Neural Networks in a Product of Hyperbolic Spaces

Jun Takeuchi[1], **Noriki Nishida**[2], **and Hideki Nakayama**[1]

[1,3]The University of Tokyo
[2]RIKEN
{takeuchi, nakayama}@nlab.ci.i.u-tokyo.ac.jp
noriki.nishida@riken.jp

Abstract

Machine learning in hyperbolic spaces has attracted much attention in natural language processing and many other fields. In particular, Hyperbolic Neural Networks (HNNs) have improved a wide variety of tasks, from machine translation to knowledge graph embedding. Although some studies have reported the effectiveness of embedding into the product of multiple hyperbolic spaces, HNNs have mainly been constructed in a single hyperbolic space, and their extension to product spaces has not been sufficiently studied. Therefore, we propose a novel method to extend a given HNN in a single space to a product of hyperbolic spaces. We apply our method to Hyperbolic Graph Convolutional Networks (HGCNs), extending several HNNs. Our model improved the graph node classification accuracy especially on datasets with tree-like structures. The results suggest that neural networks in a product of hyperbolic spaces can be more effective than in a single space in representing structural data.

1 Introduction

Machine learning that utilizes the properties of non-euclidean spaces has attracted much attention in recent years (Bronstein et al., 2017). In representation learning on natural language and graphs, where hierarchical data appear, hyperbolic spaces have recently been shown to be effective. In natural language processing (NLP), hyperbolic spaces have been applied to a variety of tasks such as word embedding (Nickel and Kiela, 2017; Tifrea et al., 2018), document embedding (Zhu et al., 2020b), natural language inference (Ganea et al., 2018), and machine translation (Gulcehre et al., 2018; Shimizu et al., 2021). Hyperbolic spaces have also been shown to be effective in graph embedding (Chamberlain et al., 2017; Sala et al., 2018; Chami et al., 2019), which is helpful for NLP models to utilize external knowledge graphs (Chami et al., 2020).

Recent progress in the use of hyperbolic space is supported by the development of hyperbolic neural networks (HNNs) (Tifrea et al., 2018), which consist of components such as linear and attention layers that are appropriately extended to hyperbolic spaces. How to define linear operations in hyperbolic space is non-trivial, and several different methods have been proposed (Shimizu et al., 2021; Chen et al., 2021).

Unlike Euclidean spaces, a product space of non-Euclidean spaces is geometrically different from a single space of the same dimension, and some studies have reported that using the product of small hyperbolic spaces improves the performance in graph and word embedding (Tifrea et al., 2018; Gu et al., 2019). In addition, Shimizu et al. observed that the superiority of their hyperbolic machine translation model over the Euclidean counterpart is lost as the dimensionality of word features increases, and they proposed using a product of multiple small hyperbolic spaces as a possible solution. However, existing HNN frameworks are defined in a single hyperbolic space, and how to extend HNNs to product spaces is still an open question.

Therefore, this paper proposes a novel method to extend a given HNN in a single space to a product of hyperbolic spaces. More specifically, we construct a general method to extend a hyperbolic matrix-vector multiplication, a major factor that distinguishes HNN variants, to a product space.

We apply our method to Hyperbolic Graph Convolutional Networks (HGCNs) (Chami et al., 2019) and show that our method outperforms the baselines especially on datasets with tree-like structures, suggesting that neural networks in a product of hyperbolic spaces are more effective for representing structural data than neural networks in a single hyperbolic space.

Contribution to NLP community

Single-space HNNs have already been applied to a variety of NLP tasks, and our method is applicable to most of them, with the potential to improve performance. As a start, we extended HNN++ (Shimizu et al., 2021), a recently proposed HNN variant, and applied it to machine translation tasks. The preliminary experimental results show that our method performs better, at least on small datasets. We plan to conduct experiments on large datasets soon.

2 Preliminaries

2.1 Riemannian Geometry

An n-dimensional manifold $\mathcal{M} = \mathcal{M}^n$ is an n-dimensional space locally approximated to an n-dimensional Euclidean tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at each point $\boldsymbol{x} \in \mathcal{M}^n$. A Riemannian manifold is a differentiable manifold with a metric tensor $\mathfrak{g}$. The exponential map $\exp_{\boldsymbol{x}} : T_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ and its inverse function $\log_{\boldsymbol{x}}$ are bijections defined locally around $\mathbf{0} \in T_{\boldsymbol{x}}\mathcal{M}$. For more details, please refer to Petersen et al. (2006).

2.2 Hyperbolic Space

A hyperbolic space $\mathbb{H} = \mathbb{H}^n_c$ is an n-dimensional Riemannian manifold with a constant negative curvature $-c$ $(c > 0)$. There are several equivalent models to represent a hyperbolic space. In the Poincaré Ball model $\mathbb{B}$, a hyperbolic space is represented as a ball of radius $\frac{1}{\sqrt{c}}$. Other models like the hyperboloid model and the equivalence between all models are detailed in Cannon et al. (1997).

