A Self-supervised Mixed-curvature Graph Neural Network

Li Sun[1]*, Zhongbao Zhang[2], Junda Ye[2], Hao Peng[3], Jiawei Zhang[4], Sen Su[2] and Philip S. Yu[5].

[1]School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China
[2]School of Computer Science, Beijing University of Posts and Telecommunications, China
[3]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China
[4]IFM Lab, Department of Computer Science, University of California, Davis, CA, USA
[5]Department of Computer Science, University of Illinois at Chicago, IL, USA
ccesunli@ncepu.edu.cn; {zhongbaozb, susen}@bupt.edu.cn; penghao@act.buaa.edu.cn; jiawei@ifmlab.org; psyu@uic.edu

Abstract

Graph representation learning received increasing attentions in recent years. Most of the existing methods ignore the complexity of the graph structures and restrict graphs in a single constant-curvature representation space, which is only suitable to particular kinds of graph structure indeed. Additionally, these methods follow the supervised or semi-supervised learning paradigm, and thereby notably limit their deployment on the unlabeled graphs in real applications. To address these aforementioned limitations, we take the first attempt to study the self-supervised graph representation learning in the mixed-curvature spaces. In this paper, we present a novel **Self**-supervised **M**ixed-curvature **G**raph **N**eural **N**etwork (**SELFMGNN**). To capture the complex graph structures, we construct a *mixed-curvature* space via the Cartesian product of multiple Riemannian component spaces, and design hierarchical attention mechanisms for learning and fusing graph representations across these component spaces. To enable the self-supervisd learning, we propose a novel *dual contrastive approach*. The constructed mixed-curvature space actually provides multiple Riemannian views for the contrastive learning. We introduce a Riemannian projector to reveal these views, and utilize a well-designed Riemannian discriminator for the *single-view* and *cross-view contrastive learning* within and across the Riemannian views. Finally, extensive experiments show that SELFMGNN captures the complex graph structures and outperforms state-of-the-art baselines.

Introduction

Graph representation learning (**??**) shows fundamental importance in various applications, such as link prediction and node classification (**?**), and thus receives increasing attentions from both academics and industries. Meanwhile, we have also observed great limitations with the existing graph representation learning methods in two major perspectives, which are described as follows:

Representation Space: Most of existing methods ignore the complexity of real graph structures, and limit the graphs in a single *constant-curvature* representation space (**?**). Such methods can only work well on particular kinds of structure that they are designed for. For instance, the constant negative curvature hyperbolic space is well-suited for graphs with hierarchical or tree-like structures (**?**). The constant positive curvature spherical space is especially suitable for data with cyclical structures, e.g., triangles and cliques (**?**), and the zero-curvature Euclidean space for grid data (**?**). However, graph structures in reality are usually mixed and complicated rather than uniformed, in some regions hierarchical, while in others cyclical (**??**). Even more challenging, the curvatures over different hierarchical or cyclical regions can be different as will be shown in this paper. In fact, it calls for a new representation space to match the wide variety of graph structures, and we seek spaces of *mixed-curvature* to provide better representations.

Learning Paradigm: Learning graph representations usually requires abundant supervision label information (**??**). Labels are usually scarce in real applications, and undoubtedly, labeling graphs is expensive—manual annotation or paying for permission, and is even impossible to acquire because of the privacy policy. Fortunately, the rich information in graphs provides the potential for *self-supervised learning*, i.e., learning representations without labels (**?**). Self-supervised graph representation learning is a more favorable choice, particularly when we intend to take the advantages from the unlabeled graphs in real applications. Recently, contrastive learning (**??**) emerges as a successful method for the graph self-supervised learning. However, existing self-supervised methods, to the best of our knowledge, cannot be applied to the mixed-curvature spaces due to the intrinsic differences in the geometry.

To address these aforementioned limitations, we take the first attempt to study the *self-supervised graph representation learning in the mixed-curvature space* in this paper.

To this end, we present a novel **Self**-supervised **M**ixed-curvature **G**raph **N**eural **N**etwork, named **SELFMGNN**. To address the first limitation, we propose to learn the representations in a *mixed-curvature space*. Concretely, we first construct a mixed-curvature space via the Cartesian product of multiple Riemannian—hyperbolic, spherical and Eu-

*Corresponding Author: Li Sun, ccesunli@ncepu.edu.cn

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

clidean—component spaces, jointly enjoying the strength of different curvatures to match the complicated graph structures. Then, we introduce hierarchical attention mechanisms for learning and fusing representations in the product space. In particular, we design an intra-component attention for the learning within a component space and an inter-component attention for the fusing across component spaces. To address the second limitation, we propose a novel *dual contrastive approach* to enable the self-supervisd learning. The constructed mixed-curvature space actually provides multiple Riemannian views for contrastive learning. Concretely, we first introduce a Riemannian projector to reveal these views, i.e., hyperbolic, spherical and Euclidean views. Then, we introduce the *single-view* and *cross-view contrastive learning*. In particular, we utilize a well-designed Riemannian discriminator to contrast positive and negative samples in the same Riemannian view (i.e., the single-view contrastive learning) and concurrently contrast between different Riemannian views (i.e., the cross-view contrastive learning). In the experiments, we study the curvatures of real graphs and show the advantages of allowing multiple positive and negative curvature components for the first time, demonstrating the superiority of SELFMGNN.

Overall, our main contributions are summarized below:

- *Problem*: To the best of our knowledge, this is the first attempt to study the self-supervised graph representation learning in the mixed-curvature space.
- *Model*: This paper presents a novel SELFMGNN model, where hierarchical attention mechanisms and dual contrastive approach are designed for self-supervised learning in the mixed-curvature space, allowing multiple hyperbolic (spherical) components with distinct curvatures.
- *Experiments*: Extensive experiments show the curvatures over different hierarchical (spherical) regions of a graph can be different. SELFMGNN captures the complicated graph structures without labels and outperforms the state-of-the-art baselines.

Preliminaries and Problem Definition

In this section, we first present the preliminaries and notations necessary to construct a mixed-curvature space. Then, we formulate the problem of *self-supervised graph representation learning in the mixed-curvature space*.

Riemannian Manifold

A smooth *manifold* $\mathcal{M}$ generalizes the notion of the surface to higher dimensions. Each point $\mathbf{x} \in \mathcal{M}$ associates with a *tangent space* $\mathcal{T}_{\mathbf{x}}\mathcal{M}$, the first order approximation of $\mathcal{M}$ around $\mathbf{x}$, which is locally Euclidean. On tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$, the *Riemannian metric*, $g_{\mathbf{x}}(\cdot,\cdot) : \mathcal{T}_{\mathbf{x}}\mathcal{M} \times \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathbb{R}$, defines an inner product so that geometric notions can be induced. The tuple $(\mathcal{M}, g)$ is called a *Riemannian manifold*.

Transforming between the tangent space and the manifold is done via exponential and logarithmic maps, respectively. For $\mathbf{x} \in \mathcal{M}$, the *exponential map* at $\mathbf{x}$, $\mathbf{exp}_{\mathbf{x}}(\mathbf{v}) : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$, projects the vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ onto the manifold $\mathcal{M}$. The *logarithmic map* at $\mathbf{x}$, $\mathbf{log}_{\mathbf{x}}(\mathbf{y}) : \mathcal{M} \to \mathcal{T}_{\mathbf{x}}\mathcal{M}$, projects the vector $\mathbf{y} \in \mathcal{M}$ back to the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. For further expositions, please refer to mathematical materials (**??**).

Constant Curvature Space

The Riemannian metric also defines a curvature at each point $\kappa(\mathbf{x})$, which determines how the space is curved. If the curvature is uniformly distributed, $(\mathcal{M}, g)$ is called a *constant curvature space* of curvature κ. There are 3 canonical types of constant curvature space that we can define with respect to the sign of the curvature: a positively curved spherical space $\mathbb{S}$ with $\kappa > 0$, a negatively curved hyperbolic space $\mathbb{H}$ with $\kappa < 0$ and the flat Euclidean space $\mathbb{E}$ with $\kappa = 0$.
Note that, $\|\cdot\|_2$ denotes the Euclidean norm in this paper.

