Multi-Hyperbolic Space-based Heterogeneous Graph Attention Network

Jongmin Park, Seunghoon Han, Jong-Ryul Lee*, Sungsu Lim*
Department of Computer Science and Engineering, Chungnam National University, Daejeon, South Korea
{pa5398, tmdgns129}@g.cnu.ac.kr, {jongryul.lee, sungsu}@cnu.ac.kr

***Abstract*—To leverage the complex structures within heterogeneous graphs, recent studies on heterogeneous graph embedding use a hyperbolic space, characterized by a constant negative curvature and exponentially increasing space, which aligns with the structural properties of heterogeneous graphs. However, despite heterogeneous graphs inherently possessing diverse power-law structures, most hyperbolic heterogeneous graph embedding models use a single hyperbolic space for the entire heterogeneous graph, which may not effectively capture the diverse power-law structures within the heterogeneous graph. To address this limitation, we propose *Multi-hyperbolic Space-based heterogeneous Graph Attention Network (MSGAT)*, which uses multiple hyperbolic spaces to effectively capture diverse power-law structures within heterogeneous graphs. We conduct comprehensive experiments to evaluate the effectiveness of MSGAT. The experimental results demonstrate that MSGAT outperforms state-of-the-art baselines in various graph machine learning tasks, effectively capturing the complex structures of heterogeneous graphs.**

***Index Terms*—heterogeneous graph representation learning, graph neural networks, hyperbolic graph embedding**

I. Introduction

Recently, the demand for effective methods to learn semantic information and complex structures in heterogeneous graphs, which consist of various node and link types, has been steadily increasing due to their ability to represent real-world scenarios. In heterogeneous graphs, metapaths are defined as sequences of node/link types. Leveraging metapaths enables us to capture semantic information and complex structures within such graphs effectively. Accordingly, recent studies [1]–[3] have focused on efficiently learning heterogeneous graph representations by leveraging metapaths.

Despite their notable achievements, they may struggle to effectively capture complex structures (e.g., power-law structure) within heterogeneous graphs because they use Euclidean space as the embedding space. In heterogeneous graphs, we often observe hierarchical or power-law structures where the number of nodes grows exponentially, corresponding to specific metapaths. Using Euclidean space as the embedding space to learn such complex structures may result in distortions and limitations [4]. For example, as shown in Figure 1(a), given a graph with a hierarchical structure, distortions can occur where the geodesic distance between two nodes (v_2 and v_3) is far, but the vector distance in the embedding space is represented as close.

*Corresponding authors.

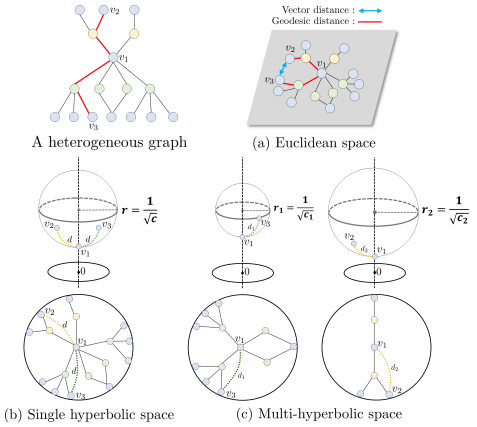


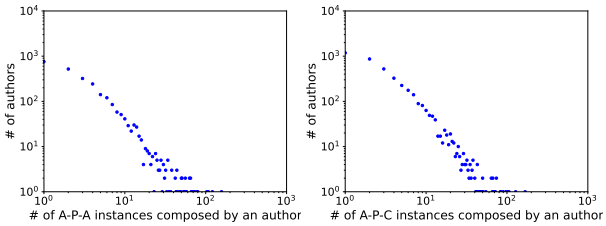
Fig. 1: Examples of heterogeneous graph representations in various embedding spaces.

Some recent heterogeneous graph embedding models address this challenge by using the hyperbolic space as the embedding space. Compared to Euclidean space, hyperbolic space has a constant negative curvature and grows exponentially. Some recent studies [5]–[10] argue that these inherent properties of hyperbolic space offer a solution to represent complex structures effectively. While these studies achieved significant performance, would representing a heterogeneous graph with different complex structures based on semantic information in a single hyperbolic space be effective? In hyperbolic space, the extent to which hyperbolic space grows exponentially is determined by the negative curvature. Conversely, we can interpret negative curvature as indicating the degree of power-law distribution in hyperbolic space. Therefore, using multiple hyperbolic spaces with distinct negative curvatures that effectively represent each power-law structure within a heterogeneous graph would be more effective for heterogeneous graph representation learning.

As illustrated in Figure 1(b), if power-law structures with different degree distributions are learned in the same hyperbolic space, it will not be effective in capturing their structural properties. This is because, according to two different metapaths representing different degree distributions, they are represented at the same hyperbolic distance d from the central embedding target node v_1 to its metapath-based neighbors. In contrast, as shown in Figure 1(c), if structures with more pronounced power-law distributions are learned in hyperbolic space with steeper curvature, and those with less pronounced power-law structures are learned in hyperbolic

2024 IEEE International Conference on Data Mining (ICDM) | 979-8-3315-0668-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDM59182.2024.00098

2374-8486/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDM59182.2024.00098



(a) A-P-A in DBLP (δ_{avg} : 0.7892) (b) A-P-C in DBLP (δ_{avg} : 0.3124)

Fig. 2: Metapath instance distributions of some metapaths in DBLP. δ represents Gromov's δ-hyperbolicity of each metapath-based subgraph.

space with relatively softer curvature, the structural properties of each metapath can be effectively captured. This is because, according to two metapaths representing different degree distributions, they are represented at different distances d_1 and d_2 from the target node embedding to its metapath-based neighbors.

