aims to learn the attention coefficient based on the latent representation using hyperbolic similarity and to obtain the aggregated representation in hyperbolic spaces*.

We then perform a self-attention mechanism on the nodes. The attention coefficient α_{ij}, which indicates the importance of node j to node i, is as follows:

$$\alpha_{ij} = f(\mathbf{h}_i, \mathbf{h}_j), \tag{4}$$

where f represents the function of computing the attention coefficient. Here we only compute α_{ij} for nodes $j \in N_i$, where N_i is the neighbors of node i in the graph. Considering a large attention coefficient α_{ij} for the high similarity of nodes j and i, we define f based on the distance in hyperbolic spaces, which is able to measure the similarity between nodes. Specifically, given two node latent representations $\mathbf{h}_i, \mathbf{h}_j \in \mathbb{D}_c^d$, the distance is given by:

$$d_c(\mathbf{h}_i, \mathbf{h}_j) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\| - \mathbf{h}_i \oplus_c \mathbf{h}_j\|), \tag{5}$$

where the operator $\oplus_c$ is the Möbius addition in $\mathbb{D}_c^d$ as:

$$\mathbf{h}_i \oplus_c \mathbf{h}_j := \frac{(1 + 2c\langle \mathbf{h}_i, \mathbf{h}_j \rangle + c\|\mathbf{h}_j\|^2)\mathbf{h}_i + (1 - c\|\mathbf{h}_i\|^2)\mathbf{h}_j}{1 + 2c\langle \mathbf{h}_i, \mathbf{h}_j \rangle + c^2\|\mathbf{h}_i\|^2\|\mathbf{h}_j\|^2}. \tag{6}$$

Notably, when $c = 0$, the Eq. (6) recovers the Euclidean addition of two vectors in $\mathbb{R}^d$. Thus, the Eq. (5) recovers Euclidean geometry in the limit, i.e., $\lim_{c \to 0} d_c(\mathbf{h}_i, \mathbf{h}_j) = 2\|\mathbf{h}_i - \mathbf{h}_j\|$ and the factor 2 comes from the conformal factor. For a general $c > 0$, this operation is neither commutative nor associative, but it satisfies $\mathbf{h}_i \oplus_c \mathbf{0} = \mathbf{0} \oplus_c \mathbf{h}_i$. Moreover, this operation satisfies left-cancellation law: $(-\mathbf{h}_i) \oplus_c \mathbf{h}_i = \mathbf{h}_i \oplus_c (-\mathbf{h}_i) = \mathbf{0}$ and $(-\mathbf{h}_i) \oplus_c (\mathbf{h}_i \oplus_c \mathbf{h}_j) = \mathbf{h}_j$. Then, we perform the self-attention coefficient as:

$$\alpha_{ij} = -d_c(\mathbf{h}_i, \mathbf{h}_j). \tag{7}$$

Because the hyperbolic spaces are metric spaces, there are two advantages of using distance in hyperbolic spaces to obtain the self-attention coefficient. (1) Different from inner product in Euclidean spaces, the hyperbolic distance satisfies the triangle inequality, so the self-attention can preserve the transitivity among nodes. (2) The attention coefficient of a given node i with itself is $\alpha_{ii} = -d_c(\mathbf{h}_i, \mathbf{h}_i) = 0$, which is always the largest over its neighbors. As the representation should mainly maintain its own characteristics, this attention coefficient can meet this requirement in mathematics, while some other graph attention networks, e.g., GAT [6], cannot guarantee this.

For all the neighbors of node i (including itself), we should make their attention coefficients easily comparable, so we normalize them using the softmax function:

$$w_{ij} = \frac{\exp(\alpha_{ij})}{\sum_{l \in N_i} \exp(\alpha_{il})}. \tag{8}$$

The normalized attention coefficient w_{ij} is used to compute a linear combination of the latent representations of all the nodes $j \in N_i$. So the final aggregated representation $\mathbf{h}_i'$ for node i is as follows:

$$\mathbf{h}_i' = \sigma^{\otimes_c}\Big(\sum\nolimits_{j \in N_i}^{\oplus_c} w_{ij} \otimes_c \mathbf{h}_j \Big), \tag{9}$$

where the $\sum^{\oplus_c}$ is the accumulation of Möbius addition. $\sigma^{\otimes_c}$ is hyperbolic nonlinearity, and the detail of it will be shown in Section 4.4. The operation $w_{ij} \otimes_c \mathbf{h}_j$ can be realized by the Möbius scalar multiplication. The Möbius scalar multiplication of $\mathbf{h}_j \in \mathbb{D}_c^d \backslash \{\mathbf{0}\}$ by $w_{ij} \in \mathbb{R}$ is defined as:

$$w_{ij} \otimes_c \mathbf{h}_j := \frac{1}{\sqrt{c}} \tanh\left(w_{ij} \tanh^{-1}(\sqrt{c}\|\mathbf{h}_j\|)\right) \frac{\mathbf{h}_j}{\|\mathbf{h}_j\|}, \tag{10}$$

and $w_{ij} \otimes_c \mathbf{0} := \mathbf{0}$. Please notice that we have $\lim_{c \to 0} w_{ij} \otimes_c \mathbf{h}_j = w_{ij}\mathbf{h}_j$, meaning that the operation recovers the Euclidean scalar multiplication when c goes to zero. This operation satisfies the scalar distributivity ($(r_1 + r_2) \otimes_c \mathbf{h} = r_1 \otimes_c \mathbf{h} + r_2 \otimes_c \mathbf{h}$) and scalar associativity ($(r_1 r_2) \otimes_c \mathbf{h} = r_1 \otimes_c (r_2 \otimes_c \mathbf{h})$) for scalar r_1, r_2 and vector $\mathbf{h}$.

4.2 Multi-head Attention

Multi-head attention can make model get better results [6]. We aim to extend our self attention mechanism to multi-head attention. However, it is unclear how to design the multi-head attention in hyperbolic spaces. The main challenge of designing hyperbolic multi-head attention is to realize the *concatenation operation* in an elegant way.