Problem Definition

In this paper, we propose to study the self-supervised graph representation learning in the mixed-curvature space. Without loss of generality, a graph is described as $G = (V, E, \mathbf{X})$, where $V = \{v_1, \cdots, v_n\}$ is the node set and $E = \{(v_i, v_j)|\ v_i, v_j \in V\}$ is the edge set. We summarize the edges in the adjacency matrix $\mathbf{G}$, where $\mathbf{G}_{ij} = 1$ iff $(v_i, v_j) \in E$, otherwise 0. Each node v_i is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, and matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ represents the features of all nodes. Now, we give the studied problem:

Problem Definition (Self-supervised graph representation learning in the mixed-curvature space). *Given a graph $G = (V, E, \mathbf{X})$, the problem of self-supervised graph representation learning in the mixed-curvature space is to learn an encoding function $\Phi : V \to \mathcal{P}$ that maps the node v to a vector $\boldsymbol{z}$ in a mixed-curvature space $\mathcal{P}$ that captures the intrinsic complexity of graph structure without using any label information.*

In other words, the graph representation model should align with the complex graph structures — hierarchical as well as cyclical structure, and can be learned without external guidance (labels). Graphs in reality are usually mixed-curvatured rather than structured uniformly, i.e., in some regions hierarchical, while in others cyclical. A constant-curvature model (e.g., hyperbolic, spherical or the Euclidean model) benefits from their specific bias to better fit particular structure types. To bridge this gap, we propose to work with the **mixed-curvature space** to cover the complex graph structures in real-world applications.

SELFMGNN: Our Proposed Model

To address this problem, we present a novel **Self**-supervised **M**ixed-curvature **G**raph **N**eural **N**etwork (**SELFMGNN**). In a nutshell, SELFMGNN learns graph representations in the mixed-curvature space, and is equipped with a dual contrastive approach to enable its self-supervised learning. We illustrate the architecture of SELFMGNN in Fig. 1. We will elaborate on the mixed-curvature graph representation learning and the dual contrastive approach in following sections.

Mixed-curvature GNN

To learn the graph representations in a mixed-curvature space, we construct a mixed-curvature space by the Cartesian product of multiple Riemannian component spaces, in

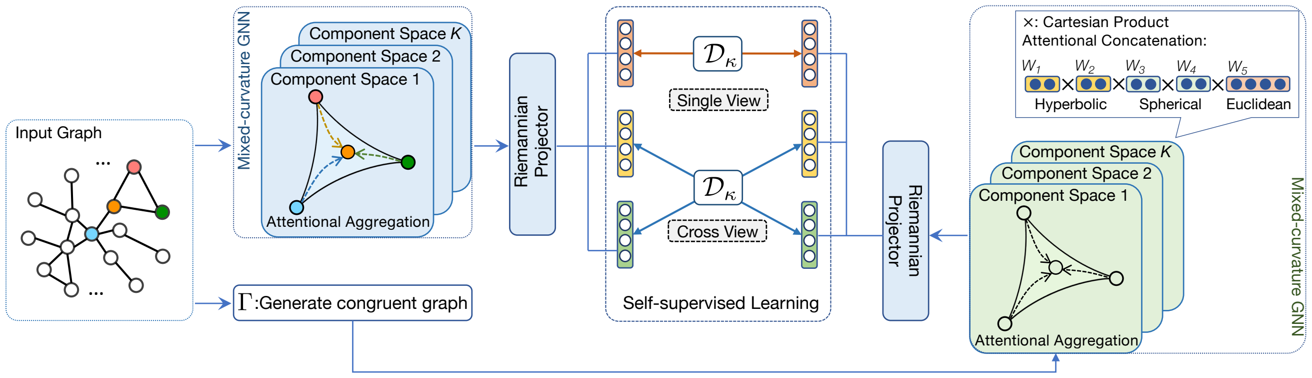

Figure 1: Overall architecture of SELFMGNN: In SELFMGNN, we first introduce a mixed-curvature GNN to learn graph representations. Specifically, we perform attentional aggregation within the component space where the triangle is to show its geometry, e.g., triangles curve inside in $\mathbb{H}$, and attentional concatenation among component spaces whose example with learnable weights is on the top right. Then, to enable self-supervised learning, we design a Riemannian projector to reveal different views of the mixed-curvature space, and utilize a well-designed Riemannian discriminator $\mathcal{D}_\kappa$ to contrast samples for single- and cross-view contrastive learning, shown in the middle. In practice, we feed the graph and its congruent augmentation, generated by $\Gamma(\cdot)$, for the contrastive learning as specified in Algo. 1.

which we propose a mixed-curvature GNN with hierarchical attention mechanisms. In particular, we first stack *intra-component attentional layers* in each component space to learn constant-curvature component representations. Then, we design an *inter-component attentional layer* across component spaces to fuse these component representations so as to obtain the output mixed-curvature representations matching the complex graph structures.

Constructing a Mixed-Curvature Space: We leverage the *Cartesian product* to construct the mixed-curvature space. With K constant-curvature spaces $\{\mathcal{M}_i\}_{i=1}^K$ indexed by subscript i, we perform the Cartesian product over them and obtain the resulting product space $\mathcal{P} = \times_{i=1}^K \mathcal{M}_i$, where $\times$ denotes the Cartesian product, and $\mathcal{M}_i$ is known as a component space. By fusing multiple constant-curvature spaces, the product space is constructed with non-constant mixed curvatures, matching the complex graph structures.

The product space $\mathcal{P}$ with the dimensionality d is described by its *signature*, which has three degrees of freedom per component: i) the model $\mathcal{M}_i$, ii) the dimensionality d_i and iii) the curvature κ_i, where $\sum_{i=1}^K d_i = d$. We use a shorthand notation for repeated components, $(\mathcal{M}_k)^j = \times_{i=1}^j \mathcal{M}_k$. Note that, SELFMGNN can have multiple hyperbolic or spherical components with distinct learnable curvatures, and such a design enables us to cover a wider range of curvatures for the better representation. However, we only need one Euclidean space, since the Cartesian product of Euclidean space is $\mathbb{E}^{d_0} = \times_{i=1}^j \mathbb{E}_i$, and $\sum_{i=1}^j d_i = d_0$.

The Cartesian product introduces a simple and interpretable combinatorial construction of the mixed-curvature space. For $\mathbf{x} = ||_{i=1}^K \mathbf{x}_{\mathcal{M}_i}$ in product space $\mathcal{P}$, $\mathbf{x}_{\mathcal{M}_i}$ denotes the component embedding in $\mathcal{M}_i$ and $||$ denotes the vector concatenation. Thanks to the combinatorial construction, we can first learn representations in each component space and then fuse these representations in the product space.

Model and Operations: Prior to discussing about the learning and fusing of the representations, we give the model of component spaces and provide the generalized Riemannian operations in the component spaces in this part.

We opt for the κ-*stereographic model* as it *unifies spaces of both positive and negative curvature* and *unifies operations with gyrovector formalism.* Specifically, the κ-stereographic model is a smooth manifold $\mathcal{M}_\kappa^d = \{\boldsymbol{z} \in \mathbb{R}^d | -\kappa ||\boldsymbol{z}||_2^2 < 1\}$, whose origin is $\mathbf{0} \in \mathbb{R}^d$, equipped with a Riemannian metric $g_{\boldsymbol{z}}^\kappa = (\lambda_{\boldsymbol{z}}^\kappa)^2 \mathbf{I}$, where $\lambda_{\boldsymbol{z}}^\kappa$ is given below:

$$\lambda_{\boldsymbol{z}}^\kappa = 2\left(1 + \kappa ||\boldsymbol{z}||_2^2\right)^{-1}. \tag{1}$$

In particular, $\mathcal{M}_\kappa^d$ is the stereographic sphere model for spherical space ($\kappa > 0$), while it is the Poincaré ball model of radius $1/\sqrt{-\kappa}$ for hyperbolic space ($\kappa < 0$). We summarize all the necessary operations for this paper in Table 1 with the curvature-aware definition of trigonometric functions. Specifically, $\tan_\kappa(\cdot) = \tan(\cdot)$ if $\kappa < 0$ and $\tan_\kappa(\cdot) = \tanh(\cdot)$ if $\kappa > 0$. Note that, the bold letter denotes the vector on the manifold.