Additionally, as shown in Figure 2, in real-world heterogeneous graphs, we can observe multiple power-law structures corresponding to specific metapaths that are similar but distinct. Also, we calculate the average Gromov's δ-hyperbolicity [11] of each metapath-based subgraph, where lower δ values indicate that the structure of the subgraph tends to exhibit more hierarchical structures. The average Gromov δ-hyperbolicity allows for identifying distinct hierarchical structures in power-law structures with similar distributions, despite their apparent similarity.

Based on these observations, we propose Multi-hyperbolic Space-based heterogeneous Graph Attention Network (MSGAT) to effectively learn various semantic structural properties in heterogeneous graphs with power-law structures. Specifically, MSGAT addresses metapath predefinition challenges by sampling metapath instances and utilizing intra-hyperbolic space attention to learn representations in metapath-specific hyperbolic spaces, where curvatures are learnable parameters. Also, we utilize inter-hyperbolic space attention to aggregate semantic information across distinct metapaths. Through these approaches, our model can effectively learn power-law structures and semantic information within the heterogeneous graph.

The main contributions of our work can be summarized as follows:

- We propose a novel hyperbolic heterogeneous graph attention network that uses multiple hyperbolic spaces as embedding spaces for distinct metapaths.
- We design graph attention mechanisms in multiple hyperbolic spaces to enhance heterogeneous graph representations, capturing diverse degree distributions and semantic heterogeneity.
- The experimental results demonstrate that MSGAT outperforms state-of-the-art baselines in various downstream tasks with heterogeneous graphs.

II. Preliminaries

A. Hyperbolic Space

Definition 1 (Poincaré ball model). Poincaré ball model with curvature $-c$ $(c > 0)$ is defined by the Riemannian manifold $(\mathbb{D}^{n,c}, g_x^c)$, where

$$\mathbb{D}^{n,c} = \{x \in \mathbb{R}^n : c||x||^2 < 1\},$$
$$g_x^c = (\lambda_x^c)\, I_d.$$

Here, $\mathbb{D}^{n,c}$ is the open n-dimensional unit ball with radius $\frac{1}{\sqrt{c}}$ and g_x^c is the Riemannian metric tensor where $\lambda_x^c = \frac{2}{1-c||x||^2}$ and I_d is the identity matrix. We denote $\mathcal{T}_x\mathbb{D}^{n,c}$ as the tangent space centered at point x

Definition 2 (Möbius addition). Given the point $x, y \in \mathbb{D}^{n,c}$, Möbius addition which represents the equation for the addition operation in the Poincaré ball model with curvature $-c$ $(c > 0)$ is defined as follows:

$$x \oplus_c y = \frac{\left(1 + 2c\langle x, y\rangle + c||y||^2\right) x + \left(1 - c||x||^2\right) y}{1 + 2c\langle x, y\rangle + c^2||x||^2||y||^2},$$

where $\langle\cdot\rangle$ is the Euclidean inner product and $||\cdot||$ is the Euclidean norm.

Definition 3 (Exponential and logarithmic maps). In Poincaré ball model with curvature $-c$ $(c > 0)$, the exponential map $exp_x^c : \mathcal{T}_x\mathbb{D}^{n,c} \to \mathbb{D}^{n,c}$ and logarithmic map $log_x^c : \mathbb{D}^{n,c} \to \mathcal{T}_x\mathbb{D}^{n,c}$ are defined as shown below:

$$exp_x^c(v) = x \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_x^c||v||}{2}\right)\frac{v}{\sqrt{c}||v||}\right),$$
$$log_x^c(y) = \frac{2}{\sqrt{c}\lambda_x^c}\tanh^{-1}\left(\sqrt{c}||-x \oplus_c v||\right)\frac{-x \oplus_c y}{||-x \oplus_c y||},$$

where x and y are the points in the hyperbolic space $\mathbb{D}^{n,c}$ and $x \neq y$. v is a nonzero tangent vector in the tangent space $\mathcal{T}_x\mathbb{D}^{n,c}$.

Definition 4 (Hyperbolic matrix-vector multiplication). Given a point $x \in \mathbb{D}^{n,c}$ and a matrix $M \in \mathbb{R}^{m\times n}$, the matrix multiplication operation in hyperbolic space is defined as follows:

$$M \otimes_c x = exp_0^c\left(M log_0^c(x)\right),$$

where $\mathbf{0} \in \mathbb{R}^n$ is a zero vector.

Definition 5 (Hyperbolic non-linear activation function). Given the point $x \in \mathbb{D}^{n,c}$, the hyperbolic non-linear activation function is defined as follows:

$$\sigma \otimes_c (x) = exp_0^c\left(\sigma\left(log_0^c(x)\right)\right),$$

where σ is the Euclidean non-linear activation function.

