Lorentzian Residual Neural Networks

Neil He [1] **Menglin Yang** [1] **Rex Ying** [1]

Abstract

Hyperbolic neural networks have emerged as a powerful tool for modeling hierarchical data structures prevalent in real-world datasets. Notably, residual connections, which facilitate the flow of information across layers, have been instrumental in the success of deep neural networks. However, current methods for constructing hyperbolic residual layers suffer from limitations such as increased model complexity, numerical instability, and errors due to multiple mappings to and from the tangent space. To address these limitations, we introduce LRN, a novel hyperbolic residual neural network based on the *weighted Lorentzian centroid in the Lorentz model of hyperbolic space*. Extensive experiments showcase the superior performance of LRN compared to state-of-the-art Euclidean and hyperbolic alternatives, highlighting its potential for building more expressive neural networks in hyperbolic space as a general applicable method to multiple architectures, including GNNs and graph Transformers.[1]

1. Introduction

In recent years, the exploration of neural network architectures beyond the traditional Euclidean space has opened up new frontiers in machine learning research (Peng et al., 2021; Mettes et al., 2023; Yang et al., 2022c). Among these, hyperbolic neural networks (Ganea et al., 2018; Shimizu et al., 2020; van Spengler et al., 2023; Chami et al., 2019; Liu et al., 2019; Bdeir et al., 2024; Chen et al., 2021) have garnered significant attention due to their inherent capabilities to model data with complex hierarchical structures. Hyperbolic spaces, characterized by constant negative curvature, naturally align with the geometric properties of tree-like data, offering a more fitting representation than their Euclidean counterparts (Krioukov et al., 2010; Sarkar, 2011). This alignment enables enhanced learning efficiency and improves representation learning for a wide range of applications, from graph-based data analysis (Faqeeh et al., 2018; Yang et al., 2022e; Sun et al., 2021; Yang et al., 2022d;a; Liu et al., 2019; Chami et al., 2019; Dai et al., 2021) to image understanding (Mettes et al., 2023; Desai et al., 2023).

Residual connection (He et al., 2016) is a core elements of the modern deep learning framework as a powerful mechanism that has revolutionized the development of deep neural networks. By allowing layers to learn modifications to the identity mapping rather than complete transformations, residual connections facilitate the training of substantially deeper networks and are widely used in many model architectures, including CNNs, GNNs, and Transformers (He et al., 2016; Veličković et al., 2018; Vaswani et al., 2017). However, the residual connections method within Euclidean spaces is not directly transferable to the geometry of hyperbolic spaces, which is violated through direction additions.

Limitations of existing works Poincaré ResNet (van Spengler et al., 2023) proposes projecting the hyperbolic embeddings to the tangent space at a fixed point, parallel transporting it to the tangent space at the position of interest, and then utilizing the exponential map to map it back to the hyperbolic space. Another similar work, Riemannian ResNet (Katsman et al., 2023), although not defined in hyperbolic space, employs similar concepts. Despite being intuitive, these approaches have several limitations:

(i) Computationally Expensive. The operations require multiple mappings to and from the tangent space. These are complex mappings and significantly increases computational complexity (see Table 3).

(ii) Non-Commutativity. $w_1\mathbf{x}+w_2f(\mathbf{x}) = w_2f(\mathbf{x})+w_1\mathbf{x}$ holds in Euclidean, whereas the parallel transport method is not commutative, greatly restricting the expressiveness of the model (see Appendix C.1).

(iii) Numerical Instability. For instance, the log mapping in the Lorentz model (of curvature -1) is:

$$\log_{\mathbf{u}}(\mathbf{v}) = \frac{\cosh^{-1}(\alpha)}{\sqrt{\alpha^2-1}}(\mathbf{v}-\alpha\mathbf{u}).$$

where $\alpha = \mathbf{u}_0\mathbf{v}_0 - \sum_{i=1}^{n}\mathbf{u}_i\mathbf{v}_i$ is the Lorentzian inner

[1]Yale University. Correspondence to: Menglin Yang <menglin.yang@yale.edu>.

Accepted as an extended abstract for the Geometry-grounded Representation Learning and Generative Modeling Workshop at the 41st International Conference on Machine Learning, ICML 2024, Vienna, Austria.

[1]Code: https://github.com/784956494/LRN

product of $\mathbf{u}, \mathbf{v}$. If $\mathbf{u} = \mathbf{v}$ and $\mathbf{u}$ contains large values in the coordinates, α becomes less than one, which results in NaN because the domain of $\cosh^{-1}(x)$ is $x \geq 1$.

(iv) Mapping Error. The above method requires mapping a point in hyperbolic space to the tangent space. For efficient computation, the origin is commonly used as the reference point. This introduces mapping errors for points not at the origin (Yu & De Sa, 2019).

Other previous methods have their own shortcomings. As a method previously used as hyperbolic addition for aggregation, HGCN (Chami et al., 2019) and LGCN Zhang et al. (2021b) proposed first mapping to the tangent space of the origin for addition, and then projecting the sum back into hyperbolic space. Although this method is commutative, it suffers from (i), (iii), and (iv). HCNN (Bdeir et al., 2024) adds the space component of the hyperbolic embeddings and then computes the time component. While not suffering from the same set of problems, it does not have geometric meaning to provide justification for why it works.

Proposed method In this work, we propose Lorentzian Residual Networks (LRN), a residual neural network based on the weighted Lorentzian centroid in the Lorentz model. Unlike some existing methods, we do not rely on mapping between tangent space and hyperbolic space but instead operate directly on the Lorentz embedding. Unlike HCNN (Bdeir et al., 2024), our method has geometric interpretations. LRN addresses the limitations previously mentioned in (i), (iii), and (iv), and ensures the commutativity of addition, thereby resolving limitation (ii).

Contributions The main contributions of this work can be summarized as follows: (1) We introduce LRN, a novel residual connection method for hyperbolic neural networks that operates directly on the Lorentz embedding and has geometric interpretations. (2) We theoretically demonstrate that LRN can derive previous methods while ensuring commutativity and numerical stability. (3) We experimentally validate the superior performance of LRN across multiple tasks, and across multiple architectures such as GNN and graph Transformer. (4) We provide a comprehensive analysis of the limitations of existing methods and highlight the potential of LRN in advancing the field of hyperbolic representation learning.

2. Methodology

In this section, we introduce our proposed method for Lorentzian residual connection, based on a generalization of the Lorentzian centroid (Law et al., 2019). In a typical Euclidean residual block, let $\mathbf{x}$ be the input and $f(\mathbf{x})$ be the output of a neural network layer or a series of layers. The residual connection is $\mathbf{x} + f(\mathbf{x})$, or in a more generalized form, $\alpha\mathbf{x} + \beta f(\mathbf{x})$, where α, β are scalar weights.

Adapting the general form, given vectors $\mathbf{x}, f(\mathbf{x}) \in \mathbb{L}_K^n$, the Lorentz residual connection is computed as:

$$\begin{aligned}\mathbf{m} &= \frac{w_x\mathbf{x} + w_y f(\mathbf{x})}{\sqrt{-K}\left|\|w_x\mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}}\right|} \\ &= \alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x}),\end{aligned} \tag{1}$$

where $|||\cdot|||_{\mathcal{L}} = \sqrt{|||\cdot||_{\mathcal{L}}^2|}$, $\alpha_{w_x,w_y} = w_x/\sqrt{-K}\left|\|w_x\mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}}\right|$, $\beta_{w_x,w_y} = w_y/\sqrt{-K}\left|\|w_x\mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}}\right|$ and $w_x, w_y > 0$ are weights that can be learned or fixed.