Intra-Component Attentional Layer: This is the building block layer of the proposed mixed-curvature GNN. In this layer, we update node representations by attentionally aggregating the representations of its neighbors in the constant-curvature component space. As the importance of neighbors is usually different, we introduce the *intra-component attention* to learn the importance of the neighbors. Specifically, we first lift to the tangent space via $\mathbf{log}_{\mathbf{0}}^\kappa$ and model the importance parameterized by $\boldsymbol{\theta}_{\text{intra}}$ as follows:

$$att_{\text{intra}}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma\left(\boldsymbol{\theta}_{\text{intra}}^\top (\mathbf{W}\mathbf{log}_{\mathbf{0}}^\kappa(\boldsymbol{z}_i) || \mathbf{W}\mathbf{log}_{\mathbf{0}}^\kappa(\boldsymbol{z}_j))\right), \tag{2}$$

where $\mathbf{W}$ is the shared weight matrix and $\sigma(\cdot)$ denotes the sigmoid activation. $\mathbf{log}_{\mathbf{0}}^\kappa$ and $\mathbf{exp}_{\mathbf{0}}^\kappa$ are exponential and logarithmic maps defined in Table 1, respectively. Then, we compute the attention weight via softmax:

$$\hat{\mathbf{A}}_{ij} = \frac{\exp(att_{\text{intra}}(\boldsymbol{z}_i, \boldsymbol{z}_j))}{\sum_{k \in \mathcal{N}_i} \exp(att_{\text{intra}}(\boldsymbol{z}_i, \boldsymbol{z}_k))}, \tag{3}$$

Table 1: Summary of the operations in constant-curvature space (hyperbolic $\mathbb{H}^d$, spherical $\mathbb{S}^d$ and Euclidean space $\mathbb{E}^d$).

Operation	**Formalism in $\mathbb{E}^d$**	**Unified formalism in κ-stereographic model ($\mathbb{H}^d$/ $\mathbb{S}^d$)**
Distance Metric	$d^\kappa_{\mathcal{M}}(\mathbf{x},\mathbf{y}) = \Vert\mathbf{x}-\mathbf{y}\Vert_2$	$d^\kappa_{\mathcal{M}}(\mathbf{x},\mathbf{y}) = \frac{2}{\sqrt{\lvert\kappa\rvert}}\tan^{-1}_\kappa\left(\sqrt{\lvert\kappa\rvert}\,\Vert-\mathbf{x}\oplus_\kappa\mathbf{y}\Vert_2\right)$
Exponential Mapping	$\mathbf{exp}^\kappa_{\mathbf{x}}(\mathbf{v}) = \mathbf{x}+\mathbf{v}$	$\mathbf{exp}^\kappa_{\mathbf{x}}(\mathbf{v}) = \mathbf{x}\oplus_\kappa\left(\tan_\kappa\left(\sqrt{\lvert\kappa\rvert}\frac{\lambda^\kappa_{\mathbf{x}}\Vert\mathbf{v}\Vert_2}{2}\right)\frac{\mathbf{v}}{\sqrt{\lvert\kappa\rvert}\Vert\mathbf{v}\Vert_2}\right)$
Logarithmic Mapping	$\mathbf{log}^\kappa_{\mathbf{x}}(\mathbf{y}) = \mathbf{x}-\mathbf{y}$	$\mathbf{log}^\kappa_{\mathbf{x}}(\mathbf{y}) = \frac{2}{\sqrt{\lvert\kappa\rvert}\lambda^\kappa_{\mathbf{x}}}\tan^{-1}_\kappa\left(\sqrt{\lvert\kappa\rvert}\,\Vert-\mathbf{x}\oplus_\kappa\mathbf{y}\Vert_2\right)\frac{-\mathbf{x}\oplus_\kappa\mathbf{y}}{\Vert-\mathbf{x}\oplus_\kappa\mathbf{y}\Vert_2}$
Addition	$\mathbf{x}\oplus_\kappa\mathbf{y} = \mathbf{x}+\mathbf{y}$	$\mathbf{x}\oplus_\kappa\mathbf{y} = \frac{\left(1+2\kappa\mathbf{x}^T\mathbf{y}+K\Vert\mathbf{y}\Vert^2\right)\mathbf{x}+\left(1-\kappa\vert\vert\mathbf{x}\vert\vert^2\right)\mathbf{y}}{1+2\kappa\mathbf{x}^T\mathbf{y}+\kappa^2\vert\vert\mathbf{x}\vert\vert^2\vert\vert\mathbf{v}\vert\vert^2}$
Scalar-vector Multiplication	$r\otimes_\kappa\mathbf{x} = r\mathbf{x}$	$r\otimes_\kappa\mathbf{x} = \mathbf{exp}^\kappa_{\mathbf{0}}\left(r\,\mathbf{log}^\kappa_{\mathbf{0}}(\mathbf{x})\right)$
Matrix-vector Multiplication	$\mathbf{M}\otimes_\kappa\mathbf{x} = \mathbf{M}\mathbf{x}$	$\mathbf{M}\otimes_\kappa\mathbf{x} = \mathbf{exp}^\kappa_{\mathbf{0}}\left(\mathbf{M}\,\mathbf{log}^\kappa_{\mathbf{0}}(\mathbf{x})\right)$
Applying Functions	$f^{\otimes_\kappa}(\mathbf{x}) = f(\mathbf{x})$	$f^{\otimes_\kappa}(\mathbf{x}) = \mathbf{exp}^\kappa_{\mathbf{0}}\left(f\left(\mathbf{log}^\kappa_{\mathbf{0}}(\mathbf{x})\right)\right)$

where $\mathcal{N}_i$ is neighborhood node index of node v_i in the graph. Finally, we add a self-loop to keep its initial information, i.e., we have $\mathbf{A} = \mathbf{I} + \hat{\mathbf{A}}$.

Aggregation in traditional Euclidean space is straightforward. However, aggregation in hyperbolic or spherical space is challenging as the space is curved. To bridge this gap, we define the row-wise *κ-left-matrix-multiplication* $\boxtimes_\kappa$ similar to (**?**). Let rows of $\mathbf{Z}$ hold the vectors in κ-stereographic model. We have

$$(\mathbf{A}\boxtimes_\kappa\mathbf{Z})_{i\bullet} := A\otimes_\kappa \mathbf{mid}_\kappa\left(\{\mathbf{Z}_{i\bullet}\}^n_{i=1};\mathbf{A}_{i\bullet}\right), \qquad (4)$$

where $A = \sum_j \mathbf{A}_{ij}$, $(\cdot)_{i\bullet}$ denotes the i^{th} row and $\mathbf{mid}_\kappa$ denotes the *midpoint* defined as follows:

$$\mathbf{mid}_\kappa\left(\{\mathbf{Z}_{i\bullet}\}^n_{i=1};\mathbf{A}_{i\bullet}\right) = \frac{1}{2}\otimes_\kappa\left(\sum_{l=1}^n \frac{\mathbf{A}_{il}\lambda^\kappa_{\mathbf{Z}_{i\bullet}}}{\sum_{j=1}^n \mathbf{A}_{ij}(\lambda^\kappa_{\mathbf{Z}_{i\bullet}}-1)}\mathbf{Z}_{i\bullet}\right), \qquad (5)$$

where notation $\lambda^\kappa_{\mathbf{Z}_{i\bullet}}$ has been defined in Eq. (1) before. *We show that κ-left-matrix-multiplication $\boxtimes_\kappa$ performs attentional aggregation, i.e., **Theorem 1**.*

With the attention and aggregation above, we are ready to formulate the unified intra-component layer in component $\mathcal{M}_i$ of arbitrary curvature κ_i. Given the input $\mathbf{Z}^{(\ell-1)}_{\mathcal{M}_i}$ holding embeddings in its rows, the ℓ^{th} layer outputs:

$$\mathbf{Z}^{(\ell)}_{\mathcal{M}_i} = \sigma^{\otimes_\kappa}\left(\mathbf{A}^{(\ell)}_{\mathcal{M}_i}\boxtimes_\kappa\left(\mathbf{Z}^{(\ell-1)}_{\mathcal{M}_i}\otimes_\kappa\mathbf{W}^{(\ell-1)}_{\mathcal{M}_i}\right)\right), \qquad (6)$$

where we define the κ-right-matrix-multiplication below:

$$\left(\mathbf{Z}^{(\ell-1)}_{\mathcal{M}_i}\otimes_\kappa\mathbf{W}^{(\ell-1)}_{\mathcal{M}_i}\right)_{i\bullet} := \mathbf{exp}^\kappa_{\mathbf{0}}\left(\mathbf{log}^\kappa_{\mathbf{0}}\left((\mathbf{Z}^{(\ell-1)}_{\mathcal{M}_i})_{i\bullet}\right)\mathbf{W}^{(\ell-1)}_{\mathcal{M}_i}\right). \qquad (7)$$

In particular, we have $\mathbf{Z}^{(0)}_{\mathcal{M}_i}$ hold the input features in the κ-stereographic model. After stacking L layers, we have the output matrix $\mathbf{Z}_{\mathcal{M}_i} = \mathbf{Z}^{(L)}_{\mathcal{M}_i}$ hold the constant-curvature component embedding $\boldsymbol{z}_{\mathcal{M}_i}$ of component space $\mathcal{M}_i$.