III. Methodology

A. Metapath Instance Sampling

To capture the structural properties within a heterogeneous graph $\mathcal{G}$, we sample a metapath instance set $\mathcal{P}_v$ for a given embedding target node v in $\mathcal{V}_t$. Each metapath instance p in $\mathcal{P}_v$ starts from node v and has a length within a maximum

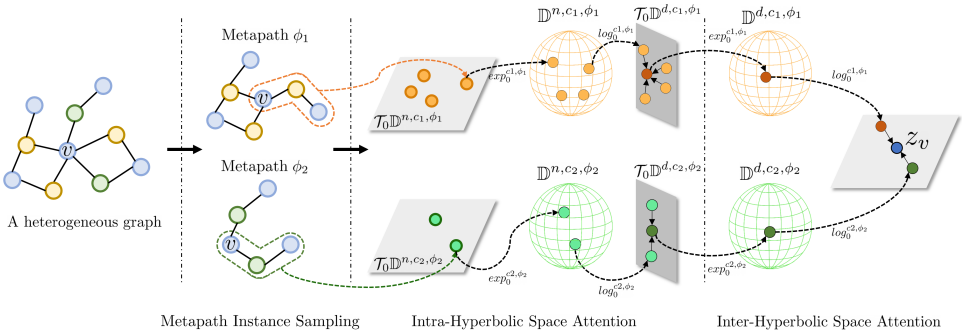


Fig. 3: The framework of proposed MSGAT.

metapath length l. We use breadth-first search for this procedure.

B. Intra-Hyperbolic Space Attention

1) Hyperbolic mean-linear encoder: After metapath instance sampling, we compose the hyperbolic mean-linear encoder to transform all the node features within a metapath instance into a single feature. This transformation function can be formulated as follows:

$$x_p^{\mathbb{E}} = \frac{1}{j}\sum_{i=1}^{j} x_i \,(j \le l)\,, \tag{1}$$

$$x_p^{\mathbb{H},\phi} = W_t \otimes_c exp_0^{c,\phi}\left(x_p^{\mathbb{E}}\right). \tag{2}$$

In Equation (1), $x_i \in \mathbb{R}^n$ denotes the features of node i, j denotes the length of the metapath instance p, $W_t \in \mathbb{R}^{n\times n}$ denotes a transformation matrix, and $x_p^{\mathbb{E}} \in \mathbb{R}^n$ denotes the Euclidean feature of the metapath instance p.

In Equation (2), given Euclidean metapath instance feature $x_p^{\mathbb{E}}$, we first map $x_p^{\mathbb{E}}$ to a metapath-specific hyperbolic space $\mathbb{D}^{n,c,\phi}$ via the exponential map $exp_0^{c,\phi} : \mathcal{T}_0\mathbb{D}^{n,c,\phi} \to \mathbb{D}^{n,c,\phi}$. To adopt exponential map, we assume $x_p^{\mathbb{E}}$ is included in the tangent space $\mathcal{T}_0\mathbb{D}^{n,c,\phi}$ at point $x = 0$. Note that, $x_p^{\mathbb{H},\phi} \in \mathbb{D}^{n,c,\phi}$ denotes hyperbolic metapath instance feature. Here, $\mathbb{D}^{n,c,\phi}$ is a metapath-specific hyperbolic space that effectively represents structural properties of metapath instances following a specific metapath ϕ.

Additionally, $-c \quad (c > 0)$ is a learnable parameter that represents the negative curvature of hyperbolic space, and each metapath-specific hyperbolic space for metapath ϕ has a distinct negative curvature.

2) Hyperbolic metapath instance embedding: We utilize hyperbolic linear transformation with a hyperbolic non-linear activation function to obtain metapath instance embedding in hyperbolic space. The formulation of this process is as follows:

$$h_p^{\mathbb{H},\phi} = \sigma \otimes_c \left(W_1 \otimes_c x_p^{\mathbb{H},\phi}\right) \oplus_c exp_0^{c,\phi}\left(b_1\right). \tag{3}$$

In Equation (3), $h_p^{\mathbb{H},\phi} \in \mathbb{D}^{d,c,\phi}$ is a latent representation of metapath instance p in metapath-specific hyperbolic space $\mathbb{D}^{d,c,\phi}$. Here, d is the dimension of hyperbolic space for latent metapath instance representations. Additionally, $W_1 \in \mathbb{R}^{d\times n}$ is a weight matrix and $b_1 \in \mathbb{R}^d$ is a bias vector.

3) Intra-metapath specific hyperbolic space attention: To aggregate different latent metapath instance representations, we define attention mechanisms in metapath-specific hyperbolic space. First, we calculate the importance of each metapath instance α_p as follows:

$$e_p = a^T \cdot log_0^{c,\phi}\left(h_p^{\mathbb{H},\phi}\right), \tag{4}$$

$$\alpha_p = \frac{\exp\left(e_p\right)}{\sum_{q\in\mathcal{P}_v^\phi}\exp\left(e_q\right)}. \tag{5}$$

In the above equations, e_p denotes the importance of each metapath instance $p \in \mathcal{P}_v^\phi$, where $log_0^{c,\phi} : \mathbb{D}^{d,c,\phi} \to \mathcal{T}_0\mathbb{D}^{d,c,\phi}$ denotes logarithmic map function and $\mathcal{P}_v^\phi$ denotes a subset of $\mathcal{P}_v$ consisting of metapath instances that follow a specific metapath ϕ and $a \in \mathbb{R}^d$ is an attention vector for metapath instance. After calculating the importance of each metapath instance, we apply the softmax function to these values to obtain the weight of each metapath instance.