Lorentz residual network (LRN) The core component of LRN is the Lorentzian residual block, consisting of a hyperbolic layer followed by a Lorentzian residual connection. The hyperbolic layer can be any type of layer that operates in $\mathbb{L}_K^n$, such as hyperbolic fully-connected layers (Chen et al., 2021), hyperbolic convolutional layers (Bdeir et al., 2024), or hyperbolic attention mechanisms (Gulcehre et al., 2019). In Algorithm 1, we present an example of LRN in a classification network for better understanding.

Previous hyperbolic addition methods Here we take a detour and introduce formally the previous methods for hyperbolic addition. Let $\mathbb{L}_K^n$ be the n-dimensional Lorentz hyperbolic space with constant negative curvature $K(K < 0)$, equipped with the Lorentzian inner product $\langle\cdot\rangle_{\mathcal{L}}$ and norm $\|\cdot\|_{\mathcal{L}} := \sqrt{\langle\cdot\rangle_{\mathcal{L}}}$. For $\mathbf{x}, \mathbf{y} \in \mathbb{L}_K^n$, $\exp_{\mathbf{x}}^K(\cdot), \log_{\mathbf{x}}^K(\cdot)$ denote the exponential and logarithmic maps at $\mathbf{x}$, and $\mathbf{P}_{\mathbf{x}\to\mathbf{y}}$ denote the parallel transport map from the tangent space of $\mathbf{x}$ to that of $\mathbf{y}$. $\mathbf{o} = [\sqrt{-1/K}, 0, \ldots, 0]^T$ is the origin.

The *parallel transport method* of Mobius Addition from (Chami et al., 2019; Ganea et al., 2018) used by Poincaré ResNet (van Spengler et al., 2023) is given by

$$\mathbf{x} \oplus_P \mathbf{y} = \exp_{\mathbf{x}}^K \circ \mathbf{P}_{\mathbf{o}\to\mathbf{x}} \circ \log_{\mathbf{o}}^K(\mathbf{y}). \tag{2}$$

The *tangent space method* used for aggregation in (Chami et al., 2019; Zhang et al., 2021b):

$$\mathbf{x} \oplus_T \mathbf{y} = \exp_{\mathbf{o}}^K\left(w_x \log_{\mathbf{o}}^K(\mathbf{x}) + w_y \log_{\mathbf{o}}^K(\mathbf{y})\right). \tag{3}$$

where $w_x, w_y > 0$ are weights. A third approach is the *space addition method* from (Bdeir et al., 2024) given by

$$\mathbf{x} \oplus_S \mathbf{y} = \left[\sqrt{||\mathbf{x}_s + \mathbf{y}_s||^2 - 1/K}, \mathbf{x}_s + \mathbf{y}_s\right]^T, \tag{4}$$

where $\mathbf{x}_s, \mathbf{y}_s$ denote the space-like dimension of $\mathbf{x}, \mathbf{y}$.

Advantages over previous limitations LRN resolves all of the previous methods' limitations mentioned in Section 1.

Algorithm 1 Lorentz Residual Network (LRN)

Require: Input data $\mathbf{x} \in \mathbb{L}_K^n$, target labels y, learning rate η, number of layers L, loss function ℓ, initial weights $w_x^{(l)}$ and $w_y^{(l)}$ for $l = 1, \ldots, L$

for each epoch **do**
  for each batch of data $\mathbf{x}_b$ **do**
    $\mathbf{h}^{(0)} \leftarrow \mathbf{x}_b$ {Initialization}
    for $l \leftarrow 1$ to L **do**
      $\mathbf{z}^{(l)} \leftarrow f^{(l)}(\mathbf{h}^{(l-1)})$ {Hyperbolic layers}
      $\alpha^{(l)} \leftarrow \frac{w_x^{(l)}}{\sqrt{-K}\left|\|w_x\mathbf{x}+w_y f(\mathbf{x})\|_{\mathcal{L}}\right|}$
      $\beta^{(l)} \leftarrow \frac{w_y^{(l)}}{\sqrt{-K}\left|\|w_x\mathbf{x}+w_y f(\mathbf{x})\|_{\mathcal{L}}\right|}$
      $\mathbf{h}^{(l)} \leftarrow \alpha^{(l)}\mathbf{h}^{(l-1)} + \beta^{(l)}\mathbf{z}^{(l)}$ {LRN }
    end for
    $\hat{\mathbf{y}} \leftarrow \text{softmax}(\mathbf{h}^{(L)})$ {Output prediction}
    $\mathcal{J} \leftarrow \ell(\hat{\mathbf{y}}, \mathbf{y})$ {Compute loss}
    Update weights $w_x^{(l)}$ and $w_y^{(l)}$ for $l = 1, \ldots, L$ using optimizer with $\mathcal{J}$ and η
  end for
end for

(i) LRN is significantly more efficient than previous methods that map between tangent and hyperbolic space. See Table 3 for further details.

(ii) LRN is commutative, as $w_x\mathbf{x} + w_y\mathbf{y} = w_y\mathbf{y} + w_x\mathbf{x}$. levitating the consequences of the non-commutativity of the parallel transport method.

(iii) The following lemma shows that LRN is computationally stable, avoiding the computational instability issues mentioned in the introduction.

Lemma 2.1. $\sqrt{-K}\,|\|w_x\mathbf{x} + w_y\mathbf{y}\|_{\mathcal{L}}| > \sqrt{w_x^2 + w_y^2}$ *for any* $\mathbf{x}, \mathbf{y} \in \mathbb{L}_K^n$ *and* $w_x, w_y > 0$.

Since the normalizing denominator divides out the ratio w_x/w_y, the output of LRN is preserved when the ratio is preserved. Thus we can fix $w_x = 1$ and obtain $||w_x\mathbf{x}+w_y\mathbf{y}||_{\mathcal{L}} > 1/\sqrt{-K}$, never risking dividing by values close to zero, ensuring stability.

(iv) By not mapping between tangent and hyperbolic spaces, LRN eliminates mapping errors for points far from the origin (Yu & De Sa, 2019).

(v) LRN has geometric interpretation, having the ability to theoretically achieve the previous methods as demonstrated by Proposition 2.3. By carefully selecting weights, LRN is able to achieve the geometric meaning of previous methods by ensuring the outputs lie on the same geodesic from the origin. This provides theoretical motivation for LRN as opposed to the space addition method (Bdeir et al., 2024) mentioned in Section 1.

For completeness and rigor, we include the following for the non-commutativity of the parallel transport method. The two directions of addition in some cases are reflected over an axis, giving theoretical motivation for its inflexibility.

Theorem 2.2. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{L}_K^n$ *be points such that* $\mathbf{x}_i = \mathbf{y}_i$ *for* $i \neq n+1$ *and* $\mathbf{x}_{n+1} = -\mathbf{y}_{n+1}$. *Let* $\mathbf{z} = \mathbf{x} \oplus_P \mathbf{y}$ *be the output of the parallel transport method, and let* $\mathbf{z}' = \mathbf{y} \oplus_P \mathbf{x}$ *be the output in the other direction. Then* $\mathbf{z}_{n+1} = -\mathbf{z}'_{n+1}$.

Relation to previous methods Our approach can theoretical derive previous methods mentioned in Section 2, based on the following Proposition 2.3. Proofs and details for all of our theoretical results can be found in Appendix D.

Proposition 2.3. *Let* $\mathbf{z}$ *be the output of one of Equation* (2)*, Equation* (3)*, Equation* (4)*. Then there exists weights* $w_x, w_y \in \mathbb{R}^+$ *such that the point* $\mathbf{m} = \alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y}\mathbf{y}$ *lies on the geodesic from* $\mathbf{o}$ *to* $\mathbf{z}$.

3. Experiments

We apply LRN as the residual connection in hyperbolic neural networks. We demonstrate that LRN achieves better results than any previous methods of residual connection. We focus on testing the residual connection methods by using a consistent base model in each task. For more details regarding our experiments, please see Appendices C and F.