2.3 Hyperbolic Neural Networks

Hyperbolic spaces have a structure similar to that of linear spaces called Gyrovector spaces (Ungar, 2008). The hyperbolic versions of addition and scalar multiplication are called Möbius Addition $\oplus$ [1] and Möbius Scalar Multiplication $\otimes$.

The matrix multiplication in hyperbolic space was proposed by Ganea et al. (2018). First, they showed that $\exp_0$ and $\log_0$ correspondence between hyperbolic space and its tangent space at origin are globally extended. Then Möbius Matrix Multiplication $\otimes$ between a matrix M and $\boldsymbol{x}$ is defined through tangent space approximation:

$$M \otimes_c \boldsymbol{x} := \exp_0(M \cdot \log_0(\boldsymbol{x})). \quad (1)$$

Hyperbolic version $\sigma^{\otimes_c}$ of any activation function σ in Euclidean space is defined in the same way:

$$\sigma^{\otimes_c}(\boldsymbol{x}) := \exp_0(\sigma(\log_0(\boldsymbol{x}))). \quad (2)$$

Shimizu et al. (2021) pointed out that the approximation using the tangent space at the origin (Eq. 1) produces distortions. They proposed a new hyperbolic affine transformation layer (Poincaré FC layer) and used it to construct a novel HNN framework, HNN++ (Shimizu et al., 2021).

3 Proposed Method

Neural network layers can be considered to be composed of basic operations: vector addition, scalar-vector multiplication, matrix-vector multiplication, and nonlinear activation functions. In this section, we introduce how to extend the basic operations in hyperbolic neural networks to the product space of m hyperbolic spaces, $\mathbf{P} = \mathbb{H}^{n_1} \times \cdots \times \mathbb{H}^{n_m}$. Here we will treat the case where $\mathbb{H}$ is $\mathbb{B}$ (Poincaré Ball model), and all the curvatures are the same.

3.1 Addition and Scalar Multiplication in a Product of Hyperbolic Spaces

In Euclidean space, addition $+$ and scalar multiplication $\times$ are element-wise operations, and there is no need to consider the interaction across different elements. Therefore, we define alternatives to these operations in $\mathbf{P}$ as element-wise Möbius operations:

$$\boldsymbol{x} \oplus_{\mathbf{P}} \boldsymbol{y} := (\boldsymbol{x}_1 \oplus \boldsymbol{y}_1, \ldots, \boldsymbol{x}_m \oplus \boldsymbol{y}_m), \quad (3)$$

$$r \otimes_{\mathbf{P}} \boldsymbol{y} := (r \otimes \boldsymbol{y}_1, \ldots, r \otimes \boldsymbol{y}_m), \quad (4)$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ and $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$ are tuples of points in the product space $\mathbf{P}$, and $r \in \mathbb{R}$ is a scalar value. Each point $\boldsymbol{x}_i$ (or $\boldsymbol{y}_i$) is a vector in an n_i-dimensional hyperbolic space. $\oplus$ and $\otimes$ on the right-hand side are the Möbius operations in a single hyperbolic space.

3.2 Matrix Multiplication in a Product of Hyperbolic spaces

Matrix multiplication involves interactions between different elements. Therefore, the extension of Möbius matrix multiplication to product spaces

[1] For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{B}$, Möbius Addition is defined as:

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

See Appendix A for explicit representation of other operations.

should take into account the interaction between any two hyperbolic spaces.

Let $\mathbf{P}' = \mathbb{H}^{n'_1} \times \cdots \times \mathbb{H}^{n'_{m'}}$ be the target product space of m' hyperbolic spaces of a total dimensionality $n' = n'_1 + \cdots + n'_{m'}$. Inspired by the block matrix multiplication in Euclidean space, we define Möbius matrix multiplication in $\mathbf{P}$ as follows:

$$M \otimes_{\mathbf{P}} \boldsymbol{x} = \begin{pmatrix} M_{11} & \cdots & M_{1m} \\ \vdots & \ddots & \vdots \\ M_{m'1} & \cdots & M_{m'm} \end{pmatrix} \otimes_{\mathbf{P}} \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_m \end{pmatrix}$$
$$:= \begin{pmatrix} M_{11} \otimes \boldsymbol{x}_1 \oplus \cdots \oplus M_{1m} \otimes \boldsymbol{x}_m \\ \vdots \\ M_{m'1} \otimes \boldsymbol{x}_1 \oplus \cdots \oplus M_{m'm} \otimes \boldsymbol{x}_m \end{pmatrix}, \quad (5)$$

where $M \in \mathbb{R}^{n' \times n}$ is a matrix, and $M_{ij} \in \mathbb{R}^{n'_i \times n_j}$ is a submatrix block of M. The off-diagonal blocks correspond to interactions between two different hyperbolic component spaces.