Theorem 1 (κ-left-matrix-multiplication as attentional aggregation). *Let rows of $\mathbf{H}$ hold the encoding $\boldsymbol{z}_{\mathcal{M}_i}$, linear transformed by $\mathbf{W}$, and $\mathbf{A}$ hold the attentional weights, the κ-left-matrix-multiplication $\mathbf{A}\boxtimes_\kappa\mathbf{H}$ performs the attentional aggregation over the rows of $\mathbf{H}$, i.e., $(\mathbf{A}\boxtimes_\kappa\mathbf{H})_{p\bullet}$ is the linear combination of $\mathbf{H}_{p\bullet}$ with respect to attentional weight $\mathbf{A}_{pq}$, where q enumerates the node index in set Ψ, $\Psi = p\cup\mathcal{N}_p$ and $\mathcal{N}_p$ is the neighborhood node index of v_p.*

Proof. Please refer to the Supplementary Material. □

Inter-Component Attentional Layer: This is the output layer of the proposed mixed-curvature GNN. In this layer, we perform attentional concatenation to fuse constant-curvature representations across component spaces so as to learn mixed-curvature representations in the product space.

The importance of constant-curvature component space is usually different in constructing the mixed-curvature space. Thus, we introduce the inter-component attention to learn the importance of component space. Specifically, we first lift component encodings to the common tangent space, and figure out their centorid by the mean pooling as follows:

$$\boldsymbol{\mu} = \text{Pooling}\left(\mathbf{log}^\kappa_{\mathbf{0}}(\mathbf{W}_i\otimes_\kappa\boldsymbol{z}_{\mathcal{M}_i})\right), \qquad (8)$$

where we construct the common tangent space by the linear transformation $\mathbf{W}_i$ and $\mathbf{log}^\kappa_{\mathbf{0}}$. Then, we model the importance of a component by the position of the component embedding relative to the centorid, parameterized by $\boldsymbol{\theta}_{\text{inter}}$,

$$att_{\text{inter}}(\boldsymbol{z}_{\mathcal{M}_i},\boldsymbol{\mu}) = \sigma\left(\boldsymbol{\theta}_{\text{inter}}{}^\top\left(\mathbf{log}^\kappa_{\mathbf{0}}(\mathbf{W}_i\otimes_\kappa\boldsymbol{z}_{\mathcal{M}_i})-\boldsymbol{\mu}\right)\right), \qquad (9)$$

Next, we compute the attention weight of each component via the softmax function as follows:

$$w_i = \frac{\exp(att_{\text{inter}}(\boldsymbol{z}_{\mathcal{M}_i},\boldsymbol{\mu}))}{\sum_{k=1}^K \exp(att_{\text{inter}}(\boldsymbol{z}_{\mathcal{M}_k},\boldsymbol{\mu}))}. \qquad (10)$$

Finally, with the learnable attentional weights, we perform attentional concatenation and have the output representation, $\boldsymbol{z} = \vert\vert^K_{i=1}(\boldsymbol{w}_i\otimes_\kappa\boldsymbol{z}_{\mathcal{M}_i})$. Note that, learning representations in the mixed-curvature space not only matches the complex structures of graphs, but also inherently provides the positive and negative samples of multiple Riemannian views for contrastive learning, which we will discuss in the next part.

Dual Contrastive Approach

With the combinatorial construction of the mixed-curvature space, we propose a novel *dual contrastive approach* of single-view and cross-view contrastive learning for the self-supervisd learning. To this end, we first design a Riemannian Projector to reveal the hyperbolic, spherical and Euclidean views with respect of the sign of curvature κ, and then design a Riemannian Discriminator $\mathcal{D}_\kappa$ to contrast the positive and negative samples. As shown in Fig. 1, we contrast the samples in the same Riemannian view (i.e., *single-view* *contrastive learning*) and concurrently contrast across different Riemannian views (i.e., *cross-view contrastive learning*). We summarize the self-supervised learning process of SelfMGNN with dual contrastive loss in Algorithm 1.

Riemannian Projector: We design the Riemannian projector to reveal different Riemannian views for contrastive learning. Recall that the mixed-curvature space $\mathcal{P}$ is a combinatorial construction of 3 canonical types of component spaces in essence, i.e., $\mathbb{H}$ ($\kappa < 0$), $\mathbb{S}$ ($\kappa > 0$) and $\mathbb{E}$ ($\kappa = 0$), where $\mathbb{H}$ and $\mathbb{S}$ can have multiple component spaces with distinct learnable curvatures. We can fuse component encodings of the same space type and obtain 3 canonical Riemannian views: hyperbolic $\mathbf{h} \in \mathbb{H}$, spherical $\mathbf{s} \in \mathbb{S}$ and Euclidean $\mathbf{e} \in \mathbb{E}$. To this end, we design a map, *RiemannianProjector*: $(\mathbf{Z})_{i\bullet} \to [\mathbf{h}_i \;\; \mathbf{e}_i \;\; \mathbf{s}_i]$, where $(\mathbf{Z})_{i\bullet}$ is the output of mixed-curvature GNN containing all component embeddings. Specifically, for each space type, we first project the component embedding $\boldsymbol{z}_{\mathcal{M}_i}$ to the corresponding space of standard curvature via MLP_κ layers defined as follows:

$$\mathrm{MLP}_\kappa(\mathbf{x}) = \sigma^{\otimes_\kappa}(\mathbf{b} \oplus_\kappa \mathbf{M} \otimes_\kappa \boldsymbol{z}_{\mathcal{M}_i}), \tag{11}$$

where $\mathbf{M}$ and $\mathbf{b}$ denote the weight matrix and bias, respectively. Then, we fuse the projected embeddings in account of the importance of component space via the $\mathbf{mid}_\kappa$ function:

$$\mathbf{v} = \mathbf{mid}_\kappa\left(\{\mathbf{W}_i \otimes_\kappa \boldsymbol{z}_{\mathcal{M}_i}\}_{i\in\Omega}; \{w_i\}_{i\in\Omega}\right), \tag{12}$$

where Ω is the component index set of the given type. $\mathbf{W}_i$ is the linear transformation. The importance weight $\boldsymbol{w}_i$ is learned by *inter-component attention*, and $\mathbf{v} \in \{\mathbf{h}, \mathbf{e}, \mathbf{s}\}$.

Riemannian Discriminator: Contrasting between positive and negative samples is fundamental for contrastive learning. However, it is challenging in the Riemannian space, and existing methods, to our knowledge, cannot be applied to Riemannian spaces due to the intrinsic difference in the geometry. To bridge this gap, we design a novel Riemannian Discriminator to scores the agreement between positive and negative samples. The main idea is that we lift the samples to the common tangent space, and evaluate the agreement score in the tangent space. Specifically, we utilize the bilinear form to evaluate the agreement. Given two Riemannian views $\mathbf{x}$ and $\mathbf{y}$ of a node, $\mathbf{x}, \mathbf{y} \in \{\mathbf{h}, \mathbf{e}, \mathbf{s}\}$, we give the formulation parameterized by the matrix $\mathbf{D}$ as follows:

$$\mathcal{D}_\kappa(\mathbf{x}, \mathbf{y}) = \left(\mathbf{log}_\mathbf{0}^{\kappa_\mathbf{x}}(\mathbf{x})\right)^\top \mathbf{D}\left(\mathbf{log}_\mathbf{0}^{\kappa_\mathbf{y}}(\mathbf{y})\right), \tag{13}$$

where we construct the common tangent space via $\mathbf{log}_\mathbf{0}^{\kappa_\mathbf{x}}$, and $\kappa_\mathbf{x}$ is the curvature of the corresponding view.