GNN model architecture We formulate a skip-connected graph convolutions network with LRN as the residual connection, using the fully hyperbolic graph convolutional layer (Chen et al., 2021). For the overall model architecture, we follow the SkipGCN architecture outlined in (Sun et al., 2021; Yang et al., 2022a), where the last layer aggregates representations from all intermediate layers. Let $\mathbf{z}_u$ be the final output of the convolution, we have $\mathbf{z}_u = \mathbf{z}_u^{(L)} \oplus \mathbf{z}_u^{(L-1)} \oplus \ldots \oplus \mathbf{z}_u^{(1)}$ where L is the number of layers. We used the Fermi-Dirac decoder (Krioukov et al., 2009; Nickel & Kiela, 2018).

GNN experimental setup We evaluate LRN on node classification and link prediction tasks, utilizing several datasets: (1) Homophilous graphs, including DISEASE (Chami et al., 2019), AIRPORT (Chami et al., 2019), and two citation networks, namely PUBMED (Sen et al., 2008) and CORA (Sen et al., 2008); (2) Heterophilic graphs, including CHAMELEON and SQUIRREL, with the splits in (Platonov et al., 2023). The hyperbolicity (δ) captures the degree of hyperbolic structure, with smaller values indicating a more hyperbolic structure (Chami et al., 2019).

Baselines For baselines, we test the effectiveness of our method against other hyperbolic residual methods, by apply-

Table 1: Test ROC AUC results (%) for Link Prediction (LP) and F1 scores (%) for Node Classification (NC). The best performance is highlighted in bold. Lower δ means the dataset is more tree-like.

	Homophilous								Heteophilic	
Dataset	DISEASE		AIRPORT		PBUMED		CORA		CHAMELEON	SQUIRREL
Hyperbolicity	$\delta = 0$		$\delta = 1$		$\delta = 3.5$		$\delta = 11$		$\delta = 2$	$\delta = 1.5$
Task	LP	NC	LP	NC	LP	NC	LP	NC	NC	NC
HyboNet	96.8 ± 0.4	$\mathbf{96.0} \pm 1.0$	$\mathbf{97.3} \pm 0.3$	90.0 ± 1.4	95.8 ± 0.2	78.0 ± 1.0	93.6 ± 0.3	$\mathbf{80.2} \pm 1.3$	40.1 ± 0.8	34.3 ± 0.5
Parallel Transport	86.4 ± 0.8	84.8 ± 3.7	93.6 ± 0.1	93.4 ± 0.6	$\mathbf{96.5} \pm 0.1$	77.8 ± 0.5	$\mathbf{94.8} \pm 0.3$	76.0 ± 0.8	36.6 ± 1.9	32.3 ± 0.8
Tangent Space	76.0 ± 2.4	91.9 ± 1.9	93.5 ± 0.1	92.0 ± 2.9	96.4 ± 0.2	76.8 ± 0.9	94.1 ± 0.3	79.2 ± 0.1	38.3 ± 0.8	34.0 ± 0.6
Space Addition	83.1 ± 1.2	88.9 ± 2.5	95.8 ± 0.3	90.0 ± 1.4	95.5 ± 0.2	75.9 ± 0.9	93.2 ± 0.2	78.6 ± 0.5	39.4 ± 2.0	34.5 ± 0.2
LRN (ours)	$\mathbf{97.3} \pm 0.4$	$\mathbf{96.1} \pm 1.0$	$\mathbf{97.3} \pm 0.3$	$\mathbf{93.9} \pm 0.7$	96.2 ± 0.1	$\mathbf{80.1} \pm 1.0$	94.1 ± 0.3	$\mathbf{80.6} \pm 0.9$	$\mathbf{41.1} \pm 0.9$	$\mathbf{37.1} \pm 1.1$

ing the baselines instead of LRN. We consider the base HyboNet (Chen et al., 2021) without residual connection, and previous baseline of the parallel transport method (Chami et al., 2019), the tangent space method (Chami et al., 2019), the space addition method (Bdeir et al., 2024).

GNN experimental results We show the results in Table 1. LRN is the best performer in 8 out of the 10 tasks, for up to 4.2%. Compared to the the base HyboNet model, LRN substantially outperforms in 9 out of the 10 tasks. Compare to the baseline residual connection methods, LRN is the best performer in 8 out of the 10 tasks, especially for the more difficult tasks of node classification on heterophilic datasets, demonstrating its effectiveness and generalizability to more difficult problem. LRN is also the best performer in every node prediction task, which benefits from deeper networks, demonstrating its superiority as a residual connection. In the more hyperbolic datasets, LRN always performs better and most of the time by large margins, suggesting that it is more suitable for hyperbolic networks as it doesn't map between hyperbolic and tangent (Euclidean) spaces.

Application of LRN to graph Transformers We test LRN as part of a hyperbolic adaptation of SGFormer (Wu et al., 2023), a recent Euclidean graph Transformer. We consider the same hyperbolic residual connection baselines as before. Following the notations (Wu et al., 2023), let $\mathbf{Z}^{(0)}$ be the input embedding, $\mathbf{Z}$ be the output of the global attention layer, and $\mathrm{GN}(\mathbf{Z}^{(\mathbf{0})}, \mathbf{A})$ be the output of the GNN layer with adjacency matrix $\mathbf{A}$. For LRN and the space addition method, we project $\mathbf{Z}$ and $\mathrm{GN}(\mathbf{Z}^{(0)}, \mathbf{A})$ to $\mathbb{L}_n^K$ and compute the final embedding as

$$(1-\alpha)\exp_{\mathbf{o}}^K(\mathbf{Z}) \oplus \alpha \exp_{\mathbf{o}}^K(\mathrm{GN}(\mathbf{Z}^{(\mathbf{0})}, \mathbf{A})), \quad (5)$$

where $\oplus$ denotes the respective residual connection method and α is a fixed weight. For the parallel transport and tangent space method, due to their dependence on the tangent space, we project weighted embeddings instead:

$$\exp_{\mathbf{o}}^K((1-\alpha)\mathbf{Z})) \oplus \exp_{\mathbf{o}}^K(\alpha \mathrm{GN}(\mathbf{Z}^{(\mathbf{0})}, \mathbf{A})). \quad (6)$$

We use the same Fermi-Dirac decoder (Krioukov et al., 2009; Nickel & Kiela, 2018) from earlier for the final output fully connected layer. For the datasets, we consider heteroliphic graphs CHAMELEON, SQUIRREL, and ACTOR. We use the same splits for ACTOR as (Lim et al., 2021).

Table 2: Accuracy% for node classification

Method	CHAMELEON	SQUIRREL	ACTOR
Hyperbolicity	$\delta = 2$	$\delta = 1.5$	$\delta = 1.5$
SGFormer	44.9 ± 3.9	41.8 ± 2.2	$\mathbf{37.9} \pm 1.1$
Parallel Transport	46.7 ± 1.2	38.5 ± 1.3	35.5 ± 0.9
Tangent Space	47.0 ± 0.8	42.1 ± 1.2	34.9 ± 0.7
Space Addition	47.2 ± 1.4	43.0 ± 1.1	35.3 ± 0.4
LRN(ours)	$\mathbf{47.8} \pm 1.3$	$\mathbf{43.9} \pm 0.8$	$\mathbf{38.0} \pm 0.4$

Table 3: Average runtime for 100 additions on random vectors

Method	$2,048/10,000$	$4,096/100,000$
Parallel Transport	0.0036s	1.448s
Tangent Space	0.0083s	3.601s
LRN (ours)	0.00025s	0.0006s

Transformer experimental results We show the results in Table 2. LRN outperforms the base Euclidean SGFormer and the baseline hyperbolic residual connection methods in all three cases, demonstrating the effectiveness of LRN in Transformer models. In 2 of the 3 datasets, the hyperbolic version of SGFormer almost always outperforms the Euclidean version as the high hyperbolicity suggests, justifying the application of hyperbolic modifications.