It is worth noting that Eq. (5) can be used to extend an arbitrary hyperbolic linear layer to a product space. For example, Shimizu et al. (2021) and Chen et al. (2021) defined hyperbolic linear layers $\mathcal{F}$ using a matrix parameter M, i.e., $\mathcal{F} = \mathcal{F}(M)$.[2] We can extend the hyperbolic linear layers $\mathcal{F}$ to the product space as follows:

$$\mathcal{F}_{\mathbf{P}}(M)(\boldsymbol{x}) := \begin{pmatrix} \mathcal{F}(M_{11})(\boldsymbol{x}_1) \oplus \cdots \oplus \mathcal{F}(M_{1m})(\boldsymbol{x}_m) \\ \vdots \\ \mathcal{F}(M_{m'1})(\boldsymbol{x}_1) \oplus \cdots \oplus \mathcal{F}(M_{m'm})(\boldsymbol{x}_m) \end{pmatrix}.$$

3.3 Activation Function in a Product of Hyperbolic spaces

In Euclidean space, activation functions are also element-wise operations. Activation functions in product of hyperbolic spaces can be defined as:

$$\sigma^{\otimes_{\mathbf{P}}}(\boldsymbol{y}) := (\sigma^{\otimes}(\boldsymbol{y}_1), \ldots, \sigma^{\otimes}(\boldsymbol{y}_m)). \quad (6)$$

3.4 HHGCN

HGCN (Chami et al., 2020) is a Hyperbolic version of Graph Convolutional Network (GCN) (Kipf and Welling, 2017), a widely used Graph Neural Network (GNN) architecture. First, in HGCN, each node's representation in the $(l-1)$-th layer, $\boldsymbol{x}_i^{l-1}$, is linearly transformed, i.e.,

$$\boldsymbol{h}_i^l = (W^l \otimes \boldsymbol{x}_i^{l-1}) \oplus \boldsymbol{b}. \quad (7)$$

[2]We omit the bias parameters b for simplicity.

Then, attention-based neighborhood aggregation is performed for each node through tangent space:

$$\boldsymbol{y}_i^l = \mathrm{Agg}(\boldsymbol{h}^l)_i$$
$$:= \exp_{\boldsymbol{h}_i^l}\left(\sum_{j \in N(i)} (w_{ij}^l \log_{\boldsymbol{h}_i^l}(\boldsymbol{h}_j^l))\right). \quad (8)$$

$N(i)$ denotes the set of neighboring nodes of the i-th node, and $\boldsymbol{h}^l = \{\boldsymbol{h}_j^l\}_j$ represents all the feature vectors at the l-th layer. w_{ij}^l is an attention weight calculated in tangent space:

$$w_{ij}^l = \mathrm{Softmax}_{j \in N(i)}(\mathrm{MLP}(\log_0(\boldsymbol{h}_i^l), \log_0(\boldsymbol{h}_j^l))).$$

Finally, a non-linear activation function is applied to each node:

$$\boldsymbol{x}_i^l = \sigma^{\otimes}(\boldsymbol{y}_i^l). \quad (9)$$

Now, we describe how to extend the HGCN architecture to a product space using the operations defined above: $\oplus_{\mathbf{P}}$, $\otimes_{\mathbf{P}}$, and $\sigma^{\otimes_{\mathbf{P}}}$. Here, we focus on the simplest case where the product space consists of two Hyperbolic spaces of the same dimension in each layer. We denote the extended model in $\mathbb{H} \times \mathbb{H}$ as **HHGCN**.

In HHGCN, each node's feature vector represents a tuple of points in the product space $\mathbf{P} = \mathbb{H} \times \mathbb{H}$. Let $\boldsymbol{x}_i^{l,\mathbf{P}} = (\boldsymbol{x}_i^{l,1}, \boldsymbol{x}_i^{l,2})$ $(\boldsymbol{x}_i^{l,k} \in \mathbb{H}, k = 1, 2)$ be the feature vector of the i-th node in the l-th layer. The product-space version of Eq. (7) is defined as follows:

$$\boldsymbol{h}_i^{l,\mathbf{P}} = (W^l \otimes_{\mathbf{P}} \boldsymbol{x}_i^{l-1,\mathbb{P}}) \oplus_{\mathbf{P}} \boldsymbol{b}. \quad (10)$$

Then, we perform neighborhood aggregation for each single hyperbolic space $\mathbb{H}$, i.e., for $k \in \{1, 2\}$,

$$\boldsymbol{y}_i^{l,k} = \mathrm{Agg}(\boldsymbol{h}^{l,k})_i. \quad (11)$$

Finally, we apply a non-linear activation function:

$$\boldsymbol{x}_i^{l,\mathbf{P}} = \sigma^{\otimes_{\mathbf{P}}}(\boldsymbol{y}_i^{l,\mathbf{P}}). \quad (12)$$

Note that the above extension can be applied to a product of any number of hyperbolic spaces.

