Lorentzian Residual Neural Networks

Neil He
neilhe6345@gmail.com
Yale University
New Haven, United States

Menglin Yang
mlyang.yale@outlook.com
Yale University
New Haven, United States

Rex Ying
rex.ying@yale.edu
Yale University
New Haven, United States

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 direct flow of information across layers, have been instrumental in the success of deep neural networks. However, current methods for constructing hyperbolic residual networks 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 LResNet, a novel Lorentzian residual neural network based on the *weighted Lorentzian centroid in the Lorentz model of hyperbolic space*. Our method enables the efficient integration of residual connections in Lorentz hyperbolic neural networks while preserving their hierarchical representation capabilities. We demonstrate that our method can theoretically derive previous methods while offering improved stability, efficiency, and effectiveness. Extensive experiments on both graph and vision tasks showcase the superior performance and robustness of our method compared to state-of-the-art Euclidean and hyperbolic alternatives. Our findings highlight the potential of LResNet for building more expressive neural networks in hyperbolic embedding space as a generally applicable method to multiple architectures, including CNNs, GNNs, and graph Transformers.

CCS Concepts

• **Computing methodologies → Machine learning**; **Knowledge representation and reasoning**; • **Mathematics of computing → Geometric topology**.

Keywords

Residual Neural Networks, Hyperbolic Geometry, Deep Neural Networks, Foundation Model

ACM Reference Format:
Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In . ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

Corresponding author: Menglin Yang
https://github.com/Graph-and-Geometric-Learning/LResNet.

1 Introduction

In recent years, exploration of neural network architectures beyond traditional Euclidean space has opened up new frontiers in machine learning research [26, 30, 47]. Among these, hyperbolic neural networks [2, 3, 5, 12, 24, 35, 39] have garnered significant attention due to their inherent capabilities to model data with complex hierarchical structures. Hyperbolic spaces, characterized by their constant negative curvature that allows for exponential growth of volume, naturally align with the geometric properties of tree-like data, offering a more fitting representation than their Euclidean counterparts that suffer from embedding distortions [19, 33]. This alignment has the potential to enhance learning efficiency and improve representation learning for a wide range of applications, from graph-based data analysis [3, 11, 24, 37, 44, 48, 49] to image understanding [4, 10, 42, 52].

One of the core elements of the modern deep learning framework is the residual connection [16], 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, which addresses issues such as the gradient vanishing and graph over-smoothing problems, residual connections facilitate the training of substantially deeper networks and are widely used in many model architectures, including CNNs, GNNs, transformers, and diffusion models [16, 36, 40, 41]. However, the application of residual connections within Euclidean spaces is not directly transferable to the complex geometry of hyperbolic spaces. This is primarily due to the curvature of hyperbolic space, where direct addition often violates geometric constraints. In the Poincaré ball model, the sum could exceed the sphere's boundary [38]. In the Lorentz model, it could deviate from the hyperboloid planes [32].

Existing works and their limitations. Several previous works have explored implementing residual connections in hyperbolic space. Poincaré ResNet [39] 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 as shown in Figure 1a. Another similar work, Riemannian ResNet [17], although not confined just in hyperbolic space, employs similar concepts for an analogous method. As a method previously used as hyperbolic addition for aggregation, HGCN [3] and LGCN [54] propose first mapping to the tangent space of the origin for addition, and then projecting the sum back into hyperbolic space, as shown in Figure 1b. HCNN [2] proposes adding the space-like component of the hyperbolic embeddings and then computing the time-like component afterwards. Each of these previous methods suffer from at least one of the limitations listed below:

(i) Computationally Expensive. The parallel transport and tangent space addition methods involve complex mappings to

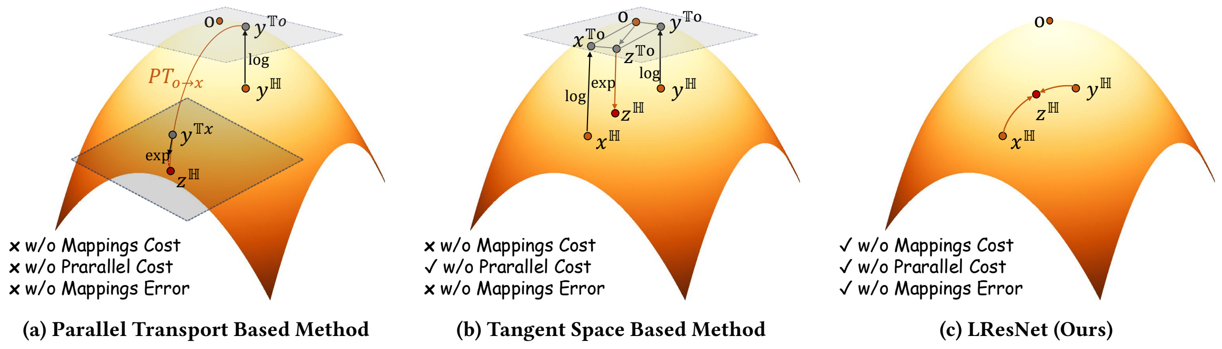


Figure 1: Visualization of hyperbolic residual connection methods. From left to right: (a) Parallel transport-based method, (b) Tangent space-based method, and (c) The proposed LResNet. Points with superscript $\mathbb{H}$ and $\mathbb{T}$ indicate their presence in hyperbolic space and tangent space, respectively. In each subfigure, $z^{\mathbb{H}}$ represents the sum of points $x^{\mathbb{H}}, y^{\mathbb{H}} \in \mathbb{H}$. PT denotes parallel Transport, and log and exp denotes the logarithmic and exponential mappings respectively. Our proposed method LResNet overcomes limitations L(i, ii, iii, iv) by eliminating mappings and parallel transport (whose absence is shown via ✓), where the other two methods depend on as least one (shown via ×).

and from the tangent space, significantly increasing computational complexity (see Table 6 for runtime comparison).

(ii) Non-Commutativity. The Euclidean residual connection proposed in [16] satisfies $\mathbf{x} + f(\mathbf{x}) = f(\mathbf{x}) + \mathbf{x}$, while the parallel transport-based residual connection is non-commutative. This greatly restricting the expressiveness and flexibility of the model (see Section 5.4).

(iii) Numerical Instability. The parallel transport and tangent space addition methods are based on numerically unstable operations. For curvature -1, the geodesic distance in the n dimensional Poincaré ball model $\mathcal{P}^n$ and the logarithmic map in the n dimensional Lorentz model $\mathcal{L}^n$ are defined by the following formulas, respectively:

$$d(\mathbf{x},\mathbf{y}) = \cosh^{-1}\left(1 + 2\frac{\|\mathbf{x}-\mathbf{y}\|^2}{(1-\|\mathbf{x}\|^2)(1-\|\mathbf{y}\|^2)}\right),$$
$$\log_{\mathbf{u}}(\mathbf{v}) = \frac{\cosh^{-1}(\alpha)}{\sqrt{\alpha^2-1}}(\mathbf{v}-\alpha\mathbf{u}), \tag{1}$$

where $\mathbf{x},\mathbf{y} \in \mathcal{P}^n$, $\mathbf{u},\mathbf{v} \in \mathcal{L}^n$, $\alpha = \mathbf{u}_0\mathbf{v}_0 - \sum_{i=1}^n \mathbf{u}_i\mathbf{v}_i$ is the Lorentzian inner product of $\mathbf{u},\mathbf{v}$. For Poincaré distance, when $\mathbf{x},\mathbf{y}$ are near the boundary, the floating point representation of $\|\mathbf{x}\|^2, \|\mathbf{y}\|^2$ approaches one and the denominator of the geodesic distance becomes zero, making parallel transport numerically instable as the operation depends on dividing by distance. For the Lorentz model, if $\mathbf{u} = \mathbf{v}$ and $\mathbf{u}$ contains large coordinate values, α becomes less than one, which results in NaN because the domain of $\cosh^{-1}(x)$ is $x \geq 1$.