Single-view Contrastive Learning: SELFMGNN employs the single-view contrastive learning in each Riemannian view of the mixed-curvature space. Specifically, we first include a congruent augmentation $\mathbf{G}^\beta$ similar to **??**. Then, we introduce a contrastive discrimination task for a given Riemannian view: for a sample in $\mathbf{G}^\alpha$, we aims to discriminate the positive sample from negative samples in the congruent counterpart $\mathbf{G}^\beta$. Here, we use superscript α and β to distinguish notations of the graph and its congruent augmentation. We formulate the InfoNCE loss (**?**) as follows:

$$\mathcal{L}_S(\alpha, \beta) = -\log \frac{\exp \mathcal{D}_\kappa(\mathbf{x}_i^\alpha, \mathbf{x}_i^\beta)}{\sum_{j=1}^{|V|} \mathbb{I}\{i \neq j\} \exp \mathcal{D}_\kappa(\mathbf{x}_i^\alpha, \mathbf{x}_j^\beta)}, \tag{14}$$

Algorithm 1: Self-supervised Learning SELFMGNN

```
Input: Graph $G = (\mathbf{G}, \mathbf{X})$, weight $\gamma$, Congruent
       Graph Generation Function $\Gamma(\cdot)$
Output: MixedCurvatureGNN para., and throw away
        RiemannianProjector para.
while not converging do
    // Views of the original graph $\mathbf{G}^\alpha$:
    Set $\mathbf{G}^\alpha = \mathbf{G}$;
    $\mathbf{Z}^\alpha = MixedCurvatureGNN(\mathbf{G}^\alpha, \mathbf{X}; \boldsymbol{\theta}^\alpha)$;
    $[\mathbf{H}^\alpha \; \mathbf{E}^\alpha \; \mathbf{S}^\alpha] = RiemannianProjector(\mathbf{Z}^\alpha; \boldsymbol{\phi})$;
    // Views of the congruent augmentation $\mathbf{G}^\beta$:
    Generate a congruent graph $\mathbf{G}^\beta = \Gamma(\mathbf{G})$;
    $\mathbf{Z}^\beta = MixedCurvatureGNN(\mathbf{G}^\beta, \mathbf{X}; \boldsymbol{\theta}^\beta)$;
    $[\mathbf{H}^\beta \; \mathbf{E}^\beta \; \mathbf{S}^\beta] = RiemannianProjector(\mathbf{Z}^\beta; \boldsymbol{\phi})$;
    // Dual contrastive loss:
    for each node $v_i$ in $\mathbf{G}^\alpha$ and $v_j$ to $\mathbf{G}^\beta$ do
        for Riemannian views $\mathbf{x}, \mathbf{y} \in \{\mathbf{h}, \mathbf{e}, \mathbf{s}\}$ do
            Single-view contrastive learning with
              Eqs. (14) and (15);
            Cross-view contrastive learning with Eqs.
              (16) and (17);
    // Update neural network parameters:
    Compute gradients of the dual contrastive loss:
        $\nabla_{\boldsymbol{\theta}^\alpha, \boldsymbol{\theta}^\beta, \boldsymbol{\phi}, \mathbf{D}_S, \mathbf{D}_C} \; \mathcal{J}_S + \lambda \mathcal{J}_C$.
```

where $\mathbf{x}_i^\beta$ and $\mathbf{x}_j^\beta$ are the positive sample and negative samples of v_i in $\mathbf{G}^\alpha$, respectively. $\mathbb{I}\{\cdot\} \in \{0, 1\}$ is an indicator function who will return 1 iff the condition $(\cdot)$ is true ($i \neq j$ in this case). We utilize the Riemannian discriminator $\mathcal{D}_\kappa(\cdot, \cdot)$ to evaluate the agreement between the samples.

In the single-view contrastive learning, for each Riemannian view, we contrast between $\mathbf{G}^\alpha$ and its congruent augmentation $\mathbf{G}^\beta$, and vice versa. Formally, we have the single-view contrastive loss as follows:

$$\mathcal{J}_S = \sum\nolimits_{i=1}^{|V|} \sum\nolimits_{\mathbf{x}\in\{\mathbf{h},\mathbf{e},\mathbf{s}\}} \left(\mathcal{L}_S(\alpha, \beta) + \mathcal{L}_S(\beta, \alpha)\right). \tag{15}$$

Cross-view Contrastive Learning: SELFMGNN further employs a novel cross-view contrastive learning. The novelty lies in that our design in essence enjoys the multi-view nature of the mixed-curvature space, i.e., we exploit the multiple Riemannian views of the mixed-curvature space, and contrast across different views. Specifically, we formulate the contrastive discrimination task as follows: for a given Riemannian view of $\mathbf{G}^\alpha$, we aim to discriminate the given view from the other Riemannian views of $\mathbf{G}^\beta$. We formulate the InfoNCE loss as follows:

$$\mathcal{L}_C(\alpha, \beta) = -\log \frac{\exp \mathcal{D}_\kappa(\mathbf{x}_i^\alpha, \mathbf{x}_i^\beta)}{\sum_\mathbf{x} \mathbb{I}\{\mathbf{x} \neq \mathbf{y}\} \exp \mathcal{D}_\kappa(\mathbf{x}_i^\alpha, \mathbf{y}_i^\beta)}, \tag{16}$$

where $\mathbb{I}\{\mathbf{x} \neq \mathbf{y}\}$ is to select the embeddings of different Riemannian views. Similarly, we contrast between $\mathbf{G}^\alpha$ and $\mathbf{G}^\beta$ and vice versa, and have the cross-view contrastive loss:

$$\mathcal{J}_C = \sum\nolimits_{i=1}^{|V|} \sum\nolimits_{\mathbf{x}\in\{\mathbf{h},\mathbf{e},\mathbf{s}\}} \left(\mathcal{L}_C(\alpha, \beta) + \mathcal{L}_C(\beta, \alpha)\right). \tag{17}$$

Table 2: The summary of AUC (%) for link prediction (LP) and classification accuracy (%) for node classification (NC) on Citeseer, Cora, Pubmed, Amazon and USA datasets. The highest scores are in **bold**, and the second blue.

		Citeseer		**Cora**		**Pubmed**		**Amazon**		**Airport**	
	Method	LP	NC	LP	NC	LP	NC	LP	NC	LP	NC
Euclidean	GCN	93.6(0.7)	70.2(0.8)	91.4(0.7)	81.3(0.3)	93.0(0.6)	78.8(0.2)	92.9(0.9)	71.2(1.1)	90.5(0.4)	50.8(0.9)
	GraphSage	87.2(0.9)	68.2(1.1)	88.7(0.6)	78.1(0.8)	87.7(0.4)	77.5(0.3)	91.8(0.5)	72.9(1.6)	85.6(1.1)	47.8(0.8)
	GAT	92.9(0.7)	72.0(0.7)	93.4(0.4)	82.1(0.7)	92.6(0.3)	77.1(0.7)	93.9(0.6)	72.6(0.8)	91.4(0.6)	49.3(0.7)
	DGI	92.7(0.5)	71.3(0.7)	91.8(0.5)	81.4(0.6)	92.8(0.7)	76.6(0.6)	93.5(0.4)	72.2(0.3)	92.5(0.8)	50.1(0.5)
	MVGRL	94.8(0.3)	72.1(0.8)	93.2(0.7)	82.7(0.7)	95.9(0.2)	78.9(0.3)	96.2(0.5)	74.0(1.0)	95.1(0.3)	52.1(1.0)
	GMI	95.0(0.6)	72.5(0.3)	93.9(0.3)	81.8(0.2)	96.5(0.8)	79.0(0.2)	96.8(0.7)	74.5(0.9)	94.7(0.5)	51.9(0.7)
Riemannian	HGCN	94.6(0.4)	71.7(0.5)	93.2(0.1)	81.5(0.6)	96.2(0.2)	78.5(0.4)	96.7(0.9)	75.2(1.3)	93.6(0.3)	51.2(0.6)
	HAT	93.7(0.5)	72.2(0.6)	93.0(0.5)	83.1(0.7)	96.3(0.3)	78.6(0.7)	96.9(1.1)	74.1(1.0)	93.9(0.6)	51.3(1.0)
	LGCN	95.5(0.5)	72.1(0.7)	93.7(0.5)	83.3(0.9)	96.6(0.2)	78.6(1.0)	**97.5**(0.9)	75.1(1.1)	96.4(0.2)	52.0(0.9)
	κ-GCN	93.8(0.7)	71.2(0.5)	92.8(0.8)	81.6(0.7)	95.0(0.3)	78.7(0.6)	94.8(0.6)	72.4(1.5)	93.5(0.7)	50.9(1.2)
	SELFMGNN	**96.9**(0.3)	**73.1**(0.9)	**94.6**(0.6)	**83.8**(0.8)	**97.3**(0.2)	**79.6**(0.5)	**97.5**(1.0)	**75.3**(0.8)	**96.9**(0.5)	**52.7**(0.7)

Dual Contrastive Loss: In SelfMGNN, we integrate the single-view and cross-view contrastive learning, and formulate the dual contrastive loss as follows:

$$\mathcal{J}_{self} = \mathcal{J}_S + \gamma \mathcal{J}_C, \tag{18}$$

where γ is the balance weight. The benefit of dual contrastive loss is that we can contrast the samples in the same Riemannian view (single-view) and contrast across different Riemannian views (cross-view), comprehensively leveraging the rich information in the mixed-curvature space to encode the graph structure. Finally, SelfMGNN learns representations in the mixed-curvature Riemannian space capturing the complex structures of graphs without labels.