3.5 Task-Specific Prediction Using HHGCN

As described in Section 3.4, HHGCN outputs embeddings $\boldsymbol{x}_i^{L,\mathbf{P}} \in \mathbf{P}$ for each node i, where L denotes the last layer. In downstream tasks such as node classification, we first project the node embeddings into a single hyperbolic space using the beta concatenation proposed in HNN++ (Shimizu et al., 2021), and then apply an appropriate task-specific decoder to the projected representation. For example, in the link prediction task, we utilize the Fermi-Dirac decoder (Krioukov et al., 2010; Nickel and Kiela, 2017) to calculate the probability score for each edge. In the node classification task, we project the representations to tangent space by $\log_0$, then perform Euclidean multinomial logistic regression, following HGCN (Chami et al., 2019).

3.6 HEGCN

We also attempt a combination of Hyperbolic space $\mathbb{H}$ and Euclidean space $\mathbb{E}$, which is expected to show high performance for less-hyperbolic datasets. We can define the linear layer like Eq. (5):

$$\begin{aligned} M \otimes \boldsymbol{x} &= \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \otimes \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} \\ &:= \begin{pmatrix} M_{11} \otimes \boldsymbol{x}_1 \oplus M_{12} \otimes \exp_0(\boldsymbol{x}_2) \\ M_{21} \log_0(\boldsymbol{x}_1) + M_{22}\boldsymbol{x}_2 \end{pmatrix}, \end{aligned} \tag{13}$$

where M denotes a matrix, and $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a point of the product space $\mathbb{H} \times \mathbb{E}$. We can extend the GCN for this product space $\mathbb{H} \times \mathbb{E}$ in a similar way to Section 3.4 and call this model **HEGCN**.

4 Experiments

4.1 Setup

Following the previous studies on Hyperbolic GCNs (Chami et al., 2019; Chen et al., 2021), we evaluate our method in two tasks: node classification (NC) and link prediction (LP). We use four network embedding datasets: Disease and Airport (Chami et al., 2019), PubMed (Namata et al., 2012), and Cora (Sen et al., 2008). For each dataset, we show the Gromov's δ-hyperbolicity (Adcock et al., 2013; Narayan and Saniee, 2011; Jonckheere et al., 2008) calculated by Chami et al. (2019) with the results. Lower δ means higher tree-likeness, and thus hyperbolic architectures are expected to show higher performance.

To test the effectiveness of our method and neural networks in a product space of hyperbolic spaces, we adopt HGCN and the following Euclidean GNNs as the baselines: GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), SAGE (Hamilton et al., 2017), and SGC (Wu et al., 2019).

We also test the effectiveness of our method using different hyperbolic space representations: hyperboloid and Poincaré ball: HHGCN$_\mathrm{h}$ and HEGCN$_\mathrm{h}$ use hyperboloid, while HHGCN$_\mathrm{p}$ and HEGCN$_\mathrm{p}$ use Poincaré ball as the hyperbolic space. The difference changes the explicit formula of the Möbius operations used in HNN and may change computational stability. Note that HGCN uses the hyperboloid model.

We mainly follow the training setups of previous studies (Chami et al., 2019; Chen et al., 2021). The dimensions are all set to $n = 16 = 8 + 8$ for fair comparison. We use Riemannian Adam (rAdam) optimizer (Becigneul and Ganea, 2019) for hyperbolic parameters. Curvatures of hyperbolic space are set to -1 for our product-space models. Please refer to Appendix D for detailed information.

4.2 Results and Discussion

Table 1 shows the performance of the proposed and baseline models.

HHGCN vs. HGCN, GCN

For the tree-like datasets with lower δ (Disease, Airport), HHGCNs show higher performance than the baselines especially in the node classification task. Particularly, HHGCN$_\mathrm{h}$ shows comparable or better results than the baselines on every task in these datasets and yields significant improvement in Disease. In contrast, for the datasets with higher δ (PubMed, Cora), HHGCNs consistently underperform HGCN. These results suggest that HHGCN is more effective than the single-space counterparts especially in hyperbolic datasets.

HHGCN vs. HEGCN

Table 1 demonstrates that HHGCN outperforms HEGCN on the datasets with lower δ and slightly underperform with lower δ. These results suggest that HEGCN is less specialized to tree-like datasets than HHGCN due to the incorporation of Euclidean space. On the other hand, unexpectedly, the performance of HEGCN in Pubmed and Cora is worse than those of HGCN, even though HGCN uses only hyperbolic space. These results may suggest that Eq. (13) for HEGCN is insufficient to represent the interaction between spaces with different properties.