(iv) Mapping Error. The parallel transport and tangent space methods require mapping a point in hyperbolic space to the tangent space, typically using the origin as the reference point for efficient computation. However, this introduces mapping errors for points not at the origin, especially for those far from it [51].

(v) Lack of Geometric Meaning. The space addition method from HCNN [2] does not have a meaning within hyperbolic geometry, providing no motivation and justification as to why it works.

In particular, the parallel transport method proposed by Poincaré ResNet [39] and Riemmanian ResNet [17] suffer from (i), (ii), (iii), (iv). The tangent space addition method used in HGCN [3] and LGCN [54], suffers from (i), (ii), (iv). The space addition method from HCNN [2] suffers from (v). In Figure 1, we show the visualization of LResNet and 2 of the previous methods: the parallel transport method and tangent space. As demonstrated, these two methods suffer from the need for multiple mappings to and from the tangent space while our method operates entirely on the hyperbolic manifold. The space addition method is not shown as it does not have geometric interpretations.

Proposed method. To overcome the above limitations, we propose **L**orentzian **Res**idual **Net**works (LResNet), a residual neural network utilizing the weighted Lorentzian centroid. Unlike existing methods that use parallel transport and tangent space addition, our approach normalizes the Euclidean weighted sum to directly project the output into the Lorentz model. Consequently, we avoid the need to map between tangent and hyperbolic spaces, operating directly on the manifold. This approach addresses the limitations (i), (iii), and (iv), and ensures the commutativity of addition, thereby resolving limitation (ii). Unlike HCNN [2], our method has geometric interpretations and alleviates limitation (v), both in the form discussed in Proposition 4.2 and as the centroid w.r.t. to the Lorentzian distance [22]. Theoretically, we demonstrate that the proposed method can derive all of the discussed previous methods, offering general applicability while ensuring stability (see Lemma 4.1). Experimentally, our method achieves superior performance across multiple graph and computer vision task datasets while being effective in addressing the graph over-smoothing problem.

We demonstrate the versatility of LResNet with adaptations to several model architectures, including (1) GNNs, with up to 10.9% improvement over previous hyperbolic residual connection methods; (2) graph Transformers, with up to 2.5% improvement; and (3) CNNs for vision tasks, with up to 0.5% improvement and more robustness. Additionally, our method achieves over 2,000 times faster computation when compared to the parallel transport and tangent space method. It should be noted that LResNet can be applied to **any** hyperbolic neural network that operates on the Lorentz hyperboloid. Additionally, HNN++[35] has proven that previous midpoint computations proposed for the Beltrami-Klein model and Poincaré ball model [38] is equivalent to the Lorentzian centroid [22] under isometric projections. As LResNet is based on a generalization of the Lorentzian centroid, it can be potentially extended to other hyperbolic formulations via these mappings.

Contributions. The main contributions of this work can be summarized as follows: (1) We introduce LResNet, a numerically stable foundational residual connection method for hyperbolic neural networks that does not rely on the tangent space and parallel transport, thus being more efficient and effective than previous methods. (2) We successfully apply the proposed method LResNet across various domains and tasks, including computer vision and graph tasks, and across multiple architectures such as CNN, GNN, and graph transformer. (3) We theoretically analyze the limitations of existing methods and show that the proposed method encompasses previous methods, greatly enhancing the understanding of hyperbolic residual networks.

2 Related Works

Hyperbolic representation and deep learning. Hyperbolic spaces have garnered extensive attention in recent years for representation learning and deep learning [26, 30, 47]. 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 [19]. 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 [28, 29], graphs [3, 24, 50, 53], social networks [46], and recommendation systems [6, 37, 44, 48] utilizing hyperbolic representations. Moreover, hyperbolic deep learning has demonstrated impressive performance in image-related tasks, including image embedding [10, 15, 18] and segmentation [1, 4, 42], offering new insights into deep learning paradigms, such as the interpretation of norms as reflections of uncertainty. Within the neural network domain, HNN [12] presented the pioneering form of hyperbolic neural networks, defining fundamental hyperbolic transformations, classifiers, and hyperbolic multiclass logistic regression (MLR). HNN++ [35] 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 [5] and Hypformer [45] 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 their hyperbolic adaptations. Residual connections are a fundamental module in modern neural networks [16], 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 [39] 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, map to the tangent space of a new point, and subsequently map them back using the exponential map. Similarly, Riemannian ResNet [17] proposes an analogous method to perform this operation. HCNN [2] proposes to sum the space components and uses post-summation processing on the time component.

However, these approaches have several limitations, as discussed in the introduction. This work aims to address these challenges.

3 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 [28], the Lorentz (Hyperboloid) model [29], and the Klein model [14]. These models are isometric, meaning that points in one model can be smoothly mapped to another while preserving distances, angles, and geodesics [32].

Lorentz model. An n-dimensional Lorentz model is a Riemannian manifold $(\mathcal{L}^n, \mathfrak{g}_n^K)$ equipped with the Riemannian metric tensor $\mathfrak{g}_n^K = \text{diag}(-1, 1, \dots, 1)$ 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 parametrized form $[x_t, \mathbf{x}_s]^T$ 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

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_t y_t + \mathbf{x}_s^T \mathbf{y}_s = \mathbf{x}^T \mathfrak{g}_n^K \mathbf{y}, \tag{2}$$

with $|||\mathbf{x}|||_{\mathcal{L}} := \sqrt{|\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}}|}$ 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 $[\sqrt{-1/K}, 0, \dots, 0]^T$.

Tangent space. The tangent space at a point $\mathbf{x} \in \mathbb{L}^{K,n}$ is set of points orthogonal to $\mathbf{x}$, defined as

$$\mathcal{T}_{\mathbf{x}} \mathbb{L}^{K,n} = \{\mathbf{y} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = 0\}.$$

Notably, the tangent space is isometric to Euclidean space.

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

$$\exp_{\mathbf{x}}^K(\mathbf{y}) = \cosh(\alpha)\mathbf{x} + \frac{\sinh(\alpha)}{\alpha}\mathbf{y}, \alpha = \sqrt{-K\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}}, \tag{3}$$

$$\log_{\mathbf{x}}^K(\mathbf{x}) = \frac{\cosh^{-1}(\beta)}{\sqrt{\beta^2 - 1}}(\mathbf{y} - \beta\mathbf{x}), \beta = K\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}. \tag{4}$$

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 Möbius addition from [3, 12] is given by

$$\mathbf{x} \oplus_P \mathbf{y} = \exp_{\mathbf{x}}^K \circ \mathbf{P}_{\mathbf{o}\to\mathbf{x}} \circ \log_{\mathbf{o}}^K(\mathbf{y}). \tag{5}$$

Another approach is the tangent space addition method used for aggregation in [3], given as:

$$\mathbf{x} \oplus_T \mathbf{y} = \exp_{\mathbf{o}}^K \left(w_x \log_{\mathbf{o}}^K(\mathbf{x}) + w_y \log_{\mathbf{o}}^K(\mathbf{y}) \right), \tag{6}$$

where $w_x, w_y > 0$ are weights. A third approach is the space-like dimension addition method from [2] given by

$$\mathbf{x} \oplus_S \mathbf{y} = \left[\sqrt{||\mathbf{x}_s + \mathbf{y}_s||^2 - 1/K}, \mathbf{x}_s + \mathbf{y}_s \right]^T, \tag{7}$$

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

4 Methodology

In this section, we introduce our proposed method for Lorentzian residual connection, based on a generalization of the Lorentzian centroid [22], and analyze its theoretical properties, including numerical stability and representation power. We also propose the use of an optional scaling method that helps to control the norm of the output from the residual layer.