Experiments

In this section, we evaluate SELFMGNN with the link prediction and node classification tasks against 10 strong baselines on 5 benchmark datasets. We report the mean with the standard deviations of 10 independent runs for each model to achieve fair comparisons.

Experimental Setups

Datasets: We utilize 5 benchmark datasets, i.e., the widely-used **Citeseer**, **Cora**, and **Pubmed** (**??**), and the latest **Amazon** and **Airport** (**?**).

Euclidean Baselines: i) *Supervised Models*: **GCN** (**?**), **GraphSage** (**?**), **GAT** (**?**). ii) *Self-supervised Models*: **DGI** (**?**), **MVGRL** (**?**), **GMI** (**?**).

Riemannian Baselines: i) *Supervised Models*: **HGCN** (**?**), **HAT** (**?**) and **LGCN** (**?**) for hyperbolic space; **κ-GCN** (**?**) with positive κ for spherical space. ii) *Self-supervised Models*: There is no self-supervised Riemannian models in the literature, and thus we propose SELFMGNN to fill this gap.

Implementation Details

Congruent graph: As suggested by **?**, we opt for the diffusion to generate a congruent augmentation. Specifically, given an adjacency matrix $\mathbf{G}^\alpha$, we use the congruent graph generation function $\Gamma(\cdot)$ to obtain a diffusion matrix $\mathbf{G}^\beta$ and treat it as the adjacency matrix of the congruent augmentation. The diffusion is computed once via fast approximated and sparsified method (**?**).

Signature: The mixed-curvature space is parameterized by the signature, i.e., space type, curvature and dimensionality of the component spaces. The space type of component $\mathcal{M}_i$ can be hyperbolic $\mathbb{H}$, spherical $\mathbb{S}$ or Euclidean $\mathbb{E}$, and we utilize the combination of them to cover the mixed and complicated graph structures. The dimensionality $d_{\mathcal{M}_i}$ is a hyperparameter. The curvature $\kappa_{\mathcal{M}_i}$ is a learnable parameters as our loss is differentiable with respect to the curvature.

Learning manner: Similar to **?**, self-supervised models first learn representations without labels, and then were evaluated by specific learning task, which is performed by directly using these representations to train and test for learning tasks. Supervised models were trained and tested by following **?**. Please refer to the Supplementary Material for further experimental details.

Table 3: Ablation study of SELFMGNN for node classification task in classification accuracy (%).

	Variants	Citesser	Core	Pubmed
CCS	$\mathbb{H}^{24}$	72.2(0.7)	82.1(0.4)	78.6(0.3)
	$\mathbb{S}^{24}$	70.5(0.8)	82.3(0.5)	77.5(0.4)
	$\mathbb{E}^{24}$	71.8(1.1)	81.0(0.7)	77.3(0.8)
Single	$\mathbb{H}^8 \times \mathbb{S}^8 \times \mathbb{E}^8$	72.6(0.3)	82.7(0.8)	78.9(0.9)
	$(\mathbb{H}^4)^2 \times (\mathbb{S}^4)^2 \times \mathbb{E}^8$	72.8(0.6)	83.1(0.6)	79.2(0.2)
	$(\mathbb{H}^2)^4 \times (\mathbb{S}^2)^4 \times \mathbb{E}^8$	72.9(0.2)	83.5(0.5)	79.3(0.5)
Ours	$\mathbb{H}^8 \times \mathbb{S}^8 \times \mathbb{E}^8$	72.8(1.0)	83.3(0.9)	79.2(0.6)
	$(\mathbb{H}^4)^2 \times (\mathbb{S}^4)^2 \times \mathbb{E}^8$	73.1(0.9)	83.8(0.5)	79.6(0.7)
	$(\mathbb{H}^2)^4 \times (\mathbb{S}^2)^4 \times \mathbb{E}^8$	**73.3**(0.5)	**84.1**(0.8)	**79.9**(1.1)

Link Prediction

For link perdition, we utilize the Fermi-Dirac decoder with distance function to define the probability based on model outputs $\boldsymbol{z}$. Formally, we have the probability as follows:

$$p((i,j) \in E|\boldsymbol{z}_i, \boldsymbol{z}_j) = \left(\exp\left((d_{\mathcal{M}}(\boldsymbol{z}_i, \boldsymbol{z}_j)^2 - r)/t\right) + 1\right)^{-1}, \tag{19}$$

Table 4: Learning results of the mixed-curvature space on the datasets — curvature (weight) of each component space.

Dataset	$\mathbb{H}^4$	$\mathbb{H}^4$	$\mathbb{S}^4$	$\mathbb{S}^4$	$\mathbb{E}^8$
Citeseer	$-0.67(0.29)$	$-0.58(0.19)$	$+0.82(0.21)$	$+2.72(0.13)$	$0(0.18)$
Cora	$-0.90(0.18)$	$-1.31(0.25)$	$+0.76(0.28)$	$+0.19(0.08)$	$0(0.21)$
Pubmed	$-1.12(0.26)$	$-0.79(0.34)$	$+0.59(0.16)$	$+1.05(0.15)$	$0(0.09)$
Amazon	$-0.78(0.11)$	$-1.02(0.48)$	$+1.13(0.05)$	$+2.24(0.24)$	$0(0.12)$
Airport	$-1.26(0.30)$	$-2.15(0.17)$	$+1.85(0.20)$	$+0.67(0.18)$	$0(0.15)$

where r, t are hyperparameters. For each method, $d_{\mathcal{M}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ is the distance function of corresponding representation space, e.g., $||\boldsymbol{z}_i - \boldsymbol{z}_j||_2$ for Euclidean models, and we have

$$d_{\mathcal{P}}(\boldsymbol{z}_i, \boldsymbol{z}_j)^2 = \sum_{l=1}^{K} d_{\mathcal{M}_l}^{\kappa_l}\left((\boldsymbol{z}_i)_{\mathcal{M}_l}, (\boldsymbol{z}_j)_{\mathcal{M}_l}\right)^2, \qquad (20)$$

for SELFMGNN. We utilize AUC as the evaluation metric and summarize the performance in Table 2. We set output dimensionality to be 24 for all models for fair comparisons. Table 2 shows that SELFMGNN outperforms the self-supervised models in Euclidean space consistently since it better matches the mixed structures of graphs with the mixed-curvature space. SELFMGNN achieves competitive and even better results with the supervised Riemannian baselines. The reason lies in that we leverage dual contrastive approach to exploit the rich information of data themselves in the mixed-curvature Riemannian space.

Node Classification

For node classification, we first discuss the classifier as none of existing classifiers, to our knowledge, can work with mixed-curvature spaces. To bridge this gap, inspired by (**?**), for Riemannian models, we introduce the Euclidean transformation to generate an encoding, summarizing the structure of node representations. Specifically, we first introduce a set of centroids $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_C\}$, where $\boldsymbol{\mu}_c$ is the centroid in Riemannian space learned jointly with the learning model. Then, for output representation $\boldsymbol{z}_j \in \mathcal{M}$, its encoding is defined as $\boldsymbol{\xi} = (\xi_{1j}, \ldots, \xi_{Cj})$, where $\xi_{ij} = d_{\mathcal{M}}(\boldsymbol{\mu}_i, \boldsymbol{z}_j)$, summarizing the position of $\boldsymbol{z}_i$ relative to the centroids. Now, we are ready to use logistic regression for node classification and the likelihood is

$$p(y|\boldsymbol{h}) = \text{sigmoid}(\mathbf{w}_C^\top \boldsymbol{h}), \qquad (21)$$

where $\mathbf{w}_C \in \mathbb{R}^{|C|}$ is the weight matrix, and y is the label. $\boldsymbol{h} = \boldsymbol{\xi}$ for Riemannian models and $\boldsymbol{h}$ is the output of Euclidean ones. We utilize classification accuracy (**?**) as the evaluation metric and summarize the performance in Table 2. SELFMGNN achieves the best results on all the datasets.