Then the metapath-specific embedding for node v is obtained from the weight of each metapath instance and their latent representations in metapath-specific hyperbolic space. This process can be formulated as below:

$$h_v^\phi = \sigma \otimes_c \left(exp_0^{c,\phi}\left(\sum_{p\in\mathcal{P}_v^\phi}\alpha_p \cdot log_0^{c,\phi}\left(h_p^{\mathbb{H},\phi}\right)\right)\right), \tag{6}$$

where $h_v^\phi \in \mathbb{D}^{d,c,\phi}$ denotes metapath specific embedding for node v. Note that, in Equation (3) and (6), $\sigma\otimes_c$ denotes the hyperbolic non-linear activation function with LeakyReLU.

4) Hyperbolic multi-head attention: As shown in the equation below, we adopt multi-head attention in hyperbolic space to enhance metapath-specific embeddings and stabilize the learning process. Specifically, we divide the attention mechanisms into K independent attention mechanisms, conduct them in parallel, and then concatenate the metapath-specific embedding from each attention mechanism to obtain the final metapath-specific embedding h_v^ϕ.

$$h_v^\phi = \|_{k=1}^{K}\, \sigma \otimes_c \left(exp_0^{c,\phi}\left(\sum_{p\in\mathcal{P}_v^\phi}\alpha_p^k \cdot log_0^{c,\phi}\left(h_p^{\mathbb{H},\phi}\right)\right)\right). \tag{7}$$

C. Inter-Hyperbolic Space Attention

1) Node embedding space mapping: Once the metapath-specific embedding h_v^ϕ is obtained for each metapath ϕ in Φ, we aggregate them using attention mechanisms to learn the importance of each metapath-specific embedding, which represents semantic structural information.

First, we map metapath-specific embeddings to the same embedding space because they are included in different metapath-specific hyperbolic spaces $\mathbb{D}^{d,c,\phi}$ corresponding to specific metapaths ϕ. Since the curvatures of these metapath-specific hyperbolic spaces are different, it is difficult to aggregate them directly in their respective hyperbolic spaces. Instead, we map metapath-specific embeddings into the same tangent space from different hyperbolic spaces. The mapping operation can be formulated as follows:

$$g_v^\phi = W_2 \cdot log_0^{c,\phi}\left(h_v^\phi\right). \tag{8}$$

In Equation (8), with the logarithmic map $log_0^{c,\phi}$, we can map $h_v^\phi \in \mathbb{D}^{d,c,\phi}$ to tangent space $\mathcal{T}_0\mathbb{D}^{d,c,\phi}$ which is a plane like Euclidean space. Then, with the transformation matrix $W_2 \in \mathbb{R}^{d\times d}$, we can project metapath-specific embeddings within different metapath-specific tangent space $\mathcal{T}_0\mathbb{D}^{d,c,\phi}$ into same semantic space $\mathbb{R}^d$.

2) Metapath aggregation: Given mapped metapath-specific embeddings g_v^ϕ, we aggregate them by using attention mechanisms. First, we calculate the importance of each mapped metapath-specific embedding β_ϕ as follows:

$$e_\phi = b^T \cdot \tanh\left(W_3 \cdot g_v^\phi + b_3\right), \tag{9}$$

$$\beta_\phi = \frac{\exp\left(e_\phi\right)}{\sum_{\pi\in\Phi}\exp\left(e_\pi\right)}. \tag{10}$$

In the above equations, e_ϕ denotes the importance of each mapped metapath-specific embedding. After calculating e_ϕ, we normalize these values using the softmax function to obtain their weights. Note that, $W_3 \in \mathbb{R}^{d'\times d}$ denotes a weight matrix for the mapped metapath-specific embeddings, $b_3 \in \mathbb{R}^{d'}$ is a bias vector, and $b \in \mathbb{R}^{d'}$ is an attention vector. The embedding vector of node v, $z_v \in \mathbb{R}^{d'}$ is calculated as a weighted sum as shown below:

$$z_v = \sum_{\phi\in\Phi}\left(\beta_\phi \cdot g_v^\phi\right). \tag{11}$$

D. Model Training

As shown in Equation (12), we employ the non-linear transformation $f(\cdot)$ to map node embedding vectors into a space with the desired output dimension, conducting various downstream tasks.

$$f\left(z_v\right) = \sigma\left(W_o \cdot z_v\right), \tag{12}$$

where $W_o \in \mathbb{R}^{d_o\times d'}$ denotes the weight matrix, d_o denotes the dimension of the output vector, and σ is the activation function. Then, we train MSGAT by minimizing the loss function $\mathcal{L}$.

For node-level tasks, MSGAT is trained by minimizing the cross-entropy loss function $\mathcal{L}_n$ which is defined as below:

$$\mathcal{L}_n = -\sum_{v\in V_t}\sum_{c=1}^{C} y_v[c] \cdot \log\left(f(z_v)[c]\right), \tag{13}$$

where v_t is the target node set extracted from the labeled node set, C is the number of classes, y_v is the one-hot encoded label vector for node v, and $f(z_v)$ is a vector predicting the label probabilities of node v.