Hyperboloid vs. Poincaré ball

We can observe that HHGCN$_\mathrm{h}$ and HEGCN$_\mathrm{h}$ show stable performance, while HHGCN$_\mathrm{p}$ and HEGCN$_\mathrm{p}$ show performance degradation in the Disease dataset in the LP task. These results may

Dataset Hyperbolicity	Disease $\delta = 0$		Airport $\delta = 1$		PubMed $\delta = 3.5$		Cora $\delta = 11$	
Method	LP	NC	LP	NC	LP	NC	LP	NC
GCN (2017)	$64.7_{\pm 0.5}$	$69.7_{\pm 0.4}$	$89.3_{\pm 0.4}$	$81.4_{\pm 0.6}$	$91.1_{\pm 0.5}$	$78.1_{\pm 0.2}$	$90.4_{\pm 0.2}$	$81.3_{\pm 0.3}$
GAT (2018)	$69.8_{\pm 0.3}$	$70.4_{\pm 0.4}$	$90.5_{\pm 0.3}$	$81.5_{\pm 0.3}$	$91.2_{\pm 0.1}$	$79.0_{\pm 0.3}$	$\mathbf{93.7}_{\pm 0.1}$	$\mathbf{83.0}_{\pm 0.7}$
SAGE (2017)	$65.9_{\pm 0.3}$	$69.1_{\pm 0.6}$	$90.4_{\pm 0.5}$	$82.1_{\pm 0.5}$	$86.2_{\pm 1.0}$	$77.4_{\pm 2.2}$	$85.5_{\pm 0.6}$	$77.9_{\pm 2.4}$
SGC (2019)	$65.1_{\pm 0.2}$	$69.5_{\pm 0.2}$	$89.8_{\pm 0.3}$	$80.6_{\pm 0.1}$	$94.1_{\pm 0.0}$	$78.9_{\pm 0.0}$	$91.5_{\pm 0.1}$	$81.0_{\pm 0.1}$
HGCN (2019)	$90.8_{\pm 0.3}$	$74.5_{\pm 0.9}$	$96.4_{\pm 0.1}$	$90.6_{\pm 0.2}$	$\mathbf{96.3}_{\pm 0.0}$	$\mathbf{80.3}_{\pm 0.3}$	$92.9_{\pm 0.1}$	$79.9_{\pm 0.2}$
HHGCN$_\mathrm{h}$	$\underline{\mathbf{96.1}}_{\pm 0.8}$	$\underline{94.0}_{\pm 1.2}$	$\underline{\mathbf{96.7}}_{\pm 0.3}$	$\underline{90.9}_{\pm 2.3}$	$93.4_{\pm 1.6}$	$76.0_{\pm 1.1}$	$92.8_{\pm 2.1}$	$77.2_{\pm 1.7}$
HHGCN$_\mathrm{p}$	$89.9_{\pm 5.6}$	$\underline{\mathbf{94.7}}_{\pm 1.3}$	$94.9_{\pm 1.3}$	$\underline{\mathbf{93.8}}_{\pm 0.8}$	$93.2_{\pm 1.9}$	$75.9_{\pm 0.4}$	$91.8_{\pm 3.0}$	$78.4_{\pm 1.3}$
HEGCN$_\mathrm{h}$	$\underline{93.6}_{\pm 1.9}$	$\underline{94.0}_{\pm 0.8}$	$94.0_{\pm 0.3}$	$\underline{91.5}_{\pm 1.6}$	$94.8_{\pm 0.8}$	$76.1_{\pm 0.6}$	$\underline{93.2}_{\pm 1.4}$	$78.3_{\pm 1.2}$
HEGCN$_\mathrm{p}$	$86.4_{\pm 2.0}$	$\underline{94.1}_{\pm 1.1}$	$95.7_{\pm 1.0}$	$\underline{92.9}_{\pm 1.1}$	$95.0_{\pm 1.7}$	$76.2_{\pm 0.5}$	$\underline{93.2}_{\pm 1.2}$	$78.{}_{\pm 1.1}3$

Table 1: ROC AUC (%) for the link prediction (LP) task and F1 scores (%) for the node classification (NC) task. The best scores for each column are shown in bold. We underline the scores of HHGCN and HEGCN if the scores are higher than the baselines' scores.

be due to the learning instability of the Poincaré ball model mentioned by Nickel and Kiela (2018).

HHGCN with HNNs Variants

Recently, Shimizu et al. (2021) proposed HNN++, which introduced a novel linear transformation in Poincaré ball with less distortion than the tangent space approximation Eq. (5). However, to the best of our knowledge, HNN++ has not been applied to the HGCN even in a single space. Thus, we apply the HNN++ to HGCN and HHGCN by replacing the hyperbolic transformation in Eq. (7) and further compare these models. We call the extensions HGCN$_{++}$ and HHGCN$_{++}$, respectively. We also extend HyboNet (Chen et al., 2021), a novel HNN architecture in the Lorentz model, to HHGCN. We denote this extension as HHGCN$_\mathrm{HN}$.

Table 2 shows the results. HHGCN$_{++}$ yields higher or comparable performance than HGCN$_{++}$ in the NC task. In contrast, in the LP task on the Disease dataset, HHGCN$_{++}$ shows performance degradation. On the other hand, HHGCN$_\mathrm{HN}$ underperforms HyboNet in most cases except for the NC task on the Airport dataset.

These results suggest that certain HNN variants are not effective in extending to the product space. We leave more in-depth investigation to future work.

5 Preliminary Experiments on Machine Translation

5.1 Setup

We also tested the applicability of our method to machine translation tasks.