Runtime analysis We compare the efficiency of LRN with previous methods that involve multiple mappings by doing 100 additions on randomly generated vectors of dimension 2048 and size 10,000, and vectors of dimension 4096 and size 100,000 on a single RTX 3070 GPU, as shown in Table 3. Our method provides significant speedup against the baselines and has much better-scaling potential.

4. Conclusion

In this work, we introduced LRN, a new hyperbolic residual neural network based on the weighted Lorentzian centroid. LRN overcomes the limitations of previous methods, offering improved efficiency, flexibility, numerical stability, and retention of geometric information. Nevertheless, it should be noted that the performance of a residual-connected model is conditioned upon the performance of the base model. One direction of future is to apply LRN to newly developed architectures.

References

Atigh, M. G., Schoep, J., Acar, E., van Noord, N., and Mettes, P. Hyperbolic image segmentation. In *CVPR*, pp. 4453–4462, 2022.

Bdeir, A., Schwethelm, K., and Landwehr, N. Fully hyperbolic convolutional neural networks for computer vision. In *ICLR*, 2024.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *NeurIPS*, pp. 4868–4879, 2019.

Chen, B., Peng, W., Cao, X., and Röning, J. Hyperbolic uncertainty aware semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

Chen, W., Han, X., Lin, Y., Zhao, H., Liu, Z., Li, P., Sun, M., and Zhou, J. Fully hyperbolic neural networks. *arXiv preprint arXiv:2105.14686*, 2021.

Chen, Y., Yang, M., Zhang, Y., Zhao, M., Meng, Z., Hao, J., and King, I. Modeling scale-free graphs for knowledge-aware recommendation. *WSDM*, 2022.

Dai, J., Wu, Y., Gao, Z., and Jia, Y. A hyperbolic-to-hyperbolic graph convolutional network. *arXiv preprint arXiv:2104.06942*, pp. 154–163, 2021.

Desai, K., Nickel, M., Rajpurohit, T., Johnson, J., and Vedantam, S. R. Hyperbolic image-text representations. In *ICML*, pp. 7694–7731. PMLR, 2023.

Ermolov, A., Mirvakhabova, L., Khrulkov, V., Sebe, N., and Oseledets, I. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7409–7419, 2022.

Faqeeh, A., Osat, S., and Radicchi, F. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Physical review letters*, 121(9):098301, 2018.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *NeurIPS*, pp. 5345–5355, 2018.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., et al. Hyperbolic attention networks. In *ICLR*, 2019.

Guo, Y., Wang, X., Chen, Y., and Yu, S. X. Clipped hyperbolic classifiers are super-hyperbolic classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11–20, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Katsman, I., Chen, E., Holalkere, S., Asch, A., Lou, A., Lim, S. N., and De Sa, C. M. Riemannian residual neural networks. *Advances in Neural Information Processing Systems*, 36, 2023.

Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., and Lempitsky, V. Hyperbolic image embeddings. In *CVPR*, pp. 6418–6428, 2020.

Krioukov, D., Papadopoulos, F., Vahdat, A., and Boguná, M. Curvature and temperature of complex networks. *Physical Review E*, 80(3):035101, 2009.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguná, M. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Law, M., Liao, R., Snell, J., and Zemel, R. Lorentzian distance learning for hyperbolic representations. In *ICML*, pp. 3672–3681. PMLR, 2019.

Lim, D., Li, X., Hohne, F., and Lim, S.-N. New benchmarks for learning on non-homophilous graphs. *arXiv preprint arXiv:2104.01404*, 2021.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *NeurIPS*, pp. 8230–8241, 2019.

Mettes, P., Atigh, M. G., Keller-Ressel, M., Gu, J., and Yeung, S. Hyperbolic deep learning in computer vision: A survey. *arXiv preprint arXiv:2305.06611*, 2023.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*, pp. 6338–6347, 2017.

Nickel, M. and Kiela, D. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*, pp. 3779–3788, 2018.

Peng, W., Varanka, T., Mostafa, A., Shi, H., and Zhao, G. Hyperbolic deep neural networks: A survey. *TPAMI*, 2021.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

Ramsay, A. and Richtmyer, R. D. *Introduction to hyperbolic geometry*. Springer Science & Business Media, 2013.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-Scale Attributed Node Embedding. *Journal of Complex Networks*, 9(2), 2021.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pp. 355–366. Springer, 2011.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Shimizu, R., Mukuta, Y., and Harada, T. Hyperbolic neural networks++. In *ICLR*, 2020.

Sun, J., Cheng, Z., Zuberi, S., Pérez, F., and Volkovs, M. HGCF: Hyperbolic graph convolution networks for collaborative filtering. In *WebConf*, pp. 593–601, 2021.

Tang, J., Sun, J., Wang, C., and Yang, Z. Social influence analysis in large-scale networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

van Spengler, M., Berkhout, E., and Mettes, P. Poincar\'e resnet. *arXiv preprint arXiv:2303.14027*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Weng, Z., Ogut, M. G., Limonchik, S., and Yeung, S. Unsupervised discovery of the long-tail in instance segmentation using hierarchical self-supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2603–2612, 2021.

Wu, Q., Zhao, W., Yang, C., Zhang, H., Nie, F., Jiang, H., Bian, Y., and Yan, J. Sgformer: Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yang, M., Zhou, M., Kalander, M., Huang, Z., and King, I. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *KDD*, pp. 1975–1985, 2021.

Yang, M., Li, Z., Zhou, M., Liu, J., and King, I. Hicf: Hyperbolic informative collaborative filtering. In *KDD*, pp. 2212–2221, 2022a.

Yang, M., Zhou, M., Li, Z., Liu, J., Pan, L., Xiong, H., and King, I. Hyperbolic graph neural networks: A review of methods and applications. *arXiv preprint arXiv:2202.13852*, 2022b.

Yang, M., Zhou, M., Li, Z., Liu, J., Pan, L., Xiong, H., and King, I. Hyperbolic graph neural networks: a review of methods and applications. *arXiv preprint arXiv:2202.13852*, 2022c.

Yang, M., Zhou, M., Liu, J., Lian, D., and King, I. HRCF: Enhancing collaborative filtering via hyperbolic geometric regularization. In *WebConf*, 2022d.

Yang, M., Zhou, M., Xiong, H., and King, I. Hyperbolic temporal network embedding. *TKDE*, 2022e.

Yang, M., Zhou, M., Ying, R., Chen, Y., and King, I. Hyperbolic representation learning: Revisiting and advancing. *arXiv preprint arXiv:2306.09118*, 2023.

Yu, T. and De Sa, C. M. Numerically accurate hyperbolic embeddings using tiling-based models. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Oct. 2019.

Zhang, Y., Wang, X., Shi, C., Jiang, X., and Ye, Y. F. Hyperbolic graph attention network. *IEEE Transactions on Big Data (TBD)*, 2021a.

Zhang, Y., Wang, X., Shi, C., Liu, N., and Song, G. Lorentzian graph convolutional networks. In *WebConf*, pp. 1249–1261, 2021b.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. *ICLR*, 2019.