For a link-level task, MSGAT is trained by minimizing binary cross-entropy function $\mathcal{L}_l$ which is defined as below:

$$\begin{aligned}\mathcal{L}_l = -\frac{1}{|\mathcal{S}|}\sum_{(u,v,y)\in\mathcal{S}} & y \cdot \log\left(\sigma\left(z_u^T z_v\right)\right) \\ & + (1-y)\log\left(1-\sigma\left(z_u^T z_v\right)\right),\end{aligned} \tag{14}$$

where $\mathcal{S}$ is the set that includes both positive and negative node pairs, y is the ground truth label for a node pair (u, v). σ is the sigmoid function, z_u and z_v are the embedding vectors of nodes u and v, respectively.

IV. Experiments and Discussion

In this section, we assess the efficiency of our proposed model MSGAT, through experiments with four real-world heterogeneous graphs. We conduct comparative analyses between MSGAT and several state-of-the-art GNN models.

TABLE I: Statistics of real-world datasets.

Node Classification and Clustering

Dataset	# Nodes	# Links	# Classes	# Features
IMDB	Movie (M) : 4,661 Director (D) : 2,270 Actor (A) : 5,841	M-D : 4,661 M-A : 13,983	3	1,256
DBLP	Author (A) : 4,057 Paper (P) : 14,328 Conference (C) : 20	A-P : 19,645 P-C : 14,328	4	334
ACM	Paper (P) : 3,020 Author (A) : 5,912 Subject (S) : 57	P-A : 9,936 P-S : 3,025	3	1,902

Link Prediction

Dataset	# Nodes	# Links	Target	# Features
LastFM	User (U) : 1,892 Artist (A) : 17,632 Tag (T) : 1,088	U-A : 85,689 A-T : 21,553	User-Artist	20,612

A. Datasets

To evaluate the performance of MSGAT on downstream tasks, we use four real-world heterogeneous graph datasets. Table I shows the statistics of these datasets. For node classification and clustering tasks, Movie, Author, and Paper types of nodes are labeled.

B. Baselines

We compare MSGAT with several state-of-the-art graph neural networks categorized into four groups- **i) Euclidean homogeneous GNNs**: GCN [12] and GAT [13], **ii) Hyperbolic homogeneous GNNs**: HGCN [14], **iii) Euclidean heterogeneous GNNs**: HAN [1], MAGNN [15], GTN [2], HetSANN [16], HGT [17], GraphMSE [3], and Simple-HGN [18], and **iv) Hyperbolic heterogeneous GNNs**: McH-HGCN [19], SHAN [9], and HHGAT [10]. For homogeneous GNNs, features are processed to be homogeneous for pair comparison with heterogeneous GNNs.

C. Node Classification and Clustering

Node classification was performed by applying support vector machines on embedding vectors of labeled nodes. Macro-F1 and Micro-F1 were used as the evaluation metrics for classification accuracy. The ratio of training data was varied

TABLE II: Experimental results (%) for the node classification task.