In the paper proposing HNN++, Shimizu et al. constructed a hyperbolic version of the convolutional sequence-to-sequence (ConvSeq2Seq) model (Gehring et al., 2017) by replacing various operations with the new hyperbolic operations they proposed, and applied it to machine translation. We extended their model to the product of two hyperbolic spaces using the operations proposed in Section 3.

They used WMT'17 English-German (Bojar et al., 2017) dataset containing 4M sentence pairs as training data. We extract 40K sentence pairs as a training dataset from it for the preliminary experiments. We train several scaled-down models with Riemannian Adam for 5K iterations. For more implementation details, please refer to Appendix E.

5.2 Results and Discussion

Table 3 shows that our model outperformed HNN++, albeit with lower overall performance due to the small size of the training data. All models show a significant performance drop at D=256. This may be due to the models being too large for the training data. Shimizu et al. suggested that the reason why the Euclidean model performs better

Dataset	Disease		Airport		PubMed		Cora	
Method	LP	NC	LP	NC	LP	NC	LP	NC
HGCN$_{++}$	$89.1_{\pm1.3}$	$88.0_{\pm4.3}$	$96.8_{\pm0.2}$	$86.8_{\pm2.5}$	$93.0_{\pm0.2}$	$75.2_{\pm1.7}$	$89.2_{\pm0.6}$	$80.1_{\pm0.6}$
HHGCN$_{++}$	$84.3_{\pm1.9}$	$\underline{90.6}_{\pm3.3}$	$96.3_{\pm0.6}$	$\underline{91.4}_{\pm1.1}$	$92.8_{\pm0.2}$	$\underline{75.9}_{\pm0.9}$	$\underline{89.3}_{\pm1.2}$	$79_{\pm0.4}$
HyboNet (2021)	$96.3_{\pm0.3}$	$94.5_{\pm0.8}$	$97.0_{\pm0.2}$	$92.5_{\pm0.9}$	$96.4_{\pm0.1}$	$77.9_{\pm1.0}$	$94.3_{\pm0.3}$	$81.3_{\pm0.9}$
HHGCN$_{\mathrm{HN}}$	$92.6_{\pm1.7}$	$94.4_{\pm3.9}$	$94.1_{\pm0.9}$	$\underline{93.6}_{\pm0.8}$	$94.1_{\pm0.8}$	$74.8_{\pm0.9}$	$88.0_{\pm1.3}$	$74.3_{\pm1.6}$

Table 2: Results of the extension of HNN variants to a product space. We underline the scores of HHGCN$_{++}$ and HHGCN$_{\mathrm{HN}}$ if these models outperform the corresponding single-space counterparts.

than the Hyperbolic model as the dimensionality increases is that sufficient computational complexity can be obtained through optimization. The fact that the Euclidean ConvSeq2Seq has the lowest results for D=256 may be due to its complexity resulting in overfitting. Comparative experiments with larger data sets are still needed, which we plan to do in the near future.

6 Related Work

GCN in a Product Space

κ-GCN with learnable curvature κ was proposed by Bachmann et al. (2020). They also attempted learning on the product of two constant curvature spaces. Unlike our results, HGCN showed better performance than their product space model in the Airport node classification task. It suggests that our proposed method is more suited to datasets with tree-like structures.

Hyperbolic-Euclidean Hybrid Model

Graph embedding in $\mathbb{H} \times \mathbb{E}$ considering the interaction between $\mathbb{H}$ and $\mathbb{E}$ has been done by GIL (Zhu et al., 2020a). While GIL is specialized for graphs, our method (Eq. 13) is applicable to general neural networks in $\mathbb{H} \times \mathbb{E}$ not limited to GNNs.

7 Conclusion

We proposed a general method to extend existing single-space HNN architectures to a product Space. We applied our method to HGCN and conducted experiments across several graph datasets and HNN architectures. The results show that models using a product of hyperbolic spaces perform better on tree-like datasets than models using a single hyperbolic space especially in the node classification task.

Future Work

We applied our method to the several HNN variants and found that our method was effective with some HNN types but not with others. The theoretical explanation for this difference will be a future issue.

We plan to conduct machine translation experiments using the entire WMT'17 Endlish-German training data and to apply our method to Transformer-based machine translation models in the near future. We are also going to investigate the effectiveness and limitations of our method on other NLP tasks such as natural language inference and document classification.

Acknowledgements

These research results were obtained from the commissioned research (No. 225) by National Institute of Information and Communications Technology (NICT), Japan. This work was also supported by JSPS KAKENHI Grant Number JP19H04166.

References

Aaron B. Adcock, Blair D. Sullivan, and Michael W. Mahoney. 2013. Tree-like structure in large social and information networks. In *2013 IEEE 13th International Conference on Data Mining*, pages 1–10.

Roy M. Anderson and Robert M. (Robert McCredie) May. 1991. *Infectious diseases of humans : dynamics and control*. Oxford University Press.