A. Related Works

Hyperbolic representation and deep learning Hyperbolic spaces, as a typical example of non-Euclidean geometries, have garnered extensive attention in recent years for hyperbolic representation learning and hyperbolic deep learning (Peng et al., 2021; Mettes et al., 2023; Yang et al., 2022b). A defining geometric characteristic of hyperbolic spaces is their negative curvature, which results in an exponential growth of volume with distance from the origin. This property closely mirrors the structure of tree-like data, where the number of child nodes increases exponentially (Krioukov et al., 2010). Consequently, hyperbolic representation learning provides a strong geometric prior for hierarchical structures, tree-like data, and power-law distributed information. Significant advancements have been achieved in various domains such as WordNet (Nickel & Kiela, 2017; 2018), graphs (Chami et al., 2019; Liu et al., 2019; Zhang et al., 2021a; Yang et al., 2023), social networks (Yang et al., 2021), and recommendation systems (Yang et al., 2022d;a; Chen et al., 2022; Sun et al., 2021) utilizing hyperbolic representations. Moreover, hyperbolic deep learning has demonstrated impressive performance in image-related tasks, including image embedding (Khrulkov et al., 2020; Guo et al., 2022; Ermolov et al., 2022) and segmentation (Atigh et al., 2022; Weng et al., 2021; Chen et al., 2023), offering new insights into deep learning paradigms, such as the interpretation of norms as reflections of uncertainty. Within the neural network domain, HNN (Ganea et al., 2018) presented the pioneering form of hyperbolic neural networks, defining fundamental hyperbolic transformations, classifiers, and hyperbolic multiclass logistic regression (MLR). HNN++ (Shimizu et al., 2020) further reduced the parameter count of hyperbolic MLR and made advancements in fully-connected layers, the splitting and concatenation of coordinates, convolutional layers, and attention mechanisms within hyperbolic space. Recent studies, such as HyboNet (Chen et al., 2021), propose frameworks that construct neural networks entirely within hyperbolic space, contrasting with existing models that partially operate in Euclidean tangent spaces.

Residual neural networks and hyperbolic adaptations Residual connections represent a fundamental module in modern neural networks (He et al., 2016), addressing the vanishing gradient problem directly and enabling the training of significantly deeper networks. By supporting identity mapping through skip connections, they enhance gradient flow across layers, thereby improving training efficiency and model performance across a diverse set of tasks.

Several works have explored the adaptation of residual connections to hyperbolic spaces. Poincaré ResNet (van Spengler et al., 2023) leverages the concept of parallel transport to map points in hyperbolic space to the tangent space at a reference point and then, via parallel transport, to the tangent space of a new point, subsequently mapping it back using the exponential map. Similarly, Riemannian ResNet (Katsman et al., 2023) proposes an analogous method to perform this operation. (Nickel & Kiela, 2018) proposes mapping the hyperbolic embeddings to the tangent space of the origin, summing them, and then projecting the sum back into hyperbolic space. HCNN (Bdeir et al., 2024) proposes to add the space components and uses post-summation processing on the time component.

B. Preliminaries

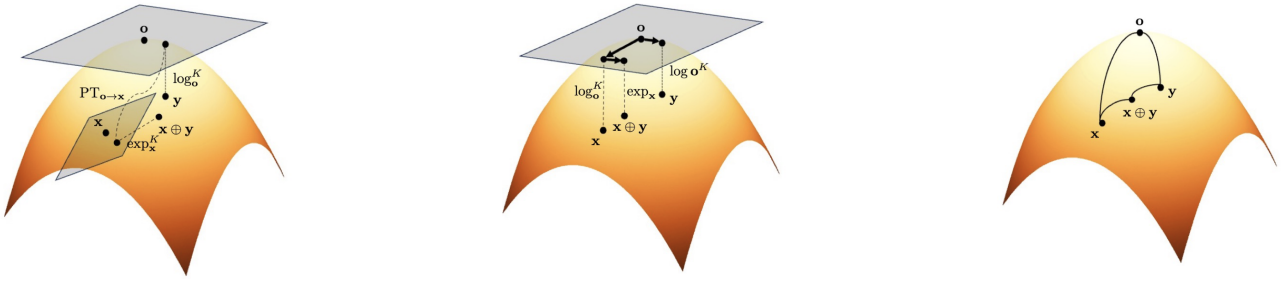
This section provides an overview of the fundamental concepts in hyperbolic geometry, focusing on the Lorentz model. Hyperbolic space can be formulated using several models, including the Poincaré ball model (Nickel & Kiela, 2017), the Lorentz (Hyperboloid) model (Nickel & Kiela, 2018), and the Klein model (Gulcehre et al., 2019). These models are isometric, meaning that points in one model can be smoothly mapped to points in another while preserving distances, angles and geodesics (Ramsay & Richtmyer, 2013).

Lorentz model An n-dimensional Lorentz model is a Riemannian manifold equipped with the Riemannian metric tensor and defined by a constant negative curvature $K < 0$, denoted as $\mathbb{L}_K^n$. Each point $\mathbf{x} \in \mathbb{L}_K^n$ has the form $[x_t, \mathbf{x}_s]$ where $x_t \in \mathbb{R}$ is called the time-like component and $\mathbf{x}_s \in \mathbb{R}^n$ is called the space-like component. $\mathbb{L}_K^n$ is equipped with the *Lorentzian inner product*. For points $\mathbf{x}, \mathbf{y} \in \mathbb{L}_K^n$, their inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}$ is given by

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_t y_t + \mathbf{x}_s^T \mathbf{y}_s = \mathbf{x}^T \mathfrak{g}_n^K \mathbf{y}, \tag{7}$$

with $||\mathbf{x}||_{\mathcal{L}} \coloneqq \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}}}$ being the Lorentzian norm. Formally, $\mathcal{L}^n$ is the point set $\mathcal{L}^n = \{\mathbf{x} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = 1/K, x_t > 0\}$. The origin $\mathbf{o} \in \mathbb{L}_K^n$ is the point $[\sqrt{-1/K}, 0, \ldots, 0]^T$.

Tangent space The tangent space at a point $\mathbf{x} \in \mathbb{L}_K^n$ is the Euclidean space of all points orthogonal to $\mathbf{x}$. Formally it is defined as $\mathcal{T}_{\mathbf{x}} \mathbb{L}_K^n = \{\mathbf{y} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = 0\}$.



(a) Parallel Transport (b) Tangent Space (c) LRN

Figure 1: Visualization of hyperbolic addition methods. From left to right: (a) Parallel transport method, (b) Tangent space method, and (c) LRN. In each subfigure, $\mathbf{x} \oplus \mathbf{y}$ represents the sum of points $\mathbf{x}, \mathbf{y} \in \mathbb{L}_K^n$. Dotted lines are exponential and logarithmic maps, while solid lines are geodesics.

Exponential and logarithmic maps For each point $\mathbf{x} \in \mathbb{L}_K^n$, the exponential map $\exp_{\mathbf{x}}^K : \mathcal{T}_{\mathbf{x}}\mathbb{L}_K^n \to \mathbb{L}_K^n$ and the logarithmic map $\log_{\mathbf{x}}^K : \mathbb{L}_K^n \to \mathcal{T}_{\mathbf{x}}\mathbb{L}_K^n$ at $\mathbf{x}$ are given by

$$\exp_{\mathbf{x}}^K(\mathbf{y}) = \cosh(\alpha)\mathbf{x} + \frac{\sinh(\alpha)}{\alpha}\mathbf{y}, \alpha = \sqrt{-K\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}}. \tag{8}$$

$$\log_{\mathbf{x}}^K(\mathbf{x}) = \frac{\cosh^{-1}(\beta)}{\sqrt{\beta^2 - 1}}(\mathbf{y} - \beta\mathbf{x}), \text{where } \beta = K\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}. \tag{9}$$

Parallel transport Parallel transport is a generalization of translation to hyperbolic geometry mapping a point $\mathbf{z} \in \mathcal{T}_{\mathbf{x}}\mathbb{L}_K^n$ to a point in $\mathcal{T}_{\mathbf{y}}\mathbb{L}_K^n$ via

$$\mathbf{P}_{\mathbf{x}\to\mathbf{y}}(\mathbf{z}) = \mathbf{z} + \frac{\langle \mathbf{y}, \mathbf{z}\rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}}(\mathbf{x} + \mathbf{y}).$$