Dataset	Metric	Train %	i		ii	iii						iv			
			GCN	GAT	HGCN	HAN	MAGNN	GTN	HGT	GraphMSE	Simple-HGN	McH-HGCN	SHAN	HHGAT	MSGAT
IMDB	Macro-F1	20%	52.17±0.35	53.68±0.26	54.38±0.48	56.19±0.51	59.33±0.38	58.74±0.74	56.14±0.65	57.72±0.56	59.97±0.61	58.16±0.49	62.23±0.76	63.16±0.39	**65.75±0.81**
		40%	53.20±0.45	56.33±0.71	57.05±0.43	56.84±0.37	60.70±0.48	59.71±0.54	57.12±0.53	62.01±0.48	61.94±0.39	60.31±0.56	63.98±0.68	65.07±0.63	**68.07±0.54**
		60%	54.35±0.46	56.93±0.54	57.86±0.56	58.95±0.71	60.68±0.56	61.88±0.50	61.52±0.57	65.51±0.61	66.73±0.42	61.93±0.33	66.68±0.71	65.72±0.56	**71.42±0.48**
		80%	54.19±0.29	57.25±0.18	57.92±0.32	58.61±0.63	61.15±0.55	62.08±0.62	63.69±0.59	67.34±0.56	67.56±0.45	62.29±0.50	68.49±0.67	67.42±0.51	**70.03±0.60**
	Micro-F1	20%	52.13±0.38	53.67±0.31	54.46±0.42	56.71±0.53	58.30±0.39	61.97±0.63	57.97±0.76	60.58±0.62	63.76±0.60	61.28±0.37	64.31±0.82	65.76±0.66	**69.09±0.93**
		40%	53.34±0.41	53.99±0.65	57.02±0.46	56.68±0.70	58.34±0.58	62.10±0.53	58.80±0.65	64.87±0.63	65.60±0.46	63.09±0.12	66.56±0.73	66.34±0.70	**70.95±0.54**
		60%	54.61±0.42	56.26±0.51	58.01±0.50	58.26±0.82	60.71±0.70	63.55±0.39	62.63±0.58	68.86±0.86	69.29±0.74	64.16±0.29	69.57±0.76	70.40±0.51	**73.60±0.39**
		80%	54.37±0.33	57.23±0.29	58.54±0.93	59.35±0.65	61.70±0.39	65.57±0.91	67.01±0.47	69.54±0.59	69.35±0.66	64.96±0.42	69.42±0.56	69.61±0.89	**73.37±0.59**
DBLP	Macro-F1	20%	87.51±0.15	91.52±0.34	91.69±0.38	92.63±0.46	93.21±0.64	92.45±0.37	90.36±0.62	93.80±0.39	93.48±0.56	90.63±0.72	94.27±0.16	94.19±0.08	**95.44±0.17**
		40%	88.55±0.46	91.07±0.39	91.93±0.35	92.35±0.64	93.51±0.29	92.39±0.41	91.57±0.29	94.02±0.50	93.98±0.27	91.74±0.62	94.33±0.08	94.27±0.10	**95.54±0.12**
		60%	89.44±0.27	91.51±0.46	92.60±0.89	92.86±0.37	93.59±0.60	93.77±0.52	92.32±0.19	94.30±0.26	94.01±0.33	92.26±0.19	94.50±0.29	94.90±0.30	**95.67±0.40**
		80%	89.45±0.36	91.77±0.27	92.58±0.39	92.73±0.66	94.36±0.43	94.46±0.60	93.46±0.55	94.21±0.82	94.25±0.57	93.13±0.24	94.67±0.12	94.77±0.19	**95.29±0.15**
	Micro-F1	20%	88.21±0.26	91.29±0.31	92.06±0.33	92.35±0.51	93.60±0.59	93.15±0.48	91.46±0.77	94.15±0.42	94.17±0.47	92.01±0.53	94.53±0.17	94.66±0.07	**95.79±0.16**
		40%	88.68±0.52	91.60±0.50	92.31±0.40	92.87±0.39	93.75±0.44	93.80±0.56	92.05±0.48	94.32±0.81	93.87±0.42	92.73±0.51	94.60±0.22	94.72±0.10	**95.90±0.11**
		60%	90.01±0.48	92.09±0.41	93.16±0.36	93.42±0.12	94.20±0.51	94.22±0.51	92.72±0.24	94.38±0.31	94.71±0.56	93.50±0.26	94.92±0.35	95.15±0.36	**95.98±0.36**
		80%	90.14±0.39	92.39±0.41	93.21±0.35	93.54±0.60	94.09±0.52	94.23±0.54	92.57±0.72	94.54±0.63	94.68±0.55	93.31±0.12	95.36±0.23	95.34±0.17	**95.85±0.16**
ACM	Macro-F1	20%	83.08±0.37	86.14±0.49	87.29±1.06	87.88±0.42	88.43±0.51	91.10±0.39	89.12±0.46	92.13±0.27	92.25±0.39	89.86±0.83	92.56±0.21	91.34±0.39	**92.73±0.52**
		40%	87.34±0.41	87.11±0.22	89.19±0.72	90.54±0.08	90.16±0.91	91.34±0.44	89.15±0.49	92.76±0.37	92.64±0.61	90.52±0.69	92.88±0.19	92.92±0.32	**93.95±0.51**
		60%	88.80±0.51	88.92±0.36	90.01±0.42	91.22±0.36	90.73±0.39	91.34±0.26	90.57±0.34	93.39±0.28	93.06±0.22	91.03±0.76	94.10±0.37	94.28±0.35	**94.83±0.16**
		80%	88.43±0.29	88.06±0.16	90.03±0.77	91.35±0.45	92.12±0.51	91.14±0.78	93.45±0.65	93.57±0.49	93.55±0.44	91.97±0.55	**94.94±0.62**	93.91±0.31	94.01±0.30
	Micro-F1	20%	87.75±0.33	87.83±0.47	88.09±0.89	91.20±0.46	91.37±0.42	91.86±0.40	89.59±0.37	92.27±0.36	91.91±0.33	90.21±0.61	92.38±0.18	92.36±0.37	**92.96±0.54**
		40%	87.86±0.42	87.39±0.41	90.06±0.73	91.78±0.28	92.60±0.48	91.89±0.46	90.70±0.43	93.05±0.36	92.86±0.84	90.63±0.52	93.37±0.26	93.46±0.50	**93.91±0.48**
		60%	88.40±0.56	87.78±0.33	90.51±0.63	92.39±0.42	92.21±0.18	92.07±0.48	91.18±0.15	93.38±0.47	93.33±0.21	91.20±0.48	94.46±0.35	94.34±0.39	**94.88±0.16**
		80%	88.56±0.33	87.87±0.51	91.10±0.44	92.03±0.16	92.14±0.48	92.21±0.66	91.77±0.57	93.37±0.36	93.53±0.42	92.06±0.66	**94.56±0.11**	93.72±0.32	94.05±0.31

TABLE III: Experimental results (%) for the node clustering task.