4.1 Lorentzian residual connection.

In a standard Euclidean residual block, let $\mathbf{x}$ and $f(x)$ represent the input and output from a neural network layer or series of layers. The residual connection is expressed as $\mathbf{x} + f(\mathbf{x})$, or more generally, as $\alpha\mathbf{x} + \beta f(\mathbf{x})$, where α and β are scalar weights.

Given vectors $\mathbf{x}, f(\mathbf{x}) \in \mathbb{L}^{K,n}$, the Lorentzian residual connection is defined as follows:

$$\mathbf{x} \oplus_{\mathcal{L}} f(\mathbf{x}) := \frac{w_x \mathbf{x} + w_y f(\mathbf{x})}{\sqrt{-K} | \|w_x \mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}} |}, \tag{8}$$

where $|\|\cdot\||_{\mathcal{L}} = \sqrt{|\langle\cdot\rangle_{\mathcal{L}}|}$ is the Lorentzian norm and $w_x, w_y > 0$ are weights that can be learned or fixed. This can be seen as a projection of the Euclidean weighted sum into $\mathbb{L}^{K,n}$ via the normalizing denominator.

Following the general form of the residual connection in the Euclidean case, we can reformulate Equation (8) as a weighted sum of Lorentzian hyperbolic vector, given by

$$\mathbf{x} \oplus_{\mathcal{L}} f(\mathbf{x}) := \alpha_{w_x,w_y} \mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x}), \tag{9}$$

where

$$\alpha_{w_x,w_y} = w_x / \sqrt{-K} | \|w_x \mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}} |$$
$$\beta_{w_x,w_y} = w_y / \sqrt{-K} | \|w_x \mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}} |$$

are normalized weights.

Lorentz residual network (LResNet) The core component of LResNet is the Lorentzian residual block, which consists of a hyperbolic layer followed by a Lorentzian residual connection. The hyperbolic layer can be any type of layer adapted to the Lorentz model, such as hyperbolic fully-connected layers [5], hyperbolic convolutional layers [2], or hyperbolic attention mechanisms [14]. In Algorithm 1, we demonstrate LResNet applied to a classification network as an example. However, this method is not confined to this particular scenario. To ensure that the weights are constraints to the feasible domain, in practice we fix w_x to be a positive constant and we take the absolute value of w_y. Please see Lemma 4.1 and accompanying discussion for more details on the feasible domain.

Next we study the theoretical aspects of LResNet. We show that LResNet is numerically stable in Lemma 4.1 and that it can derive previous methods in Proposition 4.2. We also show that LResNet is a valid hyperbolic operation and it is commutative. The longer proofs for all of our theoretical results can be found in Appendix A.

Numerical stability. We can always select weights α_{w_x,w_y} and β_{w_x,w_y} to ensure the LResNet is numerically stable. Note that the only possible source of numerical instability is from the division, hence it suffices to ensure the dominator does not approach zero, which we show via the following lemma.

Lemma 4.1. *$\sqrt{-K}|\|w_x\mathbf{x} + w_y\mathbf{y}\|_{\mathcal{L}}| > \sqrt{w_x^2 + w_y^2}$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{L}^{K,n}$ and $(w_x, w_y) \in \mathbb{R}^+ \times \mathbb{R}^+ \setminus \{(0,0)\}$, where $\mathbb{R}^+$ denotes the set of non-negative real numbers.*

Proof. Note that

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

where

$$\Gamma = w_x\sqrt{||\mathbf{x}_s||^2 - \frac{1}{K}} \cdot w_y\sqrt{||\mathbf{y}_s||^2 - \frac{1}{K}} - w_x w_x\langle\mathbf{x}_s, \mathbf{y}_s\rangle \tag{10}$$

By taking the square-roots we have the desired inequality. □

Since the normalizing denominator divides out the ratio w_x/w_y, the output of LResNet is preserved when the ratio is preserved. Thus we can fix w_x to be a positive constant. In the case of $w_x = 1$, we obtain $||w_x\mathbf{x} + w_y\mathbf{y}||_{\mathcal{L}} > 1/\sqrt{-K}$. This lemma ensures that the denominator can always lower bounded by a positive constant of our choosing (in this case, $1/\sqrt{-K}$). Thus we never risk dividing by values close to zero, making *LResNet* **numerically stable**. Note that w_y is still allowed to take on any arbitrary non-negative value, which allows for any arbitrary ratio w_x/w_y. Thus fixing $w_x = 1$ does not at all restrict the values of the output of Equation (9) and any discussion of using arbitrary w_x values still applies.

Validity of LResNet. Here we show that LResNet is a valid hyperbolic operation. To see that Equation (9) indeed maps to hyperbolic space, first note that for any $\mathbf{x}, f(\mathbf{x}) \in \mathbb{L}^{K,n}$ we have

$$\begin{aligned}
&\left\langle \alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x}), \alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x}) \right\rangle_{\mathcal{L}} \\
&= \frac{\langle w_x\mathbf{x} + w_y f(\mathbf{x}), w_x\mathbf{x} + w_y f(\mathbf{x})\rangle_{\mathcal{L}}}{-K|\|w_x\mathbf{x} + w_y f(\mathbf{x}), w_x\mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}}|} \\
&= \pm 1/K.
\end{aligned}$$

Given that $\alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x})$ is a positive time-like vector, its Lorentzian inner product with itself must be negative, specifically it must be $-1/K$ from above. Therefore $\alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x}) \in \mathbb{L}^{K,n}$, confirming that LResNet is a valid hyperbolic operation.

Relation to previous methods. Our approach can theoretically derive the geometric meaning of the previous methods mentioned in Section 3, based on the following results in Proposition 4.2.

Proposition 4.2. *Let* $\mathbf{z}$ *be the output of one of the parallel transport methods in Equation* (5)*, the tangent space method in Equation* (6)*, or the space addition method in Equation* (7)*. 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* $\mathbf{z}$*.*

Then by carefully selecting weights (or using trainable weights) in LResNet, we can derive previous methods using our method. This shows that LResNet has at least the representative power of any of the previous methods, ensuring its expressiveness.

Commutativity. LResNet is commutative as it is a generalization of the weighted sum and $w_x\mathbf{x} + w_y\mathbf{y} = w_y\mathbf{y} + w_x\mathbf{x}$. For completeness and rigor, we also include the following for the non-commutativity of the parallel transport method.

Theorem 4.3. *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* $\mathbf{z}_{n+1} = -\mathbf{z}'_{n+1}$*.*

The two directions of the parallel transport addition in the above case are reflected over an entire axis, giving theoretical motivation for its inflexibility and results shown in Section 5.4. For instance, when $n = 2$, $K = -1$, $\mathbf{x} = [3, 2, -2]^T$ and $\mathbf{y} = [3, 2, 2]^T$, we have $\mathbf{x} \oplus_P \mathbf{y} = [9, 8, -4]^T$ and $\mathbf{y} \oplus_P \mathbf{x} = [9, 8, 4]^T$.

Computational Complexity. To compute the weighted sum for LResNet in Equation (9), we calculate the Lorentzian norm in the denominator of the weights α_{w_x,w_y} and β_{w_x,w_y}, which takes $O(n)$ time. Afterward, the remaining computation is a straightforward Euclidean addition, allowing LResNet to be computed in $O(n)$ time, similar to the Euclidean case.