Ablation Study

We give the ablation study on the importance of i) mixed-curvature space (MCS) and ii) cross-view contrastive learning. To this end, we include two kinds of variants: CCS and Single, the degenerated SELFMGNNs without some functional module. CCS variants work without Cartesian product, and thus are learned by the single-view contrastive loss in a constant-curvature space. e.g., $\mathbb{H}^{24}$ is the variant in the hyperbolic space, where the superscript is the dimensionality. Single variants work with the mixed-curvature space, but disable the cross-view contrastive learning. The ours are the proposed SELFMGNNs. A specific instantiation is denoted by Cartesian product, e.g., we use $(\mathbb{H}^4)^2 \times (\mathbb{S}^4)^2 \times \mathbb{E}^8$ as default, whose mixed-curvature space is constructed by Cartesian product of 2 $\mathbb{H}^4$, 2 $\mathbb{S}^4$ and 1 $\mathbb{E}^8$ component spaces.

We show the classification accuracy of these variants in Table 3, and we find that: i) Mixed-curvature variant with single or dual contrastive learning outperforms its CCS counterpart. The reason lies in that the mixed-curvature space is more flexible than a constant-curvature space to cover the complicated graph structures. ii) Disabling cross-view contrastive learning decreases the performance as the cross-view contrasting further unleashes the rich information of data in the mixed-curvature space. iii) Allowing more than one hyperbolic and spherical spaces can also improve performance as $(\mathbb{H}^4)^2 \times (\mathbb{S}^4)^2 \times \mathbb{E}^8$ and $(\mathbb{H}^2)^4 \times (\mathbb{S}^2)^4 \times \mathbb{E}^8$ both outperform $\mathbb{H}^8 \times \mathbb{S}^8 \times \mathbb{E}^8$.

Furthermore, we discuss the curvatures of the datasets. We report the learned curvatures and weights of each component space for the real-world datasets in Table 4. As shown in Table 4, component spaces of the same space type are learned with different curvatures, showing that the curvatures over different hierarchical or cyclical regions can still be different. $(\mathbb{H}^4)^2 \times (\mathbb{S}^4)^2 \times \mathbb{E}^8$ has 2 component spaces for hyperbolic (spherical) geometry, and allowing multiple hyperbolic (spherical) components enables us to cover a wider range of curvatures of the graph, better matching the graph structures. This also explains why $(\mathbb{H}^4)^2 \times (\mathbb{S}^4)^2 \times \mathbb{E}^8$ outperforms $\mathbb{H}^8 \times \mathbb{S}^8 \times \mathbb{E}^8$ in Table 3. With learnable curvatures and weights of each component, SELFMGNN matches the mixed and complicated graph structures, and learns more promising graph representations.

Related Work

SELFMGNN learns representations via a mixed-curvature graph neural network with the dual contrastive approach. Here, we briefly discuss the related work on the *graph neural network* and *contrastive learning*:

Graph Neural Network

Graph Neural Networks (GNNs) achieve the state-of-the-art results in graph analysis tasks (**??**). Recently, a few attempts propose to marry GNN and Riemannian representation learning (**???**) as graphs are non-Euclidean inherently (**?**). In hyperbolic space, **??** introduce shallow models, while **???** formulate deep models (GNNs). Furthermore, **?** generalize hyperbolic GNN to dynamic graphs with insights in temporal analysis. **?** study the curvature exploration. **??** propose promising methods to apply the hyperbolic geometry to network alignment.

Beyond hyperbolic space, **?** studies the matrix manifold of Riemannian spaces, and **?** generalizes GCN to arbitrary constant-curvature spaces. To generalize representation learning, **?** propose to learn in the mixed-curvature space. **?** introduce the mixed-curvature VAE. Recently, **?** model the knowledge graph triples in mixed-curvature space specifically with the supervised manner. Distinguishing from these studies, we propose the first self-supervised mixed-curvature model, allowing multiple hyperbolic (spherical) subspaces each with distinct curvatures.

Contrastive Learning

Contrastive Learning is an attractive self-supervised learning method that learns representations by contrasting positive and negative pairs (**?**). Here, we discuss the contrastive learning methods on graphs. Specifically, **?** contrast patch-summary pairs via infomax theory. **?** leverage multiple views for contrastive learning. **?** formulate a general framework for pre-training. **?** incorporate the generative learning concurrently. **?** explore the graph-specific infomax for contrastive learning. **?** aim to learn graph level representations. **??** consider the heterogeneous graphs. To the best of our knowledge, existing methods cannot apply to Riemannian spaces and we bridge this gap in this paper.

Conclusion

In this paper, we take the first attempt to study the self-supervised graph representation learning in the mixed-curvature Riemannian space, and present a novel SELFMGNN. Specifically, we first construct the mixed-curvature space via Cartesian product of Riemannian manifolds and design hierarchical attention mechanisms within and among component spaces to learn graph representations in the mixed-curvature space. Then, we introduce the single-view and cross-view contrastive learning to learn graph representations without labels. Extensive experiments show the superiority of SELFMGNN.

Acknowledgments

Hao Peng is partially supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grant 62002007, S&T Program of Hebei through grant 21340301D. Sen Su and Zhongbao Zhang are partially supported by the National Key Research and Development Program of China under Grant 2018YFB1003804, NSFC under Grant U1936103 and 61921003. Philip S. Yu is partially supported by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941. Jiawei Zhang is partially supported by NSF through grants IIS-1763365, IIS-2106972 and by UC Davis. This work was also sponsored by CAAI-Huawei MindSpore Open Fund. Thanks for computing infrastructure provided by Huawei MindSpore platform.

Technical Appendix

In the technical appendix, we provide further details for the proof and derivations as well as the experiments.

A. On the Attentional Aggregation

In this section, we first start with attentional aggregation in the Euclidean space, and then elaborate on the necessary operations in the κ-stereographic model $\mathcal{M}$ and generalize the attentional aggregation in the Euclidean space to the κ-stereographic model. Finally, we prove the *Theorem 1 (κ-left-matrix-multiplication as aggregation)*.

Attentional aggregation in the Euclidean space

In this part, we show that attentional aggregation is a linear combination of the feature vectors with the learnable weights and left-matrix-multiplication essentially performs the attentional aggregation.

Given a target node i and its neighbor node set $\mathcal{N}_i$, the encoding of the target $\boldsymbol{z}_i$ is updated as $\sum_j A_{ij}\boldsymbol{z}_j$, where $j \in \Omega$ and $\Omega = \mathcal{N}_i \cup i$. We add a self-loop in particular to keep the information of the node itself, and A_{ij} denotes the attentional weight to be learned (**??**). Obviously, the attentional aggregation can be rewritten as a linear combination $\boldsymbol{L}(\cdot,\cdot)$. Specifically, the linear combination is defined as $\boldsymbol{L}(c(A_{ij}), \boldsymbol{z}_j) = \sum_j c(A_{ij})\boldsymbol{z}_j$, where $c(A_{ij})$ is the coefficient function to output a real coefficient scaling the corresponding $\boldsymbol{z}_j$. We have $c(A_{ij}) = A_{ij}$ for the attentional aggregation above.

Let us consider left-matrix-multiplication of $\mathbf{Z} \in \mathbb{R}^{N\times D}$ by $\mathbf{A} \in \mathbb{R}^{N\times N}$, i.e., we have $\mathbf{AZ} \in \mathbb{R}^{N\times D}$. The i^{th} row of $(\mathbf{AZ})$ is given as follows:

$$A_{i1}\boldsymbol{z}_1 + A_{i2}\boldsymbol{z}_2 + \cdots + A_{ij}\boldsymbol{z}_j + \cdots + A_{iN}\boldsymbol{z}_N, \qquad (22)$$

where $\boldsymbol{z}_j$ is the j^{th} row of $\mathbf{Z}$. In fact, the row-wise left-matrix-multiplication is given by the linear combination $(\mathbf{AZ})_{i\bullet} = \boldsymbol{L}(c(A_{ij}), \boldsymbol{z}_j)$ with $\mathbf{A}$, performing the attentional aggregation. Thus, we can give the lemma as follows:

Lemma 1 (Left-matrix-multiplication as attentional aggregation). *Given $\mathbf{Z} \in \mathbb{R}^{N\times D}$ holding encoding vectors in its row and weights $\mathbf{A} \in \mathbb{R}^{N\times N}$, the left-matrix-multiplication of $\mathbf{Z}$ by $\mathbf{A}$ updates encoding vectors in $\mathbf{Z}$ with attentional aggregation.*

Operations in the κ-stereographic model

In this part, we first review the notion of *midpoint* in the κ-stereographic model, and then generalize the Euclidean left-matrix-multiplication to the *κ-left-matrix-multiplication* in the κ-stereographic model with the *midpoint*.