In Euclidean spaces, the multi-head attention leverages several self-attentions and concatenates their outputs:

$$\mathbf{v} = \Big\Vert_{k=1}^{K} \mathbf{v}^{(k)}, \tag{11}$$

where $\|$ is concatenation operation, $\mathbf{v}^{(k)}$ indicates the output of k-th self-attention, K is the total number of self-attentions. Assuming that the output of each self-attention lives in a m-dimensional Euclidean space $\mathbb{R}^m$, so the output of the multi-head attention lives in the concatenation of these m-dimensional Euclidean spaces [38]. Actually, the concatenated Euclidean space is the Cartesian product of the Euclidean spaces in mathematics. Moreover, the product of K m-dimensional Euclidean spaces $\mathbb{R}^m \times \mathbb{R}^m \cdots$ is equal to a $K \times m$-dimensional Euclidean space $\mathbb{R}^{K \times m}$ [38]. As a result, the graph operations in Euclidean multi-head attention is same with those in self attention.

Different from Euclidean spaces, the product of hyperbolic spaces is unequal to a higher dimensional hyperbolic space [38], which means Euclidean multi-head attention cannot be applied in hyperbolic spaces directly, so some operations in hyperbolic self-attentions should be replaced with the multi-head attention version. To bridge this gap, we should design the multi-head attention in *the product of hyperbolic spaces*. Specifically, for HAT with K-head attention, some operations, including input feature, hyperbolic projection (as shown in Fig. 2), are carried independently for each attention without any change. For hyperbolic attention part, as the node feature lives in the product of K m-dimensional hyperbolic spaces $\mathbb{D}^m \times \mathbb{D}^m \cdots$, we denote the

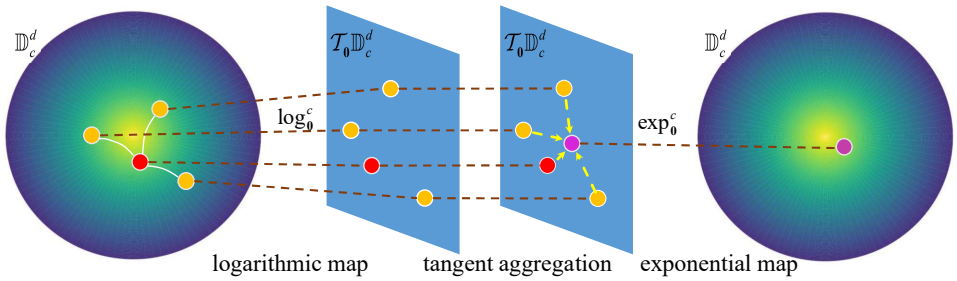


Fig. 3. The acceleration of HAT via tangent aggregation. The node representations are mapped between hyperbolic and Euclidean spaces by using logarithmic and exponential map, so the representations of neighbors can be aggregated in the tangent space.

latent representation of node i as H_i, which is composed of K m-dimensional features $\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}, \cdots, \mathbf{h}_i^{(K)}$:

$$H_i = \Big\|_{k=1}^{K} \mathbf{h}_i^{(k)}. \tag{12}$$

The attention coefficient of hyperbolic multi-head attention α'_{ij} can be computed as:

$$\alpha'_{ij} = -d_p(H_i, H_j), \tag{13}$$

where $d_p(\cdot,\cdot)$ is the distance in the product of hyperbolic spaces. Given node latent representations H_i, H_j, this distance is defined as:

$$d_p(H_i, H_j) = \sqrt{\sum_{k=1}^{K} d_c^2(\mathbf{h}_i^{(k)}, \mathbf{h}_j^{(k)})}, \tag{14}$$

which is also a metric distance. The proof is shown in the appendix. Also, we normalize the attention coefficients for all the neighbors of node i via softmax function:

$$w'_{ij} = \frac{\exp(\alpha'_{ij})}{\sum_{l\in N_i}\exp(\alpha'_{il})}. \tag{15}$$

Next, we aggregate the node representation via the acceleration strategy for each attention independently:

$$\mathbf{h}_i'^{(k)} = \sigma^{\otimes_c}\big(\sum\nolimits_{j\in N_i}^{\oplus_c} w'_{ij} \otimes_c \mathbf{h}_j'^{(k)}\big). \tag{16}$$

Thus, the final output of HAT with multi-head attention is given as:

$$H'_i = \Big\|_{k=1}^{K} \mathbf{h}'^{(k)}_i. \tag{17}$$

4.3 Acceleration of HAT

In our proposed model HAT, the calculation of Eq. (9) and Eq. (16) is very time-consuming, which seriously affects the efficiency of HAT. As mentioned before, the Möbius addition in Eq. (9) and Eq. (16) is neither commutative nor associative, meaning that we have to calculate the results by order. Specifically, taking Eq. (9) as an example, we denote $w_{ij} \otimes_c \mathbf{h}_j$ as $\mathbf{v}_{ij}$, so the accumulation term in Eq. (9) can be rewritten as follows:

$$\begin{aligned}\sum\nolimits_{j\in N_i}^{\oplus_c} \mathbf{v}_{ij} &= \mathbf{v}_{i1} \oplus_c \mathbf{v}_{i2} \oplus_c \mathbf{v}_{i3} \oplus_c \cdots \\ &= \Big(\big((\mathbf{v}_{i1} \oplus_c \mathbf{v}_{i2}) \oplus_c \mathbf{v}_{i3}\big) \oplus_c \cdots\Big).\end{aligned} \tag{18}$$

As we can see, the calculation of Eq. (18) has to be in a serial manner. It is well known that there are always some hubs which have many edges in a large graph, so the calculation becomes very impractical.