4.2 Advantages over previous approaches

LResNet overcomes all of previous methods' limitations discussed in Section 1. In this section we summarize the advantages below.

(i) Unlike previous methods that involve multiple mappings between hyperbolic and tangent spaces, LResNet is a simple weighted sum, making it significantly more efficient. Section 5.4 provides a more detailed runtime analysis in Table 6, where LResNet achieves over *2000 times* faster computation than both the parallel transport and tangent space method in the same setting.

(ii) LResNet is commutative. In the Section 4.1 and Section 5.4, we discuss the effects of the non-commutativity on the parallel transport method [3], showing that it is unpredictable which direction of addition achieves better performance.

(iii) The result from Lemma 4.1 ensures that LResNet is computationally stable, avoiding the computational instability issues mentioned in the introduction.

(iv) By eliminating the mapping between tangent and hyperbolic spaces, LResNet avoids mapping errors for points far away from the origin [51].

(v) LResNet has a geometric interpretation, with the ability to theoretically achieve previous methods (Proposition 4.2). By carefully selecting weights, LResNet is able to achieve the geometric meaning of previous methods by ensuring the outputs lie on the same geodesic from the origin. This ensures the representation power of LResNet and provides theoretical motivation for LResNet as opposed to the space addition method [2].

Algorithm 1 Lorentz Residual Network (LResNet)

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 $l = 1, \ldots, L$, hyperbolic layers $f^{(l)} : \mathbb{L}^{K,n} \to \mathbb{L}^{K,n}$ for $l = 1, \ldots, L$

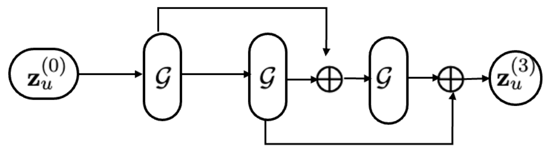
```
1:  for each epoch do
2:      for each batch of data $\mathbf{x}_b$ do
3:          $\mathbf{h}^{(0)} \leftarrow \mathbf{x}_b$ // Initialization
4:          for $l \leftarrow 1$ to $L$ do
5:              $\mathbf{z}^{(l)} \leftarrow f^{(l)}(\mathbf{h}^{(l-1)})$ // Hyperbolic layers
6:              $\alpha^{(l)} \leftarrow \frac{w_x^{(l)}}{\sqrt{-K}|\|w_x\mathbf{x}+w_y f(\mathbf{x})\|_{\mathcal{L}}|}$
7:              $\beta^{(l)} \leftarrow \frac{w_y^{(l)}}{\sqrt{-K}|\|w_x\mathbf{x}+w_y f(\mathbf{x})\|_{\mathcal{L}}|}$
8:              $\mathbf{h}^{(l)} \leftarrow \alpha^{(l)}\mathbf{h}^{(l-1)} + \beta^{(l)}\mathbf{z}^{(l)}$ // LResNet
9:          end for
10:         $\hat{\mathbf{y}} \leftarrow \text{softmax}(\mathbf{h}^{(L)})$ // Output prediction
11:         $\mathcal{J} \leftarrow \ell(\hat{\mathbf{y}}, \mathbf{y})$ // Compute loss
12:         Update weights $w_x^{(l)}$ and $w_y^{(l)}$ for $l = 1, \ldots, L$ using optimizer with $\mathcal{J}$ and $\eta$
13:     end for
14: end for
```

4.3 Optional scaling

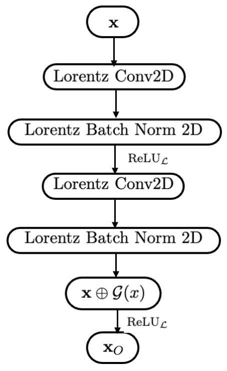
Very often the Euclidean norm of the hyperbolic embedding have important implications in model performance. For example, in image classification tasks, the norm tends to be positively correlated with classification confidence [13]. However, previous work [22] have shown that the unweighted Lorentzian centroid (a specific case of LResNet) can get very close to the origin in manifolds with low curvature, resulting in a small Euclidean norm. Additionally, have too large of a norm could also lead to mapping errors [9]. To help keeping the norm of the output within reasonable ranges, we propose to use an optional scaling method after the LResNet computation when training in low-curvature manifolds. For a hyperbolic vector $\mathbf{m} \in \mathbb{L}^{K,n}$, the scaled value is

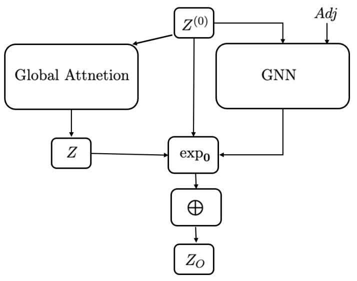
$$\begin{bmatrix} \sqrt{||\gamma \cdot \mathbf{m}_s||^2 - \frac{1}{K}} \\ \gamma \cdot \mathbf{m}_s \end{bmatrix}. \tag{11}$$

where $\gamma > 0$ is the scaling constant, which can be fixed or trained. Note that since geodesics in the Klein model are Euclidean lines and the isometry $\varphi_{\mathcal{K}}$ that maps from the Lorentz model to the Klein model is given by $\varphi_{\mathbb{K}}(\mathbf{x}) = \mathbf{x}_s/x_t$, this scaling simply slides $\mathbf{m}$ along the geodesic to or from the origin, ensuring the scaled output still satisfies Proposition 4.2. Note that the scaling also levitates the potential limitation that the output of LResNet is always bounded by the larger of the two inputs, allowing for more expressiveness and representative power.



(a) LResNet for GCN





(b) LResNet for CNN (c) LResNet for Graph Transformer

Figure 2: Adaptation of LResNet to (a) 3-layer GNN architecture (b) residual block for vision tasks, and (c) graph transformer. $\mathcal{G}$ represents graph convolutional layers in (a) and convolutional layer implemented with HL in (b). $\bigoplus$ is a hyperbolic residual connection.

5 Experiments

To demonstrate the effectiveness and versatility of our method, we apply LResNet as the residual connection in multiple adaptations of hyperbolic neural networks. We demonstrate that our method achieves better results than any previous methods of residual connection. In each task, we focus on testing the residual connection methods by using a consistent base model. In Section 5.1 and Section 5.2, we test LResNet on several graph datasets, as part of hyperbolic GNN and graph Transformer architectures to demonstrate its effectiveness. In Section 5.3, we test LResNet on image datasets and demonstrate its effectiveness and robustness. In Section 5.4, we perform further analysis that demonstrates the efficiency and effectiveness of LResNet and shortcomings of previous methods. The datasets were selected as they exhibit low Gromovs δ-hyperbolicity (δ) [3] values, indicating highly hierarchical structures[1].

For all of our experiments, we used largely the same set of hyperparameters as the did the base model without residual connection whenever available. Additionally, we find that the addition of a residual layer does not add much tuning overhead. We include a discussion of the effects of the curvature K as hyperparameters in Appendix C.4, which is missing in the work of the base models.

[1]For homogeneous graph datasets, δ is taken directly from HGCN [3]. For heterogeneous datasets, δ was computed by us using the same code from HGCN [3]. In both cases, the values was not normalized. For image datasets, δ is taken from HCNN[2], where the value is normalized by the largest distance between two points in the dataset using the method from [18].

5.1 Adaptation to GNNs

GNN model architecture. We formulate a skip-connected graph convolutional network with LResNet as the residual connection, using the fully hyperbolic graph convolutional layer [5]. For the overall model architecture, we follow the ResGCN architecture outlined in [37, 44], where the output of each intermediate layer was added to the output of the next layer via LResNet or a baseline hyperbolic residual connection method. The model architecture is shown in Figure 2a, where $\mathcal{G}$ represents the graph convolutional layer and arcs are residual connections.