Dataset	Metric	i		ii	iii						iv			
		GCN	GAT	HGCN	HAN	MAGNN	GTN	HGT	GraphMSE	Simple-HGN	McH-HGCN	SHAN	HHGAT	MSGAT
IMDB	NMI	7.84±0.24	8.06±0.18	10.29±0.76	11.21±1.09	15.66±0.73	15.01±0.11	14.55±0.32	15.70±1.25	17.58±0.82	14.32±0.46	20.60±0.92	20.75±0.36	**24.06±0.51**
	ARI	8.12±0.40	8.86±0.09	11.10±0.88	11.49±0.11	16.72±0.21	15.96±0.63	16.59±0.36	16.38±0.74	19.51±1.06	16.91±0.37	22.56±0.22	22.80±0.68	**26.33±0.46**
DBLP	NMI	75.37±0.25	75.46±0.44	76.48±0.87	77.03±0.16	80.11±0.30	81.39±0.73	79.02±0.39	37.22±0.65	82.38±0.07	78.90±0.31	82.39±0.42	83.14±0.19	**84.38±0.59**
	ARI	77.14±0.21	77.99±0.72	79.36±0.95	82.53±0.42	85.61±0.38	84.12±0.83	80.28±0.20	34.21±0.65	85.71±0.33	81.22±0.56	86.13±0.33	85.91±0.49	**88.27±0.63**
ACM	NMI	51.73±0.21	58.06±0.46	60.19±0.69	61.24±0.12	64.73±0.47	65.06±0.35	67.88±0.20	66.65±0.44	69.91±0.68	66.76±0.38	72.90±0.93	72.49±0.44	**73.33±0.79**
	ARI	53.42±0.48	59.61±0.42	62.06±0.70	64.11±0.26	66.84±0.25	65.80±0.49	72.56±0.13	73.89±0.33	72.07±0.51	71.84±0.48	77.73±0.44	77.92±0.80	**78.28± 1.07**

TABLE IV: Experimental results (%) for the link prediction task.

Dataset	Metric	i		ii	iii					iv	
		GCN	GAT	HGCN	HAN	MAGNN	HetSANN	HGT	Simple-HGN	HHGAT	MSGAT
LastFM	ROC-AUC	43.68±0.30	44.52±0.22	46.71±0.78	48.32±0.28	49.37±0.59	50.28±0.45	47.78±0.23	53.85±0.47	54.37±0.51	**55.77±0.62**
	F1-Score	56.15±0.16	56.84±0.07	57.23±0.66	57.11±0.49	58.37±0.32	60.61±0.54	61.16±0.57	63.02±0.35	62.85±0.48	**63.39±0.76**

within the range of 20% to 80%. For node clustering, the k-means clustering algorithm was applied to embedding vectors of labeled nodes. Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) were used as the evaluation metrics for clustering accuracy.

As shown in Table II and III, MSGAT achieved better performance than other baselines in most cases. The results from MSGAT and HHGAT indicate the effectiveness of using multi-hyperbolic space to learn metapath instances. Specifically, because HHGAT employs a single hyperbolic space to learn metapath instances, it cannot effectively capture the complex structures following various power-law distributions of metapath instances in a single hyperbolic space. In contrast, MSGAT effectively captures various complex structures by using multi-hyperbolic space to learn metapath instances. In each metapath-specific hyperbolic space corresponding to a distinct metapath, a learned negative curvature effectively represents the distribution of node degrees for metapath instances following that metapath. Furthermore, through intra-hyperbolic space attention and inter-hyperbolic space attention, MSGAT effectively captures important complex structures and semantic information within a heterogeneous graph, respectively. Moreover, a comparison of MSGAT and McH-HGCN demonstrates that MSGAT effectively captures a broader range of semantic information and structural properties by extensively sampling the surrounding structure of the target node, in contrast to McH-HGCN.

On the one hand, a comparison of Euclidean homogeneous GNNs and hyperbolic homogeneous GNNs demonstrates the effectiveness of hyperbolic space in representing complex structures within heterogeneous graphs. However, from a comparison of hyperbolic homogeneous GNNs and Euclidean heterogeneous GNNs, we observe that while hyperbolic homogeneous GNNs can learn complex structures, they struggle to capture the heterogeneity within heterogeneous graphs, thus failing to learn semantic information effectively. In contrast, Euclidean heterogeneous GNNs excel at learning semantic information within such graphs. At last, comparing Euclidean heterogeneous GNNs and hyperbolic heterogeneous GNNs, we can conclude that hyperbolic heterogeneous GNNs are more effective than Euclidean heterogeneous GNNs because hyperbolic heterogeneous GNNs can simultaneously learn the complex structures and heterogeneity of heterogeneous graphs.

D. Link Prediction

We also conducted a link prediction task on the LastFM dataset. To predict the probabilities of relations between user-type nodes and artist-type nodes, we used a dot product operation applied to the embeddings of the two types of nodes. The area under the ROC curve (ROC-AUC) and F1-score were used as the evaluation metrics for prediction accuracy. We considered all connected user-artist pairs as positive samples,

TABLE V: Results of the ablation study.

Dataset	IMDB				DBLP				ACM			
Metric	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	NMI	ARI
MSGAT	**71.42±0.48**	**73.60±0.39**	**24.06±0.51**	**26.33±0.46**	**95.67±0.40**	**95.98±0.36**	**84.38±0.59**	**88.27±0.63**	**94.83±0.16**	**94.88±0.16**	**73.33±0.79**	**78.28±1.07**
$\text{MSGAT}_{\text{CONCAT}}$	68.03±0.44	70.75±0.44	22.24±0.55	24.91±0.53	94.23±0.50	94.54±0.47	82.06±0.79	84.65±0.88	93.33±0.07	93.91±0.26	72.73±0.44	75.82±0.27
$\text{MSGAT}_{\text{EUCLID}}$	64.72±0.10	67.17±0.12	16.72±0.18	14.65±0.44	93.09±0.35	93.51±0.33	79.32±0.94	83.84±1.09	90.02±1.15	90.09±1.18	70.24±1.93	73.18±2.26
$\text{MSGAT}_{\text{SINGLE}}$	66.06±0.56	68.77±0.61	21.65±0.59	25.91±0.80	93.76±0.42	93.94±0.33	80.95±0.32	86.06±0.56	92.39±0.29	92.67±0.26	73.16±0.07	77.93±0.38

while unconnected user-artist pairs were considered as negative samples. For model training, an equal number of positive and negative samples were used.