Hyperbolic addition methods Several methods have been proposed for performing addition in hyperbolic space. The parallel transport method of Mobius Addition from (Chami et al., 2019; Ganea et al., 2018) is given by

$$\mathbf{x} \oplus_P \mathbf{y} = \exp_{\mathbf{x}}^K \circ \mathbf{P}_{\mathbf{o}\to\mathbf{x}} \circ \log_{\mathbf{o}}^K(\mathbf{y}). \tag{10}$$

Another approach is the tangent space addition method used for aggregation in (Chami et al., 2019), given as:

$$\mathbf{x} \oplus_T \mathbf{y} = \exp_{\mathbf{o}}^K\left(w_x \log_{\mathbf{o}}^K(\mathbf{x}) + w_y \log_{\mathbf{o}}^K(\mathbf{y})\right). \tag{11}$$

where $w_x, w_y > 0$ are weights. A third approach is the space addition method from (Bdeir et al., 2024) given by

$$\mathbf{x} \oplus_S \mathbf{y} = \left[\sqrt{||(\mathbf{x} + \mathbf{y})_s||^2 - \frac{1}{K}}, \mathbf{x} + \mathbf{y}_s\right]^T \tag{12}$$

The methods are shown in Figure 1, including LRN. The space addition method is not shown since it does not have geometric interpretation on the manifold.

C. Further Experiments

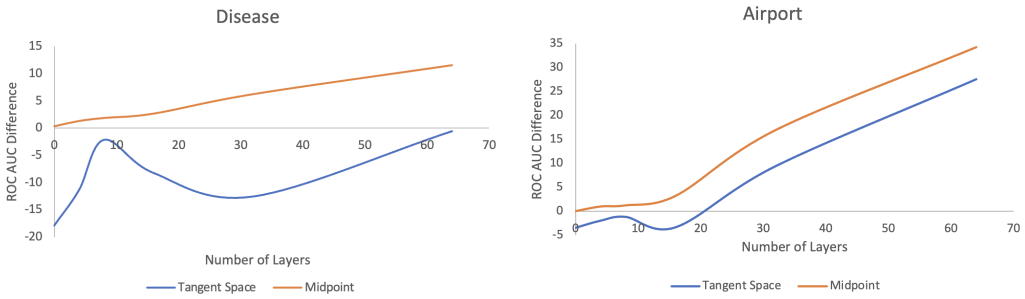
C.1. Further analysis and experiments

Effects on graph over-smoothing problem Previous work has found that stacking graph convolutional layers often results in a significant drop in performance due to gradient vanishing or over-smoothing (Zhao & Akoglu, 2019). Traditionally incorporating residual connection helps with these problems in Euclidean space. To test the effects of LRN on this problem in hyperbolic space, we run the GCN architecture from Section 3 on DISEASE and AIRPORT datasets for 4, 8, 16, 32, and 64 layers with the tangent space method as a baseline. The results are shown in Figure 2, as the difference between the ROC

Table 4: AUC ROC score(%) of both directions of the parallel transport method.

Datasets	Disease($\delta = 0$)	Airport($\delta = 1$)
Task	LP	LP
Forward	**86.8**±0.8	93.6±0.1
Backward	67.7±1.4	**93.7**±0.1

Figure 2: The ROC AUC score (%) difference between skip-connected HypoNet and base HypoNet



(a) Disease Many Layers

(b) Airport Many Layers

AUC score of the skip-connected HyboNets and the base HyboNet. The difference between HyboNet and LRN always increases as the number of layers increases, showing the effect of our method on mitigating the performance drops due to over-smoothing. The tangent space method sees a drop in the difference for both datasets, at 32 layers for DISEASE and at 16 layers for AIRPORT. This shows the stability of LRN over deep neural networks. The parallel transport method results in NaN values for 16 or more layers and is therefore not included.

Effects of commutativity of residual connection. To investigate the effect of commutativity of a residual connection, we test the accuracy of both directions of the parallel transport method on DISEASE and AIRPORT datasets using the architecture in Section 3. Forward operation represents $\mathbf{x} \oplus_P f(\mathbf{x})$ and backward operation represents $f(\mathbf{x}) \oplus_P \mathbf{x}$, where f is the convolutional layer. The results are shown in Table 4. The forward method significantly outperforms the backward method in link prediction tasks of the DISEASE dataset, while the backward method outperforms the forward method for the AIRPORT dataset. This shows the limitation in the flexibility of the parallel transport method due to its non-commutativity, as it is unpredictable which direction of addition would have the better performance.

D. Proof of theoretical results

PROOF OF LEMMA 2.1

Proof. Note that

$$
\begin{aligned}
& - K||w_x\mathbf{x} + w_y\mathbf{y}||^2_{\mathcal{L}} \\
&= -K\left(-(w_x x_t + w_y y_t)^2 + ||w_x\mathbf{x}_s + w_y\mathbf{y}_s||^2\right) \\
&= K||w_x\mathbf{x}||^2_{\mathcal{L}} + K||w_y\mathbf{y}||^2_{\mathcal{L}} - 2K\langle w_x\mathbf{x}, w_y\mathbf{y}\rangle_{\mathcal{L}} \\
&= w_x^2 + w_y^2 + 2K\langle w_x\mathbf{x}, w_y\mathbf{y}\rangle_{\mathcal{L}} \\
&= w_x^2 + w_y^2 - 2K\Gamma \\
&> w_x^2 + w_y^2 - 2K(w_x w_y||\mathbf{x}_s||||\mathbf{y}_s|| - w_x w_y\langle\mathbf{x}_s.\mathbf{y}_s\rangle) \\
&\geq w_x^2 + w_y^2 \qquad \text{(Cauchy Schwarz Inequality)}
\end{aligned}
$$

where

$$\begin{aligned}\Gamma = w_x\sqrt{\|\mathbf{x}_s\|^2 - \frac{1}{K}} \cdot w_y\sqrt{||\mathbf{y}_s||^2 - \frac{1}{K}} \\ - w_x w_x \langle \mathbf{x}_s, \mathbf{y}_s \rangle\end{aligned} \tag{13}$$

By taking the square-roots we have the desired inequality of $\sqrt{-K}\,|\|w_x\mathbf{x} + w_y\mathbf{y}\|_{\mathcal{L}}| > \sqrt{w_x^2 + w_y^2}$. □

PROOF OF THEOREM 2.2

Proof. Let $\mathbf{x} = [a_1, a_2, \ldots, -a_{n+1}] \in \mathbb{L}_K^n$ and $\mathbf{y} = [a_1, a_2, \ldots, a_{n+1}] \in \mathbb{L}_K^n$ where each $a_i \in \mathbb{R}$ and $a_1 > 0$. We can easily compute (by following the computation in Proposition 2.3) that we have

$$\mathbf{z} = \mathbf{x} \oplus \mathbf{y} = \cosh(\alpha)\mathbf{x} + \frac{\sinh(\alpha)}{\alpha}(c_u\mathbf{y}' + c_v\mathbf{x}'),$$

where

$$\begin{aligned}&\mathbf{y}' = \mathbf{y} + y_t\sqrt{-K}\mathbf{o}, \mathbf{x}' = \mathbf{x} + \mathbf{o}, \\ &c_u = \frac{\cosh^{-1}(y_t\sqrt{-K})}{\sqrt{-y_t^2K - 1}}, \mathbf{u} = c_u \cdot \mathbf{y}' \\ &c_v = \frac{\langle \mathbf{x}, \mathbf{u}\rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{o}, \mathbf{x}\rangle_{\mathcal{L}}}, \mathbf{v} = c_u \cdot \mathbf{y}' + c_v \cdot \mathbf{x}' \\ &\alpha = \sqrt{-K}||\mathbf{v}||_{\mathcal{L}}.\end{aligned}$$