GNN experimental setup. We evaluate LResNet on both node classification and link prediction tasks. We utilize several datasets: (1) homophilous graphs, including Disease[3], Airport[3], and two benchmark citation networks, namely PubMed[34] and Cora[34]; (2) Heterophilic graphs, including Chameleon and Squirrel, where we use the splits proposed by [31]. For homophilous graphs, the Disease and Airport datasets exhibit a more hierarchical structure, whereas the citation networks are less so, making them suitable for demonstrating the generalization capability of LResNet. For evaluation, we report the F1-score for the node classification tasks and the Area Under Curve (AUC) for the link predictions task. For heterophilic graphs, we focus on the more difficult task of node classification and report accuracy percentage.

We closely follow the implementation of HyboNet model [5], utilizing its fully hyperbolic linear and aggregation layers. Our optimization follows the same setup, with parameters grouped based on their presence on the hyperbolic manifold. For the decoder, we implemented the Fermi-Dirac decoder [20, 29]. For implementation of LResNet, we applied constant weights of 1 for simplicity.

Baselines. We test the effectiveness of LResNet against other hyperbolic residual methods, by applying the baselines instead of LResNet. We consider the base HyboNet [5] without residual connection, and the previous residual connection methods of the parallel transport method [3], the tangent space method [3], the space addition method [2].

Experimental findings. We show the results in Table 1. Due to space constraints and for readability purposes, we omit the comparison to results from Euclidean GCNs and other hyperbolic GCNs. These comparisons can be found in Table 5 of [5], where the baseline HyboNet outperforms the aforementioned GCNs. As shown in Table 1, LResNet is the best performer in 8 out of the 10 tasks, for up to 4.2% in the case of node prediction for Chameleon. Compared to the base HyboNet without residual connection, LResNet substantially outperforms in 9 out of the 10 tasks, with the one remaining tasks being of comparable performance. Compare to the baseline residual connection methods, LResNet is the notably 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 problems. LResNet 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, LResNet always performs better and by large margins, suggesting that it is more suitable for hyperbolic networks as it doesn't map between hyperbolic and tangent (Euclidean) spaces.

Table 1: Test ROC AUC results (%) for Link Prediction (LP), F1 scores (%) for Node Classification (NC) for homophilous graph, and accuracy (%) for NC for heterophilic graph. The best performance is highlighted in bold. A lower δ value indicates a more tree-like dataset. The first four delta values are sourced from HGCN [3], and we used the same code for the remaining datasets.

	Homophilous								Heterophilic	
Dataset	Disease		Airport		PubMed		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 [5]	96.8 ± 0.4	**96.0** ± 1.0	**97.3** ± 0.3	90.0 ± 1.4	95.8 ± 0.2	78.0 ± 1.0	93.6 ± 0.3	**80.2** ± 1.3	40.1 ± 0.8	34.3 ± 0.5
Parallel Transport [39]	86.4 ± 0.8	84.8 ± 3.7	93.6 ± 0.1	93.4 ± 0.6	**96.5** ± 0.1	77.8 ± 0.5	**94.8** ± 0.3	76.0 ± 0.8	36.6 ± 1.9	32.3 ± 0.8
Tangent Space [3]	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 [2]	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
LResNet (ours)	**97.3** ± 0.4	**96.1** ± 1.0	**97.3** ± 0.3	**93.9** ± 0.7	96.2 ± 0.1	**80.1** ± 1.0	94.1 ± 0.3	**80.6** ± 0.9	**41.1** ± 0.9	**37.1** ± 1.1

5.2 Adaptation to Graph Transformers

We also investigate the application of our method to graph Transformers, where we test LResNet as part of a hyperbolic adaptation of SGFormer [43], a recent SOTA Euclidean graph Transformer. We consider the same hyperbolic residual connection baselines as did in Section 5.1, namely the parallel transport method, the tangent space method, and the space addition method.

Lorentzian graph Transformer architecture. Following the notations in SGFormer [43], let $\mathbf{Z}^{(0)}$ be the input embedding, $\mathbf{Z}$ be the output of the global attention layer, and $\mathrm{GN}(\mathbf{Z}^{(0)}, \mathbf{A})$ be the output of the GNN layer where $\mathbf{A}$ is the adjacency matrix. For LResNet and the space addition method, we can project $\mathbf{Z}$ and $\mathrm{GN}(\mathbf{Z}^{(0)}, \mathbf{A})$ directly to $\mathbb{L}^{K,n}$. The final embedding is computed as

$$(1-\alpha)\exp_{\mathbf{o}}^{K}(\mathbf{Z}) \oplus \alpha \exp_{\mathbf{o}}^{K}(\mathrm{GN}(\mathbf{Z}^{(0)}, \mathbf{A})), \tag{12}$$

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 and the non-linearity of the exponential and logarithmic maps, we project weighted embeddings instead. In this case, the final embedding is then computed as

$$\exp_{\mathbf{o}}^{K}((1-\alpha)\mathbf{Z})) \oplus \exp_{\mathbf{o}}^{K}(\alpha \mathrm{GN}(\mathbf{Z}^{(0)}, \mathbf{A})). \tag{13}$$

The final embedding of the model is then in hyperbolic space. Figure 2c shows the visualization of the model.

Experimental setup. We closely followed the setup of the base SGFormer model [43]. For the datasets, we consider 3 heterophilic graphs datasets, namely Chameleon, Squirrel, and Actor. We use the same splits for Actor as [23] and the same splits from earlier for Chameleon and Squirrel. We report node classification accuracy as did in SGFormer [43].

As for the decoder, we use the same Fermi-Dirac decoder [20, 29] from earlier for the final fully connected layer (see below for more detail). For the optimizer, we utilized two separate optimizers similar to HyboNet [5] and the GCN experiments: the Euclidean Adam optimizer for non-manifold parameters and the Riemannian Adam optimizer for manifold parameters. We use trainable weights for LResNet.

Here we elaborate on the decoder used for node classification for clarity. Specifically, let $\mathbf{x}_h$ be the final embedding preceding the decoder layer. Then the decoder layers learns d vectors in hyperbolic space $\mathbf{v}_1, \ldots, \mathbf{v}_d$ to be the boundary of the d node classes.

Table 2: Test accuracy (%) for Node Classification on heterophilic graph datasets in a SGFormer-based graph Transformer architecture.

Method	Chameleon	Squirrel	Actor
Hyperbolicity	$\delta = 2$	$\delta = 1.5$	$\delta = 1.5$
SGFormer [43]	44.9 ± 3.9	41.8 ± 2.2	**37.9** ± 1.1
Parallel Transport [39]	46.7 ± 1.2	38.5 ± 1.3	35.5 ± 0.9
Tangent Space [3]	47.0 ± 0.8	42.1 ± 1.2	34.9 ± 0.7
Space Addition [2]	47.2 ± 1.4	43.0 ± 1.1	35.3 ± 0.4
LResNet (ours)	**47.8** ± 1.3	**43.9** ± 0.8	**38.0** ± 0.4

Table 3: Test accuracy (ACC) (%) for Image Classification task on CIFAR-10 and CIFAR-100 in the hyperbolic ResNet. The δ values are taken from HCNN [2], which are normalized by the diameter of the dataset.