Gregor Bachmann, Gary Bécigneul, and Octavian-Eugen Ganea. 2020. Constant curvature graph convolutional networks.

Gary Becigneul and Octavian-Eugen Ganea. 2019. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara

Model	D=16	D=64	D=256
ConvSeq2Seq (Gehring et al., 2017)	1.30	**2.47**	0.0
HNN++ (Shimizu et al., 2021)	1.41	1.29	0.86
Ours	**1.94**	2.17	**0.87**

Table 3: BLEU-4 (Papineni et al., 2002) scores for the machine translation using the small dataset we extracted from WMT'17 English-German dataset. D denotes the dimensions of token embeddings.

Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. 1997. Hyperbolic geometry. *Flavors of geometry*, 31(59-115):2.

Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. 2017. Neural embeddings of graphs in hyperbolic space.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4869–4880.

Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Fully hyperbolic neural networks. *CoRR*, abs/2105.14686.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2019. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. 2018. Hyperbolic attention networks.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034.

Edmond Jonckheere, Poonsuk Lohsoonthorn, and Francis Bonahon. 2008. Scaled gromov hyperbolic graphs. *Journal of Graph Theory*, 57(2):157–180.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Max Kochurov, Rasul Karimov, and Serge Kozlukov. 2020. Geoopt: Riemannian optimization in pytorch.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. 2010. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82:036106.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. 2012. Query-driven active surveying for collective classification. In *ICML Workshop on MLG*.

Onuttom Narayan and Iraj Saniee. 2011. Large-scale curvature of networks. *Phys. Rev. E*, 84:066108.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations.

Maximilian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Peter Petersen, S Axler, and KA Ribet. 2006. *Riemannian geometry*, volume 171. Springer.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. 2018. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4460–4469. PMLR.

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

Ryohei Shimizu, YUSUKE Mukuta, and Tatsuya Harada. 2021. Hyperbolic neural networks++. In *International Conference on Learning Representations*.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2018. Poincaré glove: Hyperbolic word embeddings.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. 2020a. Graph geometry interaction learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yudong Zhu, Di Zhou, Jinghui Xiao, Xin Jiang, Xiao Chen, and Qun Liu. 2020b. HyperText: Endowing FastText with hyperbolic geometry. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1166–1171, Online. Association for Computational Linguistics.

A Operations in hyperbolic space

Möbius Addition $\oplus$ and Möbius Scalar Multiplication $\otimes$ in the Poincaré Ball model

The hyperbolic versions of addition(Möbius Addition $\oplus$) and scalar multiplication (Möbius Scalar Multiplication $\otimes$) are defined as follows:

$$\boldsymbol{x} \oplus_c \boldsymbol{y} := \frac{(1 + 2c\langle \boldsymbol{x}, \boldsymbol{y}\rangle + c\|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 - c\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 + 2c\langle \boldsymbol{x}, \boldsymbol{y}\rangle + c^2\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2},$$
$$r \otimes_c \boldsymbol{x} := \frac{1}{\sqrt{c}} \tanh(r \tanh^{-1}(\sqrt{c}\|\boldsymbol{x}\|))\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|},$$

where $\boldsymbol{x}, \boldsymbol{y}$ are points in a hyperbolic space $\mathbb{B}$ and $r \in \mathbb{R}$ is scalar value.

These were first introduced in the context of Einstein's special theory of relativity in order to successfully describe the composite law of velocities such that the absolute value does not exceed the speed of light (Ungar, 2008).

Exp and Log Maps in the Poincaré Ball model

For $\boldsymbol{v} \in T_0\mathbb{B}^n_c \setminus \{0\}$ and $\boldsymbol{y} \in \mathbb{B}^n_c \setminus \{0\}$,

$$\exp^c_0(\boldsymbol{v}) = \tanh(\sqrt{c}\|\boldsymbol{v}\|)\frac{\boldsymbol{v}}{\sqrt{c}\|\boldsymbol{v}\|},$$
$$\log^c_0(\boldsymbol{y}) = \tanh^{-1}(\sqrt{c}\|\boldsymbol{y}\|)\frac{\boldsymbol{y}}{\sqrt{c}\|\boldsymbol{y}\|}.$$

Attention Mechanism in Hyperbolic Space

In order to realize the attention mechanism, centroid (weighted sum) in hyperbolic space had several definitions depending on the model, but Shimizu et al. (2021) showed that they are equivalent to Möbius gyromidpoint. For the Poincaré Ball model, the Möbius gyromidpoint $\boldsymbol{m}$ of hyperbolic vectors $\{\boldsymbol{b}_i \in \mathbb{B}^n_c\}_{i=1}^N$ with the scalar weights $\{\nu_i \in \mathbb{R}\}_{i=1}^N$ is defined as:

$$\boldsymbol{m} = \mathsf{Centroid}(\{\nu_i \in \mathbb{R}\}_{i=1}^N, \{\boldsymbol{b}_i \in \mathbb{B}^n_c\}_{i=1}^N) := \frac{1}{2} \otimes_c \left(\frac{\sum_{i=1}^N \nu_i \quad \lambda^c_{\boldsymbol{b}_i} \boldsymbol{b}_i}{\sum_{i=1}^N |\nu_i|(\lambda^c_{\boldsymbol{b}_i} - 1)} \right). \qquad (14)$$