The Möbius version of operation [23] provide a feasible way to solution this problem. Actually, some operations in gyrovector spaces can be derived with logarithmic map and exponential map [23]. Taking the Möbius scalar multiplication as an example, it first uses the logarithmic map to project the representation into a tangent space, and then multiply the projected representation by a scalar in the tangent space, and finally project it back on the manifold with the exponential map [23]. The logarithmic and exponential map can project the node representations between the two manifolds in a correct manner. Specifically, for two points $\mathbf{v}_i \in \mathbb{D}_c^d$ and $\mathbf{v}_j \in \mathbb{D}_c^d \backslash \{\mathbf{0}\}$, the logarithmic map $\log_{v_i}^c : \mathbb{D}_c^n \to T_{v_i}\mathbb{D}_c^n$ is given for $\mathbf{v}_j \neq \mathbf{v}_i$ by:

$$\log_{v_i}^c(\mathbf{v}_j) = \frac{2}{\sqrt{c}\lambda_{v_i}^c}\tanh^{-1}(\sqrt{c}\| -\mathbf{v}_i \oplus_c \mathbf{v}_j\|)\frac{-\mathbf{v}_i\oplus_c\mathbf{v}_j}{\| -\mathbf{v}_i\oplus_c\mathbf{v}_j\|}, \tag{19}$$

when $\mathbf{v}_i = \mathbf{0}$, we have:

$$\log_{\mathbf{0}}^c(\mathbf{v}_j) = \tanh^{-1}(\sqrt{c}\|\mathbf{v}_j\|)\frac{\mathbf{v}_j}{\sqrt{c}\|\mathbf{v}_j\|}. \tag{20}$$

The logarithmic map enables us to get the representation $\log_{\mathbf{0}}^c(\mathbf{v}_j)$ in a tangent space. As the tangent spaces are vector spaces, we can combine the representations, just as we do it in the Euclidean spaces, $\sum_{j\in N_i}\log_{\mathbf{0}}^c\left(w_{ij}\otimes_c\mathbf{h}_j\right)$. After the linear combination, we use the exponential map to project the representations back to the hyperbolic spaces, giving rise to the final representation as [21], [22]:

$$\mathbf{h}'_i = \sigma^{\otimes_c}\left(\exp_{\mathbf{0}}^c\left(\sum\nolimits_{j\in N_i}\log_{\mathbf{0}}^c\left(w_{ij}\otimes_c\mathbf{h}_j\right)\right)\right). \tag{21}$$

The diagram of this operation is shown in Fig. 3. Different from Eq. (9) and Eq. (16), the accumulation operation in the Eq. (21) is commutative and associative, so it can be computed in a parallel way. Thus, our model becomes more efficient.

In practice, we implemented HAT via Eq. (9) and Eq. (21) in TensorFlow [49], respectively. Taking the Cora graph (2708 nodes and 5429 edges) [50] as an example, conducted on a GPU (NVIDIA GTX 1080 Ti), HAT only costs about 266 seconds to converge with Eq. (21), while cannot converge within 2 hours with Eq. (9).

4.4 Trainable Curvature and Nonlinearity Activation

As we known, hyperbolic spaces are spaces with negative curvature, i.e., $\beta = -c$. Some researchers found that some

graphs have hyperbolic spaces with different curvature underneath [14], [22]. For a input graph, HAT can also learn the curvature of hyperbolic spaces. Moreover, each layer in HAT has different curvature, so how to smoothly vary curvature at each layer is very important.

Here we leverage the hyperbolic nonlinearity activation to smoothly vary curvature at each layer. As mentioned in Eq. (9), the hyperbolic nonlinearity activation is given as [22]:

$$\sigma^{\otimes_c}(\cdot) = \exp_{\mathbf{0}}^{c'}\Big(\sigma\big(\log_{\mathbf{0}}^{c}(\cdot)\big)\Big), \tag{22}$$

where $\beta = -c$ and $\beta' = -c'$ are curvatures of the current layer and next layer respectively. For two hyperbolic spaces with different curvatures, as they share a same tangent space at $\mathbf{0}$, i.e., $T_{\mathbf{0}}\mathbb{D}_c^d$, we can smooth the curvatures of two layers in the process of nonlinearity activation.

The overall process of HAT is shown in Algorithm 1. We can apply the final representations to specific tasks and optimize them with different loss functions. In this paper, we consider the semi-supervised node classification task, and use cross-entropy loss function to train our model.

Algorithm 1 The overall process of HAT.

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node features $\mathbf{f}$;
Output: The predicted labels of nodes $\hat{y}$

1: Map the input node features into the hyperbolic space via $\mathbf{p}_i = \exp_0^c(\mathbf{f}_i)$ (Eq. (2));
2: Transform $\mathbf{p}_i$ into a higher-level latent representation $\mathbf{h}_i$ (Eq. (3));
3: Calculate the distance $d_c(\mathbf{h}_i, \mathbf{h}_j)$ between node i and its neighborhood node j (Eq. (5) or Eq. (14));
4: Calculate the self-attention coefficient α_{ij} of node i and node j (Eq. (7));
5: Normalize the self-attention coefficient using the softmax function (Eq. (8));
6: Calculate the final aggregated representation $\mathbf{h}_i'$ for node i (Eq. (21));
7: Calculate the cross-entropy;
8: Back propagation and update parameters;
9: **return** Predicted labels of nodes;

4.5 Complexity Analysis

The time complexity of HAT is $O(|V| \cdot d \cdot d' + |E| \cdot d')$, where d and d' are the dimension of input and output features, respectively. $|V|$ and $|E|$ are the numbers of nodes and edges in the graph, respectively. The complexity is on par with other GNN methods, such as GAT [6], GCN [4]. More importantly, our model can be parallelized. For example, the operations of the self-attention can be parallelized across all edges, and the computation of the aggregated representation can also be parallelized across all nodes.

4.6 Discussion on Related Works

Here we compare HAT with some existing hyperbolic GNNs, i.e., HGNN [21] and HGCN [22]. We analyze these methods w.r.t. some technologies, including manifold, the object to be represented, trainable curvature, multi-head attention, and we denote them as "Manifold", "Rep-obj", "Train-curv", "Multi-att", respectively. Also, we denote the Poincaré ball manifold and hyperboloid manifold as $\mathbb{D}$ and $\mathbb{H}$, respectively.

TABLE 2
Hyperbolic graph neural networks discussion.

Method	Manifold	Rep-obj	Train-curv	Multi-att
HGNN	$\mathbb{D}\&\mathbb{H}$	Graph	✗	✗
HGCN	$\mathbb{H}$	Node	✓	✗
HAT	$\mathbb{D}$	Node	✓	✓

TABLE 3
Summary of the datasets.

Dataset	Cora	Citeseer	Pubmed	Amazon Photo
# Nodes	2708	3327	19717	7650
# Edges	5429	4732	44338	143663
# Features	1433	3703	500	745
# Classes	7	6	3	8

The statistics of these methods are shown in Table 2. We can find that these methods leverages different manifold to design the graph convolutional layers. Specifically, HAT leverages the Poincaré ball manifold, while HGCN uses the hyperboloid manifold. HGNN is designed in both Poincaré ball and hyperboloid manifolds. Also, HGNN focuses on learning the representations of graphs in the hyperbolic spaces, while HGCN and HAT focus on learning the representations of nodes. Moreover, the curvatures of spaces in HGCN and HAT is trainable, while HGNN can only be trained in the hyperbolic space with a fixed curvature. Furthermore, HAT first leverages the multi-head attention in the hyperbolic spaces, which can better model the relationship between the center node with its neighbors.