Method	CIFAR-10 (ACC)	CIFAR-100 (ACC)
Hyperbolicity	$\delta = 0.26$	$\delta = 0.23$
Parallel Transport [39]	94.1 ± 0.13	72.9 ± 0.23
Tangent Space [3]	94.0 ± 0.19	71.5 ± 0.30
Space Addition [2]	94.3 ± 0.11	74.3 ± 0.22
LResNet (ours)	**94.6 ± 0.17**	**74.8 ± 0.25**

Intuitively, the distance between $\mathbf{x}_h$ and $\mathbf{v}_k$ represents the negative log-likelihood of $\mathbf{x}_h$ being in class k. The classification of $\mathbf{x}_h$ is thus $\arg\max -d(\mathbf{v}_k, \mathbf{x}_h)$. Please see Section 3.3 of [29] for more details.

Experimental findings. The results are presented in Table 2. LResNet consistently outperforms both the base Euclidean SGFormer and the baseline hyperbolic residual connection methods across all three cases, highlighting its effectiveness in Transformer models. Moreover, in 2 of the 3 datasets, the hyperbolic SGFormer nearly always surpasses the Euclidean version, supporting the advantages of hyperbolic modifications.

5.3 Adaptation to Vision Model

Lorentzian ResNet architecture. Here we define the hyperbolic ResNet structure we use for the image experiments. The network is made simple to highlight the effects of residual connections. For the traditional convolutional layer, batch normalization layer, and ReLu activation layer utilized in Euclidean ResNet in [16], we consider

Table 4: ROC AUC (AUROC,%), AUPR(%), and FPR96(%) results for OOD detection on CIFAR-10 and CIFAR-100 with Places365, DTD, and SVHN, as OOD datasets. R20 and R32 here denote ResNet-20 and ResNet-32 architectures, both with channel widths (8, 16, 32). The best performance is highlighted in bold. For FPR95, lower is better. For AUROC and AUPR, higher is better.

Dataset	ResNet	CIFAR-10 $\delta = 0.26$						CIFAR-100 $\delta = 0.23$					
		FPR95↓		AUROC ↑		AUPR ↑		FPR95↓		AUROC ↑		AUPR ↑	
		R20	R32	R20	R32	R20	R32	R20	R32	R20	R32	R20	R32
Places	Euclidean [16]	64.2	72.3	**84.7**	82.0	**96.2**	95.6	89.5	93.9	62.5	57.9	89.3	87.9
	w/ HNN++[39]	63.8	72.7	79.6	77.7	94.5	94.2	93.2	86.3	63.6	66.6	89.8	91.1
	Poincaré[39]	70.2	70.7	82.3	82.6	95.7	95.9	**82.8**	**83.8**	**71.5**	71.1	**92.3**	92.2
	LResNet (ours)	**63.0**	**63.3**	82.6	**82.7**	95.7	**96.0**	93.2	93.2	53.6	**71.7**	84.5	**92.4**
SVHN	Euclidean[16]	97.3	94.7	68.8	73.4	92.8	94.1	99.5	98.8	43.7	54.6	83.7	88.2
	w/ HNN++[39]	73.1	79.1	85.5	82.2	96.9	96.1	92.1	88.6	66.4	68.9	91.1	92.0
	Poincaré[39]	66.0	69.3	85.0	83.6	96.6	96.3	**76.9**	83.0	**76.8**	72.6	**94.1**	**92.9**
	LResNet(ours)	**39.0**	**56.9**	**90.8**	**88.0**	**98.0**	**97.4**	80.1	**80.7**	66.9	**73.4**	90.7	**92.9**
Texture	Euclidean[16]	87.3	88.0	73.6	77.3	93.2	94.7	98.1	96.0	33.5	42.9	75.9	79.4
	w/ HNN++[39]	**63.8**	**56.6**	79.6	**85.8**	94.5	96.6	85.9	**77.5**	58.9	65.7	86.8	89.0
	Poincaré[39]	68.2	66.2	82.1	82.3	95.5	95.6	83.9	84.2	**67.7**	68.8	**91.0**	**91.5**
	LResNet (ours)	**63.8**	68.4	**84.2**	**85.8**	**96.3**	**96.7**	**83.7**	83.7	57.6	**69.1**	86.3	**91.5**

a simple hyperbolic adaptation. For an input vector $\mathbf{x} \in \mathbb{L}^{K,n}$, we define a *hyperbolic layer* (HL) to be a neural network layer that first applies the Euclidean layer on the space-like dimension of an input vector $\mathbf{x}$ to obtain $f(\mathbf{x}_s)$ where f is the Euclidean layer, and then calculating the time-like dimension afterwards. Formally, it outputs a vector $\mathbf{y} \in \mathbb{L}^{K,n}$ such that $\mathbf{y}_s = f(\mathbf{x}_s)$ and $y_t = \sqrt{||\mathbf{y}_s||^2 - \frac{1}{K}}$. For LResNet, we use trainable weights as outlined in Equation (8). We also use the scaling method outlined in Equation (11) to take advantage of the positive correlation between embedding norm and classification confidence [13]. Figure 2b shows a visualization of our residual block, where the Lorentz variation (and subscript $\mathcal{L}$) of an operation indicates implementation with a hyperbolic layer. For the decoder, we experienced numerical instability with the Lorentz MLR in HCNN[2]. As a result, we use the isometry map [3] to project the final embedding from the Lorentz model into the Poincaré ball model to use the Poincaré MLR from HNN++[35]. Specifically, for the final fully connect output layer, we first map the Lorentzian embedding to the corresponding Poincaré ball model, then we use the pooling method in Poincaré ResNeT [39] where pooling is done in Euclidean space and mapped back into the Poincaré ball. The Poincaré MLR layer is then applied to the hyperbolic pooling output. For the optimizer, we used the Riemannian SGD optimizer that was used in HCNN [2] for classification and the Adam optimizer for ODD-detection as did in Poincaré ResNet [39].

Classification accuracy experiment. To demonstrate the effectiveness of LResNet over previous methods, we evaluate image classification performance using the ResNet-18 architecture [16] and two datasets: CIFAR-10[21] and CIFAR-100[21]. For the experiment, we use the structure shown in Figure 2b and test our residual connection against previous methods. For the baseline, we test against the parallel transport method, the tangent space method, and the space addition method. As parallel transport is not commutative, we use $\mathbf{x} \oplus_{\mathcal{P}} f(\mathbf{x})$ where $f(\mathbf{x})$ is the output of the convolutional layer. We adopt the training procedure of [2, 8] which have been optimized for Euclidean CNNs and yielded strong results for both Euclidean and Hyperbolic ResNets. The results are shown in Table 3. LResNet performs notably better than all of the previous methods for both datasets, demonstrating its effectiveness as a residual connection method for vision tasks.

Table 5: Test ROC AUC result (%) for both directions of the parallel transport (PT) method in Link Prediction.

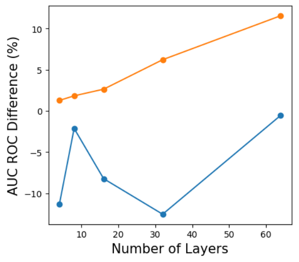
Direction of PT	Disease ($\delta = 0$)	Airport ($\delta = 1$)
Forward	$\mathbf{86.8 \pm 0.8}$	93.6 ± 0.1
Backward	67.7 ± 1.4	$\mathbf{93.7 \pm 0.1}$

Out-of-distribution (OOD) robustness. To check that LResNet is robust to out-of-distribution samples, we test the OOD detection performance of our Lorentzian ResNet-20 and ResNet-32 with channel widths (8, 16, 32), trained on either CIFAR-10 or CIFAR-100. We closely follow the experimental setup in Poincaré ResNet [39]. We use the Places-365 dataset [56], the Texture dataset [7], and the SVHN dataset [27]. The baseline we compare to is the Euclidean and hyperbolic ResNets in Poincaré ResNet [39]. Poincaré ResNet is a baseline hyperbolic ResNet model using specially designed hyperbolic convolutional layers and uses the parallel transport method as residual connection, thus it can be seen as the parallel transport baseline. As for the detection of the OOD sample, we use the energy score which was introduced by [25] and also used in [39]. Finally, we compare our results against the baselines with the three commonly used metrics of FPR95, AUROC, and AUPR.