The midpoint in the Euclidean space is intuitive, however, it is nontrivial in the κ-stereographic model as the manifold is curved. We give the definition of the (weighted) midpoint in the $\boldsymbol{\kappa}$-stereographic model as follows:

Definition 1 (Midpoint in the $\boldsymbol{\kappa}$-stereographic model). *Given a set of κ-stereographic vectors $\{\mathbf{x}_i\}_{i=1}^n$, and weights $\boldsymbol{\alpha} \in \mathbb{R}^n$, the weighted midpoint in the κ-stereographic model is calculated via $\mathbf{mid}_\kappa\left(\{\mathbf{x}_i\}_{i=1}^n; \boldsymbol{\alpha}\right)$ as follows:*

$$\mathbf{mid}_\kappa\left(\{\mathbf{x}_i\}_{i=1}^n; \boldsymbol{\alpha}\right) = \frac{1}{2} \otimes_\kappa \left(\sum_{i=1}^n \frac{\alpha_i \lambda_{\mathbf{x}_i}^\kappa}{\sum_{j=1}^n \alpha_j (\lambda_{\mathbf{x}_j}^\kappa - 1)} \mathbf{x}_i \right), \qquad (23)$$

where $\lambda_{\mathbf{x}_j}^\kappa = 4\left(1 + \kappa||\mathbf{x}||_2^2\right)^{-2}$ is the conformal factor.

Note that, the midpoint in the κ-stereographic model is essentially a linear combination regulated with a $\kappa-$scaling.

With the geometry of the κ-stereographic model, we give the definition of κ-left-matrix-multiplication following **?**, the generalization of left-matrix-multiplication in the Euclidean space, below:

Definition 2 (κ-left-matrix-multiplication). *Given* $\mathbf{Z} \in \mathbb{R}^{N \times D}$ *holding* κ*-stereographic vectors in its row and weights* $\mathbf{A} \in \mathbb{R}^{N \times N}$*, the* κ*-left-matrix-multiplication of* $\mathbf{Z}$ *by* $\mathbf{A}$ *is defined as follows:*

$$(\mathbf{A} \boxtimes_\kappa \mathbf{Z})_{i\bullet} := A \otimes_\kappa \mathbf{mid}_\kappa \left(\{\mathbf{Z}_{i\bullet}\}_{i=1}^{n}; \mathbf{A}_{i\bullet}\right), \qquad (24)$$

where $A = \sum_j \mathbf{A}_{ij}$*,* $\mathbf{mid}_\kappa$ *denotes midpoint in the* κ*-stereographic model.*

Attentional Aggregation in κ-stereographic Model

In this part, we show that the κ-left-matrix-multiplication performs the attentional aggregation in κ-stereographic model. In other words, we prove **Theorem 1** in the subsection of *attentional aggregation layer* in our paper.

With the formal definition of the linear combination, we rewrite the **Theorem 1** equivalently as follows:

Theorem 1 (κ-left-matrix-multiplication as attentional aggregation). *Let rows of* $\mathbf{H}$ *hold the encoding* $\boldsymbol{z}_{\mathcal{M}_i}$*, (linear transformed by* $\mathbf{W}$*) and* $\mathbf{A}$ *hold the attentional weights, the* κ*-left-matrix-multiplication* $\mathbf{A} \boxtimes_\kappa \mathbf{H}$ *performs the attentional aggregation over the rows of* $\mathbf{H}$*, i.e.,* $\mathbf{A} \boxtimes_\kappa \mathbf{H}$ *is the row-wise linear combination of* $\mathbf{H}$ *with respect to attentional weight* $\mathbf{A}_{ij}$*:*

$$(\mathbf{A} \boxtimes_\kappa \mathbf{H})_{i\bullet} = \boldsymbol{L}(c_{stereo}(\mathbf{A}_{ij}), \mathbf{h}_j), \qquad (25)$$

where $\mathbf{h}_j$ *is the* j^{th} *row of* $\mathbf{H}$*,* j *enumerates the index set* Ψ*,* $\Psi = i \cup \mathcal{N}_i$ *and* $\mathcal{N}_i$ *is the neighbors of* i *on the graph.* $c_{stereo}(\cdot)$ *is the function to output the coefficient in the* κ*-stereographic model.*

Proof. Recall the design of the (intra-component) attentions in our paper: $\mathbf{A}$ is given as $\hat{\mathbf{A}} + \mathbf{I}$, where $\hat{\mathbf{A}}$ is filled with softmax values in its row, and $\mathbf{I}$ is the identity matrix to keep the initial information of the node itself. That is, we have the row sum, $A = 2$. Then, with the definitions of *κ-left-matrix-multiplication* and *midpoint in the κ-stereographic model*, we give the derivation as follows:

$$\begin{aligned} &(\mathbf{A} \boxtimes_\kappa \mathbf{H})_{i\bullet} \\ =&A \otimes_\kappa \mathbf{mid}_\kappa \left(\{\mathbf{h}_j\}_{j=1}^{n}; \{\mathbf{A}_{ij}\}_{j=1}^{n}\right) \\ =&2 \otimes_\kappa \frac{1}{2} \otimes_\kappa \left(\sum_{j=1}^{n} \frac{\mathbf{A}_{ij} \lambda_{\mathbf{h}_j}^{\kappa}}{\sum_{l=1}^{n} \mathbf{A}_{il}(\lambda_{\mathbf{h}_l}^{\kappa} - 1)} \mathbf{h}_j \right) \\ =&\sum_{j=1}^{n} \frac{\mathbf{A}_{ij} \lambda_{\mathbf{h}_j}^{\kappa}}{\sum_{l=1}^{n} \mathbf{A}_{il}(\lambda_{\mathbf{h}_l}^{\kappa} - 1)} \mathbf{h}_j \\ =&\boldsymbol{L}(c_{\text{stereo}}(j), \mathbf{h}_j), \end{aligned} \qquad (26)$$

where the coefficient function is given as $c_{\text{stereo}}(j) = \frac{1}{C}\mathbf{A}_{ij}\lambda_{\mathbf{h}_j}^{\kappa}$, and $C = \sum_{l=1}^{n} \mathbf{A}_{il}(\lambda_{\mathbf{h}_l}^{\kappa} - 1)$. As shown above, with the well-designed attention mechanism, we eliminate the κ-scaling and make the κ-left-matrix-multiplication to be the linear combination with respect to the learnable attentions, performing the attentional aggregation. □

Table 2: The statistics of the datasets.

Dataset	#(Node)	#(Links)	#(Labels)
Citeseer	$3,327$	$4,732$	6
Cora	$2,708$	$5,429$	7
Pubmed	$19,717$	$44,338$	3
Amazon	$13,381$	$245,778$	10
Airport	$1,190$	$13,599$	4

B. Experimental Details

In this section, we give further experimental details, including data & code and implementation notes, in order to enhance the *reproducibility*.

Data and Code

Data The datasets used in this paper are publicly available, i.e., Citeseer, Cora, Pubmed, Amazon and Airport. We briefly describe these datasets as follows:

- Citeseer , Cora and Pubmed are the widely used citation networks, where nodes represent papers, and edges represent citations between them.
- The Amazon is a co-purchase graph, where nodes represent goods and edges indicate that two goods are frequently bought together.
- The Airport is an air-traffic graph, where nodes represent airports and edges indicate the traffic connection between them.

We list the statistics of the datasets in Table 2.

Code We submit the source code of an instance implementation of SELFMGNN in a ZIP named *Code*, and will publish the source code after acceptance.

Implementation Notes

In SELFMGNN, we stack the attentive aggregation layer twice to learn the component embedding. We employ a two-layer MLP$_\kappa$ in the Riemannian projector to reveal the Riemannian views for the self-supervised learning. In the experiments, we set the weight γ to be 1, i.e., the single-view and cross-view contrastive learning are considered to have the same importance. The grid search is performed over the learning rate in $[0.001, 0.003, 0.005, 0.008, 0.01]$ as well as the dropout probability in $[0, 0.8]$ with the step size of 0.1.

For all the comparison model, we perform a hyperparameter search on a validation set to obtain the best results, and the κ-GCN is trained with positive curvature in particular to evaluate the representation ability of the spherical space. We set the dimensionality to be 24 for all the models for the fair comparison. Note that, in SELFMGNN, the component space can be set to arbitrary dimensionality, whose curvature and importance are learned from the data, and thereby we construct a mixed-curvature space of any dimensionality, matching the curvatures of any datasets.

Fugit commodi dicta, vitae fuga ipsa cumque rem vero sequi illum, qui deleniti inventore dolore ut blanditiis fugiat aliquid consectetur magni, deserunt cum soluta numquam odit?Accusantium quisquam voluptas excepturi consequuntur ducimus dolore quidem error quaerat facere, omnis tempora corporis voluptatibus aliquam