B Decoding Mechanism

B.1 Beta Concatenation

We utilized beta concatetenation proposed in HNN++ (Shimizu et al., 2021) to project product-space representations into a single hyperbolic space:

$$\boldsymbol{x}_i^{\text{out}} = \exp_0\left(\frac{\beta_n}{\beta_{n_1}}\log_0(\boldsymbol{x}_i^{L,1}), \frac{\beta_n}{\beta_{n_2}}\log_0(\boldsymbol{x}_i^{L,2})\right). \qquad (15)$$

Where n is the overall dimension, and n_i is the dimension of i-th space (here $n_1 = n_2 = \frac{n}{2}$). Inside the $\exp_0$ parentheses, the usual concatenation of two Euclidean vectors is performed. $\beta_N := B(\frac{N}{2}, \frac{1}{2})$ (B : beta function) are the scaling factors to preserve the expectation of the norm.

B.2 Fermi-Dirac Decoder

For link prediction task, we utilize the Fermi-Dirac decoder (Krioukov et al., 2010; Nickel and Kiela, 2017), a generalization of sigmoid, to calculate the probability score for edges:

$$p(i,j) = [e^{(d_{\mathbb{H}}(\boldsymbol{x}_i^{\text{out}}, \boldsymbol{x}_j^{\text{out}}) - r)/t}]^{-1}. \qquad (16)$$

where $r, t > 0$ are hyperparameters and $d_{\mathbb{H}}$ is distance function of hyperbolic space $\mathbb{H}$.

C Dataset Description

We use four network embedding datasets, Disease (Chami et al., 2019), Airport (Chami et al., 2019), PubMed (Namata et al., 2012), and Cora (Sen et al., 2008) following Chami et al. (2019); Chen et al. (2021). PubMed and Cora are standard benchmarks, where nodes are scientific papers, edges are citations between them, and node labels represent the academic domains of the papers. The first two datasets are constructed by Chami et al. (2019). Disease is a tree dataset built by simulating SIR disease spread model (Anderson and May, 1991), and Airport is a graph dataset consisting of airports and air routes obtained from OpenFlights.org [3].

The four datasets are preprocessed by Chami et al. (2019) and published in their code repository.[4] We show statistics of the datasets in table 4. For more information, please refer to the paper (Chami et al., 2019).

D Details on Network Embedding Experiments

We utilize Geoopt (Kochurov et al., 2020) and Riemannian Adam (rAdam) optimizer (Becigneul and Ganea, 2019) for hyperbolic parameters. For each dataset and model, we conduct hyper-parameter

[3] `https://openflights.org`
[4] `https://github.com/HazyResearch/hgcn`

Name	Nodes	Edges	Classes	Node features
Disease	1044	1043	2	1000
Airport	3188	18631	4	4
PubMed	19717	88651	3	500
Cora	2708	5429	7	1433

Table 4: Datasets' statistics.

search over Dropout $\in \{0, 0.1, 0.3, 0.7, 0.9\}$, Weight-Decay $\in \{0, 0.01, 0.1, 0.2, 0.4\}$. The performance is evaluated using five different random seeds for each condition. We fixed curvatures of hyperbolic spaces to -1. This is because learnable curvature sometimes showed instability. Some hyper-parameters, such as the initial learning rate and the number of layers, are fixed as shown in table 5, with reference to the previous study (Chen et al., 2021).

E Machine Translation Experiments

Following the setting of HNN++ (Shimizu et al., 2021), each model is the encoder-decoder model, both of which are composed of five convolutional layers with a kernel size of three and a channel size of D, five convolutional layers with a kernel size of three and a channel size of 2D, and two convolutional layers with a kernel size of one and a channel size of 4D. In each layer of our model, the hyperbolic affine transformation of HNN++ is replaced by its extension to the product of two hyperbolic spaces.

For training and optimization, we mainly follow the setting of HNN++. The main differences are the size of the dataset and iteration numbers. We extract (40K, 10K, 1K) sentences from the entire WMT'17 English-German dataset consisting of (4M, 40K, 3K) sentences for (training, validation, test). We trained models for 5K iterations instead of 100K iterations.

We use the same parameter as HNN++ for the Riemannian Adam optimizer; $\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The warm-up period was set as the first 400 iteration instead of 4000 iteration.

Dataset	Disease		Airport		PubMed		Cora	
Task	LP	NC	LP	NC	LP	NC	LP	NC
Layers	2	4	2	6	2	3	2	3
Initial Learning Rate	0.005	0.005	0.01	0.02	0.008	0.02	0.02	0.02
Max Grad Norm	None	0.5	0.5	1	0.5	0.5	0.5	1

Table 5: Hyper-parameters for each task.