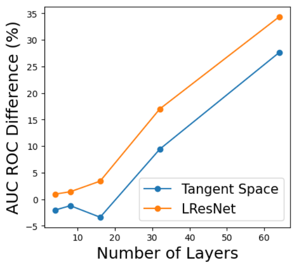
OOD detection results. The results in Table 4 indicate that LResNet, when trained on CIFAR-10, significantly outperforms Euclidean counterparts and previous hyperbolic ResNets on almost all metrics. On CIFAR-100, our model notably surpasses the baseline with a ResNet-32 architecture. This demonstrates that LResNet is more robust to OOD samples than its Euclidean and hyperbolic counterparts using parallel transport for residual connections, confirming that LResNet is not only effective and reliable.

Table 6: Average runtime to perform 100 additions on random vectors of dimension 2,048 and dimension size 10,000, and dimension 4,096 and size 100,000, on an RTX 3070.

Method	$2,048/10,000$	$4,096/100,000$
Parallel Transport [39]	0.0036s	1.448s
Tangent Space [3]	0.0083s	3.601s
LResNet (ours)	0.00025s	0.0006s



(a) Disease



(b) Airport

Figure 3: Comparison of ROC AUC (%) differences between HyboNet with and without residual connections: orange indicates our LResNet as the residual connection, while blue represents the tangent space method as the residual connection. In Figure 3a we show the results on the Disease dataset and in Figure 3b we show the results on the airport dataset.

5.4 Further Analysis

This section presents a detailed analysis of our proposed method, *LResNet*, highlighting its high efficiency through runtime experiments. We also examine its effectiveness in addressing the over-smoothing issue in deeper graph neural networks. We also demonstrate the need for a commutative residual connection by examining the effects of non-commutativity on the parallel transport method.

Efficiency analysis. We compare the efficiency of LResNet with previous multi-mapping methods, specifically the parallel transport and tangent space method, by conducting 100 additions on randomly generated hyperbolic vectors (on the manifold with curvature $K = -1$) with dimensions of 2048 and sizes of 10,000, and dimensions of 4096 and sizes of 100,000, using a single RTX 3070 GPU. The average runtime for each method is presented in Table 6. LResNet demonstrates significant speedup over the baselines and offers better scalability as the size and dimension increase.

Overcoming graph over-smoothing. Previous studies have shown that stacking graph convolutional layers often leads to performance drops due to gradient vanishing or over-smoothing [55]. Residual connections traditionally address these issues in Euclidean space. To examine the impact of the proposed method in hyperbolic space, we implemented the GCN architecture from Section 5.1 on the Disease and Airport datasets across 4, 8, 16, 32, and 64 layers, using the tangent space method as a baseline for link prediction tasks. The results in Figure 3 depict the difference in ROC AUC scores between residual-connected HyboNets and the base HyboNet for clarity. We observed that the performance gap between HyboNet and LResNet widens with more layers, indicating the effectiveness of LResNet in countering over-smoothing. Conversely, the tangent space method shows a reduction in differences, especially at 32 layers for Disease and 16 for Airport, highlighting LResNet's stability in deep neural networks. The parallel transport method yielded NaN values for 16 or more layers, reflecting its numerical instability, so it was excluded.

Commutative capability. To investigate the effect of commutativity of the residual connection in hyperbolic models, we test the performance on link prediction of both directions of addition for the parallel transport method on Disease and Airport datasets using the architecture in Figure 2a. Forward operation represents $\mathbf{x} \oplus_P f(\mathbf{x})$ and backward operation represents $f(\mathbf{x}) \oplus_P \mathbf{x}$, where $f(\mathbf{x})$ is the output of the hyperbolic graph convolutional layer. The results are shown in Table 5. 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.

6 Conclusion

In this work, we proposed LResNet, a hyperbolic residual neural network based on the weighted Lorentzian centroid in the Lorentz model of hyperbolic space. LResNet addresses the limitations of previous methods, offering increased efficiency, flexibility, numerical stability, and geometric information retention. We proved that LResNet can theoretically derive previous methods while ensuring numerical stability. Empirically, LResNet outperforms previous methods in image classification, link prediction, and node classification tasks, demonstrating superior performance and robustness compared to Euclidean and hyperbolic counterparts across multiple architectures, including GNNs, CNNs, and graph Transformers. However, it should also be noted that the performance of a residual-connected model is still conditioned upon the performance of the base model, while there lacks an abundance of hyperbolic models, such as in the case of diffusion models. One direction of future work is to apply LResNet to newly developed architectures.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) IIS Div Of Information & Intelligent Systems 2403317. We also gratefully acknowledge support in part from an Amazon Research Award, funding provided by Snap Research, and the Yale Office of the Provost. Additionally, this research has greatly benefited from the discussions and research talks held at the IMS-NTU Joint Workshop on Applied Geometry for Data Sciences. We extend our sincere gratitude to the reviewers for their valuable feedback and insightful suggestions