5 Experiments and Analysis

5.1 Experiments Setup

5.1.1 Datasets

We employ three widely used real-world graphs for evaluations, including Cora, Citeseer, Pubmed [50] and Amazon Photo [51]. In Core, Citeseer, and Pubmed, nodes represent documents and edges represent citation relations. Each document has a label and a bag-of-words representation. Amazon Photo is a segment of the Amazon co-purchase graph. In this dataset, nodes represent goods while edges indicate that two goods are frequently bought together. Node features are bag-of-words encoded product reviews, and class labels are given by the product category. Their detailed descriptions are summarized in Table 3. For each dataset, we use only 20 nodes per class for training, 500 nodes for validation, 1000 nodes for test, and the training algorithm could access all nodes' features. These settings are same with former literature [52][3], [4][4], [6][5].

In order to quantify which dataset are suitable to be modeled in hyperbolic spaces. We also compute δ-hyperbolicity [53] to quantify the tree-likeliness of these datasets. However, the complexity of δ-hyperbolicity is $O(|V|^4)$, which is very time consuming. Fortunately, δ_{avg}

3. https://github.com/kimiyoung/planetoid
4. https://github.com/tkipf/gcn
5. https://github.com/PetarV-/GAT

[54] provides a more efficient way to compute the tree-likeness of a graph, because δ_{avg} can be approximated via sampling. A low δ_{avg}-hyperbolicity of a graph indicates that it has an underlying hyperbolic geometry. Moreover, if hyperbolic spaces would constitute a good choice to embed a graph, then most likely the product of hyperbolic spaces would as well [38]. Here we compute δ_{avg}-hyperbolicity of the datasets and the results are shown in Table 4. The detailed results would be analyzed in Section 5.2.

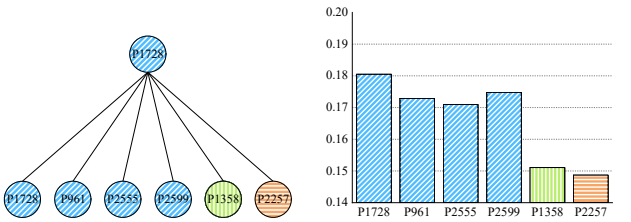
5.1.2 Baselines

We compare our method with the following state-of-the-art methods, including the network embedding methods and GNN based methods.

- **DeepWalk** [55] learns the node representation in Euclidean spaces by modeling the random walks via a skip-gram objective function.
- **Node2vec** [56] can be considered as the generalized DeepWalk. It models the biased random walks via a skip-gram objective function.
- **PoincaréEmb** [15] is a graph embedding method in hyperbolic spaces, which preserves proximities of node pairs via embedding a graph into a Poincaré ball.
- **GCN** [4] is a semi-supervised graph convolutional network designed in Euclidean spaces to learn the node representations.
- **GAT** [6] is a semi-supervised GNN in Euclidean spaces, which incorporated the attention mechanism into the propagation step.
- **GraphSAINT** [57] is a graph sampling based inductive learning method. It trains the graphs in a mini-batch manner.
- **HGNN** [21] is a semi-supervised GNN in hyperbolic spaces. Here we leverages the same loss with HAT for HGNN to conduct the node-level experiments.
- **HGCN** [22] is also a hyperbolic GNN. It leverages the hyperboloid model of hyperbolic spaces to learn the node representations.

5.1.3 Parameter Settings

For all the methods, we carry the experiments in the embedding dimension of 8, 16, 32, 64 (i.e., the number of hidden units in GNN). For DeepWalk and Node2vec, we set window size as 5, walk length as 80, walks per node as 40. Also, for Node2vec, we choose p, q in $\{0.5, 1, 1.5\}$. For PoincaréEmb, we set the number of negative samples as 10. For GAT and HAT, we set a single attention head as 8-dimension and the number of attention mechanism is associated to the embedding dimension. For instance, if the embedding dimension is 64, we set the number of multi-head attention as 8. For GraphSAINT, since it is a sampling based supervised learning method, we should modified its sampler to ensure it can be trained in our semi-supervised learning setting. Specifically, instead of picking nodes from the whole graph, we randomly pick some nodes which have labels and expand their neighborhoods as the samplings to ensure GraphSAINT can be trained in a proper way. For all the GNNs (i.e., GCN, GAT, HGCN and HAT), we choose the learning rate in $\{0.005, 0.008, 0.01\}$, l2 normalization in $\{0.0001, 0.0005, 0.001\}$, dropout in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. Moreover, all the GNNs leverage early stop. We tune the parameters for all methods via validation data.



(a) Neighbors of P1728 (b) Attention values of P1728's neighbors

Fig. 4. Neighbors of node P1728 and corresponding attention values. Different colors mean different classes.

5.2 Node Classification

Node classification is a basic task widely used to evaluate the embedding effectiveness. For GCN, GAT, and HAT, they are the semi-supervised models which can be directly used to classify the nodes. For DeepWalk and Node2vec, we employ KNN classifier with $k = 5$ to perform the node classification. Because the KNN classifier cannot be directly applied to hyperbolic spaces, for PoincaréEmb, we project the representations in the tangent space at $\mathbf{0}$ via $\log_{\mathbf{0}}^{c}$, and then feed the representations into classifier. This operation is inspired by [23], which projects the hyperbolic parameters into the tangent space at $\mathbf{0}$ via logarithmic map, and then feeds these parameters into Euclidean algorithms. We report the average Macro-F1 and Micro-F1 of 10 runs with random weight initialization.