Similarly, we can compute that for the other direction of parallel transport, we have

$$\mathbf{z}' = \mathbf{y} \oplus \mathbf{x} = \cosh(\alpha')\mathbf{y} + \frac{\sinh(\alpha')}{\alpha'}(c_u'\mathbf{p} + c_v'\mathbf{q}),$$

where

$$\begin{aligned}&\mathbf{p} = \mathbf{x} + x_t\sqrt{-K}\mathbf{o}, \mathbf{q} = \mathbf{y} + \mathbf{o} \\ &c_u' = \frac{\cosh^{-1}(x_t\sqrt{-K})}{\sqrt{-x_t^2K - 1}}, \mathbf{u}' = c_u' \cdot \mathbf{p} \\ &c_v' = \frac{\langle \mathbf{y}, \mathbf{u}'\rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{o}, \mathbf{y}\rangle_{\mathcal{L}}}, \mathbf{v}' = c_u' \cdot \mathbf{p} + c_v' \cdot \mathbf{q} \\ &\alpha' = \sqrt{-K}||\mathbf{v}'||_{\mathcal{L}}.\end{aligned}$$

By analyzing the symmetry in the construction of $\mathbf{x}, \mathbf{y}, \mathbf{y}', \mathbf{p}, \mathbf{u}, \mathbf{u}', \mathbf{v}$, and $\mathbf{v}'$, it can be shown that $c_u = c_u'$ and $c_v = c_v'$ due to the identical nature of the transformations applied in either direction. This implies that $\alpha = \alpha'$, leading to the conclusion that $\mathbf{z}_{n+1} = -\mathbf{z}'_{n+1}$ as the operations in the parallel transport in both directions result in vectors whose $(n+1)$-th components are negations of each other. In the following, we give the detailed derivation.

First, since $x_t = y_t$, we have $c_u = c_u'$. Let $c_u = c_u' = \beta$, then we have,

$$\begin{aligned}\mathbf{y}' = \mathbf{y} + y_t\sqrt{-K}\mathbf{o} &= \mathbf{y} + a_1 \cdot \sqrt{-K}\left[\sqrt{-1/K}, 0, \ldots, 0\right]^T \\ &= [2a_1, a_2, \ldots, a_{n+1}]^T\end{aligned}$$

and

$$\begin{aligned}\mathbf{p} = \mathbf{x} + x_t\sqrt{-K}\mathbf{o} &= \mathbf{x} + a_1 \cdot \sqrt{-K}\left[\sqrt{-1/K}, 0, \ldots, 0\right]^T \\ &= [2a_1, a_2, \ldots, -a_{n+1}]^T.\end{aligned}$$

So $\mathbf{u} = \beta\mathbf{y}' = [2\beta a_1, \beta a_2, \ldots, \beta a_{n+1}]^T$ and $\mathbf{u}' = \beta\mathbf{p} = [2\beta a_1, \beta a_2, \ldots, -\beta a_{n+1}]$. So we can compute that

$$\langle \mathbf{x}, \mathbf{u} \rangle_{\mathcal{L}} = \left[-2\beta a_1^2 - \beta a_{n+1}^2 + \sum_{i=2}^{n} \beta a_i^2 \right]^T$$
$$\langle \mathbf{y}, \mathbf{u}' \rangle_{\mathcal{L}} = \left[-2\beta a_1^2 - \beta a_{n+1}^2 + \sum_{i=2}^{n} \beta a_i^2 \right]^T.$$

So we have

$$\begin{aligned} c_v &= \frac{\langle \mathbf{x}, \mathbf{u} \rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{o}, \mathbf{x} \rangle_{\mathcal{L}}} \\ &= \frac{\left[-2\beta a_1^2 - \beta a_{n+1}^2 + \sum_{i=2}^{n} \beta a_i^2 \right]^T}{-1/K - \sqrt{-1/K} a_1} \end{aligned}$$

and

$$\begin{aligned} c'_v &= \frac{\langle \mathbf{y}, \mathbf{u}' \rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{o}, \mathbf{y} \rangle_{\mathcal{L}}} \\ &= \frac{\left[-2\beta a_1^2 - \beta a_{n+1}^2 + \sum_{i=2}^{n} \beta a_i^2 \right]^T}{-1/K - \sqrt{-1/K} a_1}. \end{aligned}$$

So we have $c_v = c'_v$, call this constant γ. Now, we can compute further that

$$\begin{aligned} \mathbf{v} &= \beta\mathbf{y}' + \gamma\mathbf{x}' \\ &= \beta[2a_1, a_2, \ldots, a_{n+1}]^T \\ &+ \gamma \left[(a_1 + \sqrt{-1/K}), a_2, \ldots, a_{n+1} \right]^T \\ &= \left[2\beta a_1 + \gamma(a_1 + \sqrt{-1/K}), \ldots, (\beta + \gamma) a_{n+1} \right]^T. \end{aligned}$$

and

$$\begin{aligned} \mathbf{v}' &= \beta\mathbf{p} + \gamma\mathbf{q} \\ &= [2\beta a_1, \beta a_2, \ldots, -\beta a_{n+1}]^T \\ &+ \left[\gamma(a_1 + \sqrt{-1/K}), \gamma a_2, \ldots, -\gamma a_{n+1} \right]^T \\ &= \left[2\beta a_1 + \gamma(a_1 + \sqrt{-1/K}), \ldots, -(\beta + \gamma) a_{n+1} \right]^T. \end{aligned}$$

So we have $||\mathbf{v}||_{\mathcal{L}} = ||\mathbf{v}'||_{\mathcal{L}}$, and thus $\alpha = \alpha'$, let us call this constant simply α. Finally:

$$\mathbf{z}_{n+1} = \cosh(\alpha)(-a_{n+1}) + \frac{\sinh(\alpha)}{\alpha}(\beta a_{n+1} - \delta a_{n+1})$$
$$\mathbf{z}'_{n+1} = \cosh(\alpha)(a_{n+1}) + \frac{\sinh(\alpha)}{\alpha}(\beta(-a_{n+1}) + \delta a_{n+1}).$$

Hence, $\mathbf{z}_{n+1} = -\mathbf{z}'_{n+1}$. □

Proof of Proposition 2.3

Proof. a. For **parallel transport method**: The isometry from Lorentz model to Klein model is given by the map

$$\varphi_K(\mathbf{x}) = \mathbf{x}_t / x_s$$

. Thus given the fact that geodesics in the Klein model are Euclidean straight lines, it suffices to show that $\mathbf{z}_s = \lambda \mathbf{m}_s$ for some $\lambda \in \mathbb{R}$. Now we can compute:

$$\begin{aligned}
\mathbf{u} &= \log_{\mathbf{o}}^K(\mathbf{y}) \\
&= \frac{\cosh^{-1}\left(y_t\sqrt{-K}\right)}{\sqrt{-y_t^2K-1}}(\mathbf{y} + y_t\sqrt{-K}\mathbf{o}) \\
&= c_u \cdot \mathbf{y}'
\end{aligned}$$

$$\begin{aligned}
\mathbf{v} &= \mathbf{P}_{\mathbf{o}\to\mathbf{x}}(\mathbf{u}) \\
&= \mathbf{u} + \frac{\langle \mathbf{x}, \mathbf{u}\rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{o}, \mathbf{x}\rangle_{\mathcal{L}}}(\mathbf{o} + \mathbf{x}) \\
&= c_u \cdot \mathbf{y}' + c_v \cdot \mathbf{x}'
\end{aligned}$$