References

[1] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne van Noord, and Pascal Mettes. 2022. Hyperbolic Image Segmentation. In *CVPR*. 4453–4462.
[2] Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. 2024. Fully Hyperbolic Convolutional Neural Networks for Computer Vision. In *ICLR*.
[3] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *NeurIPS*. 4868–4879.
[4] Bike Chen, Wei Peng, Xiaofeng Cao, and Juha Röning. 2023. Hyperbolic uncertainty aware semantic segmentation. *TITS* (2023).
[5] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Fully Hyperbolic Neural Networks. *arXiv preprint arXiv:2105.14686* (2021).
[6] Yankai Chen, Menglin Yang, Yingxue Zhang, Mengchen Zhao, Ziqiao Meng, Jianye Hao, and Irwin King. 2022. Modeling Scale-free Graphs for Knowledge-aware Recommendation. *WSDM* (2022).
[7] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. 2014. Describing textures in the wild. *CVPR* (2014).
[8] Terrance DeVries and Graham W. Taylor. 2017. Improved regularization of convolutional neural networks with cutout. (2017).
[9] A. Dooley and N. Wildberger. 1993. Harmonic analysis and the global exponential map for compact Lie groups. In *Functional Analysis and Its Applications, 27(1):21–27*.
[10] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. 2022. Hyperbolic vision transformers: Combining improvements in metric learning. In *CVPR*. 7409–7419.
[11] Ali Faqeeh, Saeed Osat, and Filippo Radicchi. 2018. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Physical review letters* 121, 9 (2018), 098301.
[12] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *NeurIPS*. 5345–5355.
[13] Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. 2021. Hyperbolic Busemann Learning with Ideal Prototypes. *NeurIPS* 34 (2021).
[14] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. 2019. Hyperbolic attention networks. In *ICLR*.
[15] Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X Yu. 2022. Clipped hyperbolic classifiers are super-hyperbolic classifiers. In *CVPR*. 11–20.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[17] Isay Katsman, Eric Chen, Sidhanth Holalkere, Anna Asch, Aaron Lou, Ser Nam Lim, and Christopher M De Sa. 2023. Riemannian residual neural networks. *NeurIPS* 36 (2023).
[18] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. 2020. Hyperbolic image embeddings. In *CVPR*. 6418–6428.
[19] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82, 3 (2010), 036106.
[20] Dmitri Krioukov, Fragkiskos Papadopoulos, Amin Vahdat, and Marián Boguná. 2009. Curvature and temperature of complex networks. *Physical Review E* 80, 3 (2009), 035101.
[21] A. Krizhevsky. 2009. Learning multiple layers of features from tiny images.
[22] Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. 2019. Lorentzian distance learning for hyperbolic representations. In *ICML*. PMLR, 3672–3681.
[23] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. 2021. New Benchmarks for Learning on Non-Homophilous Graphs. *arXiv preprint arXiv:2104.01404* (2021).
[24] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *NeurIPS*. 8230–8241.
[25] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *NeurIPS* (2020).
[26] Pascal Mettes, Mina Ghadimi Atigh, Martin Keller-Ressel, Jeffrey Gu, and Serena Yeung. 2023. Hyperbolic Deep Learning in Computer Vision: A Survey. *arXiv preprint arXiv:2305.06611* (2023).
[27] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2011).
[28] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*. 6338–6347.
[29] Maximillian Nickel and Douwe Kiela. 2018. Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *ICML*. 3779–3788.
[30] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. 2021. Hyperbolic deep neural networks: A survey. *TPAMI* (2021).
[31] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at evaluation of GNNs under heterophily: Are we really making progress?. In *ICLR*.
[32] Arlan Ramsay and Robert D Richtmyer. 2013. *Introduction to hyperbolic geometry*. Springer Science & Business Media.
[33] Rik Sarkar. 2011. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*. Springer, 355–366.
[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
[35] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. 2020. Hyperbolic Neural Networks++. In *ICLR*.
[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*.
[37] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. 2021. HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering. In *WebConf*. 593–601.
[38] Abraham Albert Ungar. 2008. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics* 1, 1 (2008), 1–194.
[39] Max van Spengler, Erwin Berkhout, and Pascal Mettes. 2023. Poincaré ResNet. *CVPR* (2023).
[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.
[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[42] Zhenzhen Weng, Mehmet Giray Ogut, Shai Limonchik, and Serena Yeung. 2021. Unsupervised discovery of the long-tail in instance segmentation using hierarchical self-supervision. In *CVPR*. 2603–2612.
[43] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. 2023. SGFormer: Simplifying and Empowering Transformers for Large-Graph Representations. In *NeurIPS*.
[44] Menglin Yang, Zhihao Li, Min Zhou, Jiahong Liu, and Irwin King. 2022. Hicf: Hyperbolic informative collaborative filtering. In *KDD*. 2212–2221.
[45] Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In *KDD*. 3770–3781.
[46] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space. In *KDD*. 1975–1985.
[47] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. 2022. Hyperbolic graph neural networks: a review of methods and applications. *arXiv preprint arXiv:2202.13852* (2022).
[48] Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. 2022. HRCF: Enhancing Collaborative Filtering via Hyperbolic Geometric Regularization. In *WebConf*.
[49] Menglin Yang, Min Zhou, Hui Xiong, and Irwin King. 2022. Hyperbolic Temporal Network Embedding. *TKDE* (2022).
[50] Menglin Yang, Min Zhou, Rex Ying, Yankai Chen, and Irwin King. 2023. Hyperbolic Representation Learning: Revisiting and Advancing. *ICML* (2023).
[51] Tao Yu and Christopher M De Sa. 2019. Numerically Accurate Hyperbolic Embeddings Using Tiling-Based Models. In *NeurIPS 2019*.
[52] Renyu Zhang, Aly A Khan, and Robert L Grossman. 2020. Evaluation of Hyperbolic Attention in Histopathology Images. In *2020 IEEE 20th BIBE*. IEEE, 773–776.
[53] Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Fanny Ye. 2021. Hyperbolic graph attention network. *TBD* (2021).
[54] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021. Lorentzian Graph Convolutional Networks. In *WebConf*. 1249–1261.
[55] Lingxiao Zhao and Leman Akoglu. 2019. PairNorm: Tackling oversmoothing in GNNs. *ICLR* (2019).
[56] Bolei Zhou, Agata Lapedriza, Khosla Aditya, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *TPAMI* (2017).

A Proof of Theoretical Results

Proof of Theorem 4.3.

Proof. Let $\mathbf{x} = [a_1, \ldots, -a_{n+1}] \in \mathbb{L}_K^n$ and $\mathbf{y} = [a_1, \ldots, a_{n+1}] \in \mathbb{L}_K^n$ where each $a_i \in \mathbb{R}$ and $a_1 > 0$. We can easily compute (by following the computation in Proposition 4.2) 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

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

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

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

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,

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

and

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

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

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

and

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

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

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

and

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

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 4.2.

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:

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

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

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

Table 7: AUC-ROC for LP and NC for varying curvature.

Curvature	DISEASE		AIRPORT	
	LP	NC	LP	NC
−0.5	96.7 ± 0.4	95.4 ± 1.2	96.8 ± 0.2	93.9 ± 0.7
−1.0	97.1 ± 0.3	96.0 ± 0.8	97.0 ± 0.3	92.9 ± 0.5
−1.5	97.1 ± 0.3	95.8 ± 1.2	97.3 ± 0.3	93.6 ± 1.0
−2.0	97.3 ± 0.4	95.6 ± 0.9	97.3 ± 0.1	93.3 ± 0.8
Trainable	96.5 ± 0.5	96.1 ± 1.0	96.4 ± 0.5	93.1 ± 1.0

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.

□

B Dataset Processing Details

B.1 Graph datasets

For the homophilous datasets, we used the same splits as in HGCN [3]. For the heterophilic datasets, we use the same splits as in SFGormer [43], where the splits for Chameleon and Squirrel are from [31], and the splits for Actor are from [23].

B.2 Image datasets

For the image classification tasks, we used the splits implemented in PyTorch for CIFAR-10 and CIFAR-100, where there are 50,000 training images and 10,000 testing images. For OOD-dection, we followed the set up in [39], where we use the same data splits as in [39] and [25].

C Implementation and Training Details

C.1 Implementation details for graph datasets

C.1.1 GNN hyperparameters. For the homophilous graphs datasets, we used the largely the same hyperparmeters as in [5], 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 $\{3, 4, 5, 6, 7, 8\}$. While many values were searched, we find that the performance of LResNet and the baselines depend mostly on the performance of the base HyboNet without residual connection. As a result, we used **the same** hyperparameters as in [5].

For homophilous graph datasets, we used a constant weight of 1 for LResNet shown in Equation (9). For heterophilic datasets, we used trainable weights instead.

C.1.2 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 [43].

C.2 Implementation details for image datasets

For hyperparameters in the classification task, we performed a grid search where learning rate is within the set $\{0.1, 0.01, 0.001\}$ and weight decay is within the set $\{0, 5e-4, 5e-3\}$. For OOD-detection, we trained the model with the same hyperparameter as in [39] and using the Riemmanian Adam optimizer as in [39] as well. In both cases, we performed search for curvature in the set $\{0.1, 1, 1.5\}$ and the scaling coefficient in Equation (11) in the set $\{0.5, 1.0, 2.0, \text{trainable}\}$.

C.3 Further details and code

All experiments were performed on float32 datatype and on a single NVIDIA RTX 3070 GPU with 8gb of memory. For details on our implementations, please refer to the code at the GitHub link.

C.4 Analysis of curvature as hyperparameter

We examine the sensitivity of LResNet to the choice of curvature K of the hyperbolic manifold by conducting the link prediction and node classification experiments in Section 5.1 for $K = -0.5, -1.0, -1.5, -2.0$, trainable on the AIRPORT and DISEASE datasets. The results are shown in Table 7. We do not observe the model to be sensitive to curvature choices.