The results are shown in Table 4. It is obvious that HAT achieves the best performance in most cases, and its superiority is more significant for the low dimension setting. Specifically, as we can infer from Table 4, compared with the second best results, HAT achieves better results on these 4 datasets with 8 dimensions. These results validate the claim in Section 1: some graphs can be modeled into low dimensional hyperbolic spaces naturally. Also, as mentioned in Section 5.1.1, we compute δ_{avg} for the 4 datasets. A lower δ_{avg}-hyperbolicity of a graph indicates that this graph is more suitable to be embedded in the hyperbolic spaces. The performance of the hyperbolic GNNs (i.e., HGNN, HGCN and HAT) roughly meet this point. Compared with the Euclidean GNNs (i.e., GCN, GAT and GraphSAINT), the hyperbolic GNNs perform better in Amazon Photo, that because Amazon Photo has a low δ_{avg}-hyperbolicity. One the other hand, because Citeseer has a higher δ_{avg}-hyperbolicity, HAT performs not well in Citeseer with 64-dimension. Nonetheless, HAT also has competitive results. Moreover, we can find that the GNN based methods usually preform better than other baselines (i.e., DeepWalk, Node2vec and PoincaréEmb), because of combining the graph structure and node features in their models. Compared to GCN and HGCN, GAT and HAT achieve better results in most cases, by reason of the multi-head attention mechanism in GAT and HAT. We notice GraphSAINT performs not well on Cora and Citeseer datasets, that is because GraphSAINT is designed for large-scale graphs while Cora

TABLE 4
Quantitative Results on the Node Classification Task.

Dataset	metric	dimension	Deepwalk	Node2vec	PoincaréEmb	GCN	GAT	GraphSAINT	HGNN	HGCN	HAT
Cora $\delta_{avg} = 0.353$	macro	8	64.7±1.1	65.3±1.2	55.0±0.4	79.6±0.7	80.2±0.9	71.5±1.0	79.7±0.9	78.2±0.8	**81.0±0.8**
		16	65.5±1.5	65.7±1.1	65.4±0.4	81.0±0.5	79.6±0.8	75.1±0.8	80.1±0.8	80.1±0.5	**81.9±0.7**
		32	65.9±1.4	65.9±1.4	65.6±0.3	81.4±0.4	80.1±0.7	76.4±0.6	80.4±0.6	80.5±0.6	**82.1±0.5**
		64	67.0±1.0	67.3±0.9	67.6±0.4	81.7±0.4	81.8±0.6	77.9±0.5	81.0±0.5	80.3±0.6	**82.0±0.5**
	micro	8	64.5±1.2	65.7±1.1	57.5±0.6	80.3±0.8	80.4±0.8	72.4±0.9	80.4±1.2	80.0±0.7	**82.6±0.7**
		16	65.2±1.6	66.0±1.3	64.4±0.3	81.9±0.6	81.7±0.7	76.4±0.7	81.6±0.8	81.3±0.6	**83.3±0.6**
		32	65.9±1.5	65.9±1.5	64.9±0.4	81.5±0.4	82.6±0.7	77.2±0.5	81.3±0.6	81.7±0.7	**83.6±0.5**
		64	66.5±1.7	66.9±1.2	68.6±0.4	81.6±0.4	83.1±0.6	78.5±0.4	81.9±0.7	81.4±0.6	**83.4±0.5**
Citeseer $\delta_{avg} = 0.461$	macro	8	46.6±1.3	46.6±1.3	37.8±0.4	66.1±0.9	66.1±0.7	53.7±1.1	66.2±1.2	66.6±0.8	**67.5±0.7**
		16	45.4±1.4	46.2±1.1	39.4±0.5	67.3±0.6	67.3±0.7	55.3±0.8	67.2±0.9	67.6±0.6	**68.9±0.3**
		32	42.3±1.5	45.2±1.2	41.9±0.3	68.5±0.5	67.8±0.6	59.6±0.7	68.4±0.8	68.6±0.5	**69.1±0.5**
		64	43.9±1.5	45.9±1.2	41.9±0.2	68.1±0.5	68.1±0.6	61.3±0.6	68.3±0.5	68.4±0.5	**68.9±0.5**
	micro	8	47.8±1.6	47.8±1.6	38.6±0.4	68.9±0.7	69.5±0.8	56.9±0.9	70.6±0.9	70.9±0.6	**71.3±0.7**
		16	46.2±1.5	46.5±1.3	40.4±0.5	69.8±0.5	70.4±0.7	58.2±0.8	71.0±0.8	71.2±0.5	**72.2±0.6**
		32	43.6±1.9	45.8±1.3	43.5±0.5	70.4±0.5	71.9±0.7	62.2±0.6	71.8±0.5	71.9±0.4	**72.2±0.4**
		64	46.6±1.4	46.6±1.2	43.6±0.4	70.8±0.4	**72.4±0.7**	63.5±0.7	71.5±0.5	71.7±0.5	72.1±0.4
Pubmed $\delta_{avg} = 0.355$	macro	8	71.1±0.7	71.1±0.7	63.6±0.7	76.9±0.5	76.2±0.7	75.0±1.0	76.8±0.4	77.2±0.6	**77.7±0.8**
		16	72.2±0.5	72.9±0.7	67.5±0.3	78.6±0.5	77.7±0.6	75.7±0.6	77.5±1.0	78.3±0.4	**78.9±0.4**
		32	70.8±0.8	71.5±0.8	66.3±0.5	78.1±0.4	77.9±0.6	76.1±0.5	78.2±0.4	78.5±0.5	**79.1±0.5**
		64	71.6±0.8	71.6±0.8	67.7±0.6	78.6±0.5	78.5±0.5	76.7±0.3	78.0±0.5	78.6±0.5	**79.2±0.5**
	micro	8	73.2±0.7	73.2±0.7	66.0±0.8	78.6±0.4	71.9±0.7	75.7±0.8	75.6±0.4	77.9±0.6	**78.9±0.8**
		16	73.9±0.8	74.3±0.8	68.0±0.4	79.1±0.5	75.9±0.7	76.3±0.5	78.3±0.6	78.4±0.4	**79.3±0.5**
		32	72.4±1.0	73.3±0.7	68.4±0.5	78.7±0.5	78.2±0.6	76.4±0.4	78.7±0.4	78.6±0.6	**79.6±0.5**
		64	73.5±1.0	73.5±1.0	69.9±0.6	79.1±0.5	78.7±0.4	76.8±0.4	78.7±0.5	79.3±0.5	**79.5±0.5**
Amazon Photo $\delta_{avg} = 0.268$	macro	8	77.1±0.7	77.9±0.9	75.3±1.1	82.8±1.1	80.6±0.8	80.4±1.1	83.3±1.2	85.4±0.7	**87.6±0.3**
		16	78.2±0.5	78.2±0.5	76.3±0.9	84.8±0.9	82.6±0.8	81.8±0.7	84.5±0.7	85.0±0.5	**88.5±0.3**
		32	80.8±0.8	81.6±0.9	75.9±0.9	85.4±0.8	83.5±0.6	84.9±0.7	85.0±1.0	86.7±0.6	**88.8±0.3**
		64	78.1±0.8	78.7±0.6	76.5±0.8	85.9±0.7	83.3±0.7	85.3±0.5	86.6±0.8	87.8±0.5	**88.6±0.3**
	micro	8	76.2±0.7	78.5±0.8	77.9±0.9	84.1±0.9	81.9±0.9	83.1±1.0	84.2±1.3	86.7±0.8	**88.7±0.5**
		16	78.9±0.8	78.9±0.8	78.6±0.9	86.0±0.7	83.4±0.8	84.9±0.6	85.9±0.8	86.3±0.5	**89.3±0.5**
		32	81.7±1.0	82.1±0.9	76.2±0.8	86.7±0.7	84.3±0.7	84.5±0.7	86.5±0.5	87.9±0.4	**89.7±0.5**
		64	77.5±1.0	79.2±1.1	78.6±0.6	86.9±0.6	84.5±0.7	85.9±0.6	87.9±0.8	88.9±0.4	**89.4±0.4**