$$\begin{aligned}
\mathbf{z} &= \cosh(\sqrt{-K}||\mathbf{v}||_{\mathcal{L}})\mathbf{x} + \frac{\sinh(\sqrt{-K}||\mathbf{v}||_{\mathcal{L}})}{\sqrt{-K}||\mathbf{v}||_{\mathcal{L}}}\mathbf{v} \\
&= \cosh(\alpha)\mathbf{x} + \frac{\sinh(\alpha)}{\alpha}(c_u\mathbf{y}' + c_v\mathbf{x}')
\end{aligned}$$

where $c_u = \frac{\cosh^{-1}\left(y_t\sqrt{-K}\right)}{\sqrt{-y_t^2K-1}}$, $c_v = \frac{\langle \mathbf{x}, \mathbf{u}\rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{o}, \mathbf{x}\rangle_{\mathcal{L}}}, \mathbf{y}' = \mathbf{y} + y_t\sqrt{-K}\mathbf{o}, \mathbf{x}' = \mathbf{o} + \mathbf{x}, \alpha = \sqrt{-K}||\mathbf{v}||_{\mathcal{L}}$. Clearly $\mathbf{y}'_s = \mathbf{y}_s, \mathbf{x}'_s = \mathbf{x}_s$. So $w_x = \cosh(\alpha) + c_v$ and $w_y = \frac{\sinh(\alpha)}{\alpha}c_u$ gives the desired result.

b. For **tangent space method**: One can check that

$$\begin{aligned}
&\exp_{\mathbf{o}}^K(\log_{\mathbf{o}}^K(\mathbf{x}) + \log_{\mathbf{o}}^K(\mathbf{y}))_s \\
&= \frac{\sinh(\sqrt{-K}||c_1\mathbf{x}' + c_2\mathbf{y}'||_{\mathcal{L}})}{\sqrt{-K}||c_1\mathbf{x}' + c_2\mathbf{y}'||_{\mathcal{L}}}\left(c_1\mathbf{x}' + c_2\mathbf{y}'\right)_s
\end{aligned}$$

with $c_1 = \frac{\cosh^{-1}\left(-x_t\sqrt{-K}\right)}{\sqrt{x_t^2K-1}}, c_2 = \frac{\cosh^{-1}\left(-y_t\sqrt{-K}\right)}{\sqrt{y_t^2K-1}}, \mathbf{x}' = \mathbf{x} + x_t\sqrt{-K}\mathbf{o}, \mathbf{y}' = \mathbf{y} + y_t\sqrt{-K}\mathbf{o}$. Again $\mathbf{x}_s = \mathbf{x}'_s, \mathbf{y}_x = \mathbf{y}'_s$ So one can pick $w_x = c_1, w_y = c_2$.

c. For **space addition method**: This follows immediately from the isometry to Klein model.

□

E. Dataset details

E.1. Graph datasets

Table 5 provides a detailed breakdown of the main proprieties of the graph datasets we used. The datasets are all publicly available.

- CORA and PUBMED (Sen et al., 2008) are citation networks, where each node presents a document and each edge represents a citation link. Each node has a label consisting of the academic subarea the document is in.
- AIRPORT(Chami et al., 2019) is a dataset where each node represents an airport and each edge represents an airline route. The node features are geogrphic information.

Table 5: Graph Datasets

Dataset	Property	#Nodes	#Edges	#Feats	#Classes
Cora	homophilous	2,708	5,429	1,433	7
PubMed	homophilous	19,717	44,324	500	3
Airport	homophilous	3,188	18,631	4	4
Disease	homophilous	1,044	1,043	1000	2
Squirrel	heterophilic	5,201	216,933	2.089	5
Chameleon	heterophilic	2,277	36,101	2,325	5
Actor	heterophilic	7,600	29,926	931	5

- DIEASE(Chami et al., 2019) is a synthetic dataset generated by simulating the SIR disease spreading model, where the node features represent a node's susceptibility to the disease and node labels are whether a node was infected or not.
- CHAMELEON and SQUIRREL (Rozemberczki et al., 2021) are page-page networks for specific topics in Wikipedia. Each node represents a web page and each edge represents a mutual link between the pages. The node features consist of a selection of informative nouns extracted from the corresponding Wikipedia pages.
- ACTOR(Tang et al., 2009) is a dataset where each node represents an actor and each edge represents whether two actors co-occur on the same Wikipedia page. The node features consist of keywords extracted from the respective actor's Wikipedia pages.

For the homophilous datasets, we used the same splits as in HGCN (Chami et al., 2019). For the heterophilic datasets, we use the same splits as in SFGormer (Wu et al., 2023), where the splits for CHAMELEON and SQUIRREL are from (Platonov et al., 2023), and the splits for ACTOR are from (Lim et al., 2021).

F. Implementation and training details

F.1. Implementation details for graph datasets

F.1.1. GNN IMPLEMENTATION

We closely follow the implementation in the base HyboNet model (Chen et al., 2021), using the fully hyperbolic linear and aggregation layers they proposed. For the residual connection architecture, we adopt the skip-connected GCN architecture from (Sun et al., 2021), where the output of each intermediate layer was added to the output of the next layer via LRN (or a baseline hyperbolic residual connection method). We employ the same Fermi-Dirac decoder as used in (Chen et al., 2021). As for optimizers, we used the same setup as in (Chen et al., 2021) as well, where the parameters were separated into two groups depending on whether it is a hyperbolic parameter.

F.1.2. GNN HYPERPARAMETERS

For the homophilous graphs datasets, we used the largely the same hyperparmeters as in (Chen et al., 2021), including dropout rate, learning rate, weight decay rate, gradient clip value, and margins for the marginal loss. For link prediction tasks, we run the model for 3 layers. For node classification tasks, we performed a search for the number of layers on the set $\{3, 4, 5, 6, 7, 8\}$. For all tasks, we performed a search for the constant negative curvature K on the set $\{-0.1, -0.5, -1.0, -1.5, -2.0, \text{trainable}\}$, where the initial value of the trainable curvature is -1.0.

For heteophilic graph datasets, we used a curvature of -1 and performed a grid search in the following search space:

- dimension within {16, 32, 64}
- learning rate within {0.001, 0.01}
- dropout rate within {0, 0.1, 0.2, 0.3}
- weight-decay rate within {0, 1e-4, 1e-3}

- number of layers within {3, 4, 5, 6, 7, 8}

For homophilous graph datasets, we used a constant weight of 1 for LRN shown in Equation (1). For heterophilic datasets, we used trainable weights instead.

F.1.3. HYPERBOLIC SGFORMER IMPLEMENTATION

We closely followed the setup of the base SGFormer model (Wu et al., 2023) and the hyperbolic adaptation we outlined in Section 3. We use the Fermi-Dirac decoder that we used in the GNN implementation, adopted from (Chen et al., 2021). For optimizer, we utilized two separate optimizer: the Euclidean Adam optimizer for non-manifold parameters and the Riemannian Adam optimizer for manifold parameters. We used trainable weights for the LRN method shown in Equation (1).

F.1.4. HYPERBOLIC SGFOERMER HYPERPARAMETERS

We performed a grid search in the following search space with a constant curvature of -1.0:

- dimensions within {32, 64} for CHAMELEON and SQUIRREL, fixed dimension of 96 for FILM
- learning rate within {0.001, 0.01}
- weight-decay rate within {0, 1e-4, 1e-3}
- dropout rate within {0.3, 0.4, 0.5, 0.6}
- number of layers within {2, 3, 4, 5,6, 7, 8}

Here the number of layers refers to the GCN layer utilized in (Wu et al., 2023).

F.2. Further details and code

All experiments were performed on float32 datatype and on a single NVIDIA RTX 3070 GPU with 8gb of memory. For more details regarding our implementations, please see the accompanying GitHub code.