As shown in Table IV, MSGAT outperforms the other baselines. Compared MSGAT with HHGAT, in link prediction, when predicting the connection between two nodes of different types, the metapath defined around each type of node changes completely, and the distribution of metapath instances also differs. Consequently, while MSGAT can flexibly learn from various metapath instance distributions, HHGAT is unable to do so, making MSGAT superior to HHGAT in link prediction.

E. Ablation Study

We compose three variants of MSGAT to validate the effectiveness of each component of MSGAT. $\text{MSGAT}_{\text{CONCAT}}$ concatenates node features within metapath instances, instead of using hyperbolic mean-linear encoder, $\text{MSGAT}_{\text{EUCLID}}$ uses Euclidean space for embedding space instead of hyperbolic space, and $\text{MSGAT}_{\text{SINGLE}}$ uses only one hyperbolic space to learn metapath instance embeddings. Note that, the training percentage for node classification is set to 60%.

We report the results of the ablation study in Table V. Comparing MSGAT with $\text{MSGAT}_{\text{CONCAT}}$, we observe that transforming node features to the metapath-specific hyperbolic space is more effective than simply concatenating node features within the metapath instance. Next, a comparison of $\text{MSGAT}_{\text{EUCLID}}$ and $\text{MSGAT}_{\text{SINGLE}}$ demonstrates that using hyperbolic space as the embedding space is more effective for learning heterogeneous graphs than using Euclidean space. Moreover, comparing MSGAT with $\text{MSGAT}_{\text{SINGLE}}$, we can see that the use of multi-hyperbolic space leads to significant performance improvements. This is due to the ability of MSGAT to effectively learn the complex structures of diverse distributions within heterogeneous graphs.

V. Conclusion

In this paper, we propose the Multi-hyperbolic Space-based heterogeneous Graph Attention Network (MSGAT). Instead of the Euclidean space, MSGAT uses multiple hyperbolic spaces to capture various power-law structures effectively, and finally MSGAT aggregates metapath-specific embeddings to obtain more enhanced node representations.

We conduct comprehensive experiments to evaluate the effectiveness of MSGAT with widely used real-world heterogeneous graph datasets. The experimental results demonstrate that MSGAT outperforms the other state-of-the-art baselines. Additionally, it has been shown that using multiple hyperbolic spaces for learning various power-law distributions is effective.

For the future work, we plan to develop methods to enhance the interpretability of the hyperbolic spaces learned for each metapath in heterogeneous graphs.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2023-00214065) and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2022-00155857, Artificial Intelligence Convergence Innovation Human Resources Development (Chungnam National University), No.20200-00004, Development of Previsional Intelligence based on Long-term Visual Memory Network).

References

[1] 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.

[2] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *NeurIPS*, 2019, pp. 11 960–11 970.

[3] Y. Li, Y. Jin, G. Song, Z. Zhu, C. Shi, and Y. Wang, "Graphmse: Efficient meta-path selection in semantically aligned feature space for graph neural networks," in *AAAI*, 2021, pp. 4206–4214.

[4] H. Pei, B. Wei, K. Chang, C. Zhang, and B. Yang, "Curvature regularization to prevent distortion in graph embedding," in *NeurIPS*, 2020, pp. 20 779–20 790.

[5] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *NIPS*, 2017, pp. 6338–6348.

[6] I. Balazevic, C. Allen, and T. Hospedales, "Multi-relational poincaré graph embeddings," in *NeurIPS*, 2019, pp. 4465–4475.

[7] Z. Pan and P. Wang, "Hyperbolic hierarchy-aware knowledge graph embedding for link prediction," in *Findings of EMNLP*, 2021, pp. 2941–2948.

[8] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *AAAI*, 2019, pp. 5337–5344.

[9] J. Li, Y. Sun, and M. Shao, "Multi-order relations hyperbolic fusion for heterogeneous graphs," in *CIKM*, 2023, pp. 1358–1367.

[10] J. Park, S. Han, S. Jeong, and S. Lim, "Hyperbolic heterogeneous graph attention networks," in *WWW*, 2024, pp. 561–564.

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

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[14] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *NeurIPS*, 2019, pp. 4869–4880.

[15] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020, pp. 2331–2341.

[16] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," in *AAAI*, vol. 34, no. 04, 2020, pp. 4132–4139.

[17] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020, pp. 2704–2710.

[18] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks," in *KDD*, 2021, pp. 1150–1160.

[19] Y. Liu and B. Lang, "Mch-hgcn: Multi-curvature hyperbolic heterogeneous graph convolutional network with type triplets," *Neural Computing and Applications*, vol. 35, no. 20, pp. 15 033–15 049, 2023.