TABLE 5
Results on the Node Clustering Task.

Method	Cora	Citeseer	Pubmed	Amazon Photo
DeepWalk	0.404	0.179	0.231	0.681
Node2vec	0.415	0.209	0.286	0.710
PoincaréEmb	0.441	0.264	0.284	0.626
GCN	0.517	0.424	0.305	0.254
GAT	0.567	0.427	0.359	0.667
GraphSAINT	0.501	0.387	0.325	0.673
HGNN	0.553	0.413	0.387	0.721
HGCN	0.572	0.421	0.372	0.692
HAT	**0.585**	**0.439**	**0.399**	**0.735**

and Citeseer are too small to show its ability. Also, how to design a more powerful sampler for GraphSAINT in semi-supervised learning is an interesting problem, and we leave it as a future work. Furthermore, compared to Euclidean GNNs, HAT preforms better in most cases, especially in low dimension, suggesting the superiority of modeling graph in hyperbolic spaces.

5.3 Node Clustering

Here we conduct the node clustering task to evaluate the representations learned from different methods. For the GNN based methods, we obtain the feature representations of test nodes from the hidden layer. The dimension of feature representations is set as 16 for a fair comparison. Here we utilize K-means to perform node clustering, and the number of clusters is set to the number of labels. For hyperbolic representation learning methods, we project their feature representations to the tangent spaces via $\log_{\mathbf{0}}^{c}(\cdot)$, and then feed these feature representations into K-means. We report the average results of normalized mutual information (NMI) of 10 runs with random weight initialization.

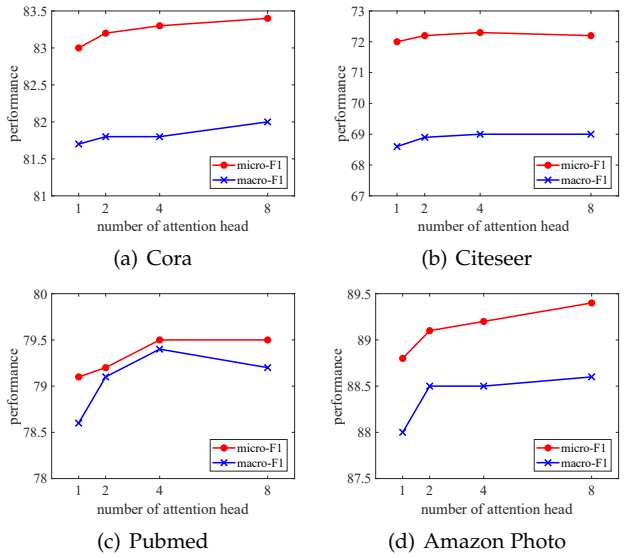


Fig. 5. The performance of HAT when changing the number of attention heads. The multi-head attention recovers self-attention when the number is 1.

The results are displayed in Table 5. As we can see, HAT performs consistently better than all the baselines, indicating the superior performance of HAT. Moreover, for Amazon Photo, some graph embedding methods achieve better results than Euclidean GNNs, while hyperbolic GNNs also achieve good results, demonstrating the superiority of designing graph neural network in hyperbolic spaces.

5.4 Analysis of Attention Mechanism

In this section, we closely examine the learned attention value in HAT w.r.t the effectiveness of hyperbolic attention

mechanism as well as the superiority of hyperbolic multi-head attention.

We first validate the effectiveness of hyperbolic attention value. Intuitively, because the output representation should mainly maintain its own characteristics, the attention value with itself should be the largest. The important neighbors tend to have larger values, while the non-relevant neighbors should have smaller attention values. Specifically, we take the paper "P1728" in Cora dataset as an illustrative example. As shown in the Fig. 4(a), the paper P1728 has 5 neighbors. Among these neighbors, P961, P2555 and P2599 have the same label with P1728, while P1358 and P2257 have different labels. We compute the attention value for 64-dimensional feature (i.e., 8-head attention, and 8 dimension for each attention). From Fig. 4(b), we can see that the paper P1728 gets the highest attention value, which means the node itself plays the most important role in its representation. It is reasonable because all information supported by neighbors are usually viewed as a kind of supplementary information. P2599, P961 and P2555 get the second, third, forth highest attention values, respectively. This is because that the three papers belong to the same class with P1728, and they can make significant contribution to identify the class of P1728. The rest neighbors, P1358 and P2257, get the smallest attention values. It is also reasonable, since, belonging to different classes from P1728, they hardly provide useful information to identify the class of P1728. Based on the above analysis, we can see that our proposed attention mechanism can automatically distinguish the difference among neighbors and assign the higher weights to the meaningful neighbors.

We also validate the impact of the proposed hyperbolic multi-head attention. For fair comparison, we set the dimension of node representation as 64 for all the cases and explore the performance of HAT with various number of attention head. The results are shown in Fig. 5. It worth noting that the multi-head attention recovers single self-attention when the number of attention head is 1. Comparing single self-attention (i.e., the number of attention head is 1), multi-head attention generally have better performance. Moreover, we can find that the more number of attention head can generally improve the performance of HAT.

5.5 Graph Visualization

Graph visualization, aiming to layout a graph on a two-dimensional space, is another important graph application. If we use the color to indicate the label of a node, a good visualization result is that the points with same color are close with each other. Here, we take Pubmed as a case to visualize the learned representations. Followed [15], [17], [39], we directly visualize the learned two-dimensional representations of the nodes.

From Fig. 6, we can find that the representations learned by DeepWalk, Node2vec, GraphSAINT and HGCN are mixed with each other. PoincaréEmb performs relatively better than the above methods, which demonstrates the superiority of modeling graph in hyperbolic spaces. Moreover, we can find that some GNN methods (i.e., HAT, HGNN, GCN and GAT) relatively clearly distinguish three categories of nodes. Compared to other GNNs, HAT distinguishes all three categories with more clear boundary and larger discrimination, which verifies the effectiveness of our proposed model.

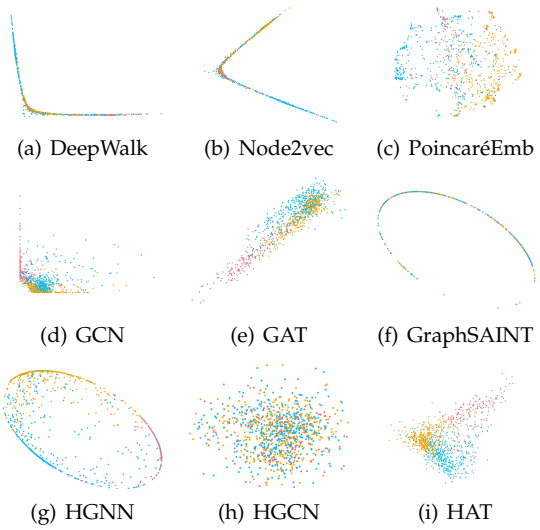


Fig. 6. Visualization of 2-dimension representations on Pubmed. Each point indicates one paper and its color indicates the label.

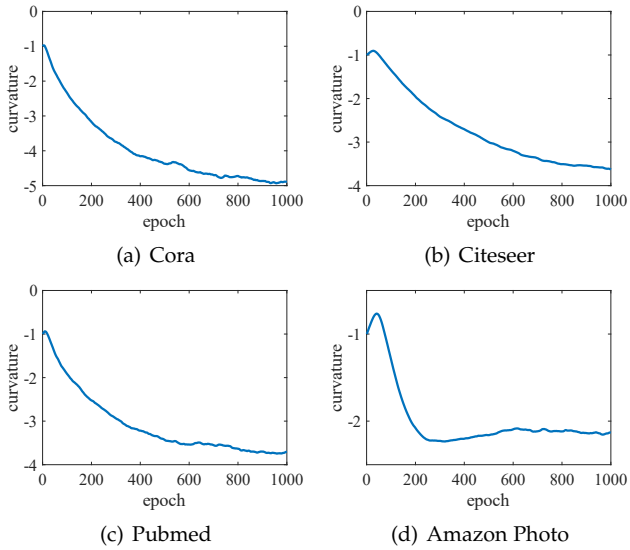


Fig. 7. The changing of curvature β in training process. HAT can learn a underlying hyperbolic geometry with a specific curvature for each dataset.

5.6 Analysis of Trainable Curvature

We also analyze the trainable curvature β of HAT. As shown in Fig. 7, we analyze the changing process of curvature in the training process for four datasets. The curvature $\beta = -c$ in training processes is increasing in the beginning and then decreasing. Finally, the curvature would be converged to a fix value. Also, the curvatures for different datasets are converged to different values. It indicates HAT can learn a underlying hyperbolic geometry with different curvature for each dataset.

6 Conclusion

In this paper, we make the effort toward investigating the graph neural network in hyperbolic spaces and propose a

novel hyperbolic graph attention network HAT. With the framework of gyrovector spaces, we redesign the graph operations in graph attention network in hyperbolic spaces, and propose a multi-head attention mechanism based on the hyperbolic proximity. We further accelerate our model through the logarithmic map and exponential map to improve the efficiency of HAT. The extensive experiments on four tasks demonstrate the superiority of HAT, compared with the state-of-the-arts.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. U20B2045, 61772082, 62002029, U1936104, 61972442), and the National Key Research and Development Program of China (2018YFB1402600). It is also supported by BUPT Excellent Ph.D. Students Foundation (No. CX2019126).

REFERENCES

[1] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *IJCNN*, 2005, pp. 729–734.
[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NeurIPS*, 2016, pp. 3844–3852.
[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.
[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[7] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*, 2018, pp. 974–983.
[8] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *WSDM*, 2019, pp. 555–563.
[9] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *WWW*, 2019, pp. 417–426.
[10] S. Parisot, S. I. Ktena, E. Ferrante, M. Lee, R. G. Moreno, B. Glocker, and D. Rueckert, "Spectral graph convolutions for population-based disease prediction," in *MICCAI*, 2017, pp. 177–185.
[11] S. Rhee, S. Seo, and S. Kim, "Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification," *arXiv preprint arXiv:1711.05859*, 2017.
[12] R. C. Wilson, E. R. Hancock, E. Pekalska, and R. P. Duin, "Spherical and hyperbolic embeddings of data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2255–2269, 2014.
[13] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.
[14] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, p. 036106, 2010.
[15] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *NeurIPS*, 2017, pp. 6338–6347.
[16] A. Muscoloni, J. M. Thomas, S. Ciucci, G. Bianconi, and C. V. Cannistraci, "Machine learning meets complex networks via coalescent embedding in the hyperbolic space," *Nature communications*, vol. 8, no. 1, p. 1615, 2017.
[17] M. Nickel and D. Kiela, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," in *ICML*, vol. 80, 2018.
[18] F. Sala, C. De Sa, A. Gu, and C. Re, "Representation tradeoffs for hyperbolic embeddings," in *ICML*, vol. 80, 2018, pp. 4460–4469.
[19] O. Ganea, G. Becigneul, and T. Hofmann, "Hyperbolic entailment cones for learning hierarchical embeddings," in *ICML*, 2018, pp. 1646–1655.
[20] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.
[21] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," in *NeurIPS*, 2019, pp. 8228–8239.
[22] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *NeurIPS*, 2019, pp. 4869–4880.
[23] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *NeurIPS*, 2018, pp. 5350–5360.
[24] A. A. Ungar, "Hyperbolic trigonometry and its application in the poincaré ball model of hyperbolic geometry," *Computers & Mathematics with Applications*, vol. 41, no. 1-2, pp. 135–147, 2001.
[25] A. A. Ungar, "A gyrovector space approach to hyperbolic geometry," *Synthesis Lectures on Mathematics and Statistics*, vol. 1, no. 1, pp. 1–194, 2008.
[26] A. A. Ungar, *Analytic hyperbolic geometry and Albert Einstein's special theory of relativity*. World scientific, 2008.
[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *ICLR*, 2014.
[28] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016, pp. 2014–2023.
[29] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *ICLR*, 2016.
[30] Y. Zhang, Q. Liu, and L. Song, "Sentence-state lstm for text representation," in *ACL*, 2018, pp. 317–327.
[31] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019, pp. 2022–2032.
[32] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *ICLR*, 2019.
[33] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *arXiv preprint arXiv:1812.04202*, 2018.
[34] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *AAAI*, vol. 33, 2019, pp. 5337–5344.
[35] B. Dhingra, C. J. Shallue, M. Norouzi, A. M. Dai, and G. E. Dahl, "Embedding text in hyperbolic spaces," *arXiv preprint arXiv:1806.04313*, 2018.
[36] Y. Tay, L. A. Tuan, and S. C. Hui, "Hyperbolic representation learning for fast and efficient neural question answering," in *WSDM*, 2018, pp. 583–591.
[37] M. Leimeister and B. J. Wilson, "Skip-gram word embeddings in hyperbolic space," *arXiv preprint arXiv:1809.01498*, 2018.
[38] A. Tifrea, G. Bécigneul, and O.-E. Ganea, "Poincaré glove: Hyperbolic word embeddings," in *ICLR*, 2018.
[39] T. D. Q. Vinh, Y. Tay, S. Zhang, G. Cong, and X.-L. Li, "Hyperbolic recommender systems," *arXiv preprint arXiv:1809.01703*, 2018.
[40] B. P. Chamberlain, S. R. Hardwick, D. R. Wardrope, F. Dzogang, F. Daolio, and S. Vargas, "Scalable hyperbolic recommender systems," *arXiv preprint arXiv:1902.08648*, 2019.
[41] C. Gulcehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K. M. Hermann, P. Battaglia, V. Bapst, D. Raposo, A. Santoro *et al.*, "Hyperbolic attention networks," in *ICLR*, 2019.
[42] D. Grattarola, D. Zambon, L. Livi, and C. Alippi, "Change detection in graph streams by learning graph embeddings on constant-curvature manifolds," *IEEE Transactions on neural networks and learning systems*, vol. 31, no. 6, pp. 1856–1869, 2019.
[43] F. Papadopoulos, M. Kitsak, M. Á. Serrano, M. Boguná, and D. Krioukov, "Popularity versus similarity in growing networks," *Nature*, vol. 489, no. 7417, p. 537, 2012.
[44] A. Faqeeh, S. Osat, and F. Radicchi, "Characterizing the analogy between hyperbolic embedding and community structure of complex networks," *Physical review letters*, vol. 121, no. 9, p. 098301, 2018.
[45] A. Clauset, C. Moore, and M. E. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, p. 98, 2008.
[46] S. Helgason, *Differential geometry, Lie groups, and symmetric spaces*. Academic press, 1979, vol. 80.
[47] J. M. Lee, B. Chow, S.-C. Chu, D. Glickenstein, C. Guenther, J. Isenberg, T. Ivey, D. Knopf, P. Lu, F. Luo *et al.*, "Manifolds and differential geometry," *Topology*, vol. 643, p. 658, 2009.
[48] J. W. Cannon, W. J. Floyd, R. Kenyon, W. R. Parry *et al.*, "Hyperbolic geometry," *Flavors of geometry*, vol. 31, pp. 59–115, 1997.
[49] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016, pp. 265–283.

[50] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.
[51] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.
[52] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICLR*, 2016.
[53] M. Gromov, "Hyperbolic groups," in *Essays in group theory*. Springer, 1987, pp. 75–263.
[54] R. Albert, B. DasGupta, and N. Mobasheri, "Topological implications of negative curvature for biological and social networks," *Physical Review E*, vol. 89, no. 3, p. 032811, 2014.
[55] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.
[56] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.
[57] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *ICLR*, 2020.

Yiding Zhang received the B.S. degree from Beijing University of Posts and Telecommunications in 2017. He is currently a Ph.D. student in Beijing University of Posts and Communications. His research interests include data mining and machine learning.

Xiao Wang received the PhD degree from the Tianjin University. He is currently an assistant professor with the Beijing University of Posts and Telecommunications. Prior to that, he was a postdoctoral researcher in the Department of Computer Science and Technology, Tsinghua University. His current research interests include data mining, social network analysis, and machine learning. Until now, he has published more than 30 papers in conferences such as AAAI, IJCAI, etc., and journals such as the IEEE Transactions on Cybernetics, the IEEE Transactions on Knowledge Discovery and Engineering, etc.

Chuan Shi received the B.S. degree from the Jilin University in 2001, the M.S. degree from the Wuhan University in 2004, and Ph.D. degree from the ICT of Chinese Academic of Sciences in 2007. He joined the Beijing University of Posts and Telecommunications as a lecturer in 2007, and is a professor and deputy director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia at present. His research interests are in data mining, machine learning, and evolutionary computing. He has published more than 40 papers in refereed journals and conferences.

Xunqiang Jiang received the B.S. degree from Beijing University of Posts and Telecommunications in 2019. He is currently a master student in Beijing University of Posts and Communications. His research interests include data mining and recommender system.

Yanfang Ye is currently the Theodore L. and Dana J. Schroeder Associate Professor in the department of Computer and Data Sciences at Case Western Reserve University. She received her Ph.D. in computer science from Xiamen University. She was previously an assistant professor and then associate professor in the department of Computer Science and Electrical Engineering at West Virginia University. Her research areas mainly include Data Mining, Machine Learning, and Health Intelligence.