Effect of Geometry on Graph Neural Networks

Abstract

Hyperbolic Graph Neural Networks (GNNs) have emerged as a promising approach for modeling graph-structured data with less embedding distortion than Euclidean GNNs. In this paper, we explore the effect of geometry on the performance of three types of GNNs for node classification and link prediction. To do so, we adopt the Hyperbolic framework outlined in [5] and propose a family of mixed geometry GNNs with Hyperbolic and Euclidean components that can be trained jointly. We compare our mixed geometry models' performance and stability against their Euclidean and Hyperbolic counterparts across various datasets. Finally, we examine the impact of the choice of geometry and graph properties on hyperparameter selection.

We see that the mixed geometry models have the best performance for node classification, while the Hyperbolic models have the best performance for link prediction. Further, we see that for node classification, the choice of architecture had a more significant impact on the performance than the choice of geometry. Whereas for link prediction, the choice of geometry had a much more significant impact than the choice of architecture.

1 Introduction

With the advent of Graph Neural Networks (GNNs), geometric deep learning has gained prominence [3]. The idea is that the graph structure captures the geometric structure present in the dataset. One example of this is curvature. Works such as [28] have defined the Ricci curvature for graphs. Recently [40] showed that for nearest neighbor graphs for data on a manifold, the discrete Ricci curvature on the graph is consistent with the continuous Ricci curvature on the manifold. Hence the graph structure provides geometric information. A GNN then uses message passing to learn a map from the feature vectors on the nodes to the class labels. This map is usually parameterized as a neural network. Recent work has shown that parameterizing this map using Hyperbolic geometry improves performance [4, 5, 46, 47]. Instead of learning a map from Euclidean space to Euclidean space, we factor it through the Hyperbolic manifold. First, embed the features into Hyperbolic space. Then we do feature extraction in Hyperbolic space. Finally, we map back to Euclidean space to get the class label. *In this paper, we are interested in understanding the effect that using different geometries to parameterize the neural network has on performance.*

Many different types of metric data can be represented in Hyperbolic space with smaller distortion than Euclidean space. Examples of such data include text [8], social or other forms of networks [1, 16, 35], and cell development trees [14]. Hyperbolic representations have also been seen to be useful for computer vision [7, 11]. More broadly, data with a semantically rich hierarchical structure is better represented in Hyperbolic space. See surveys [22, 31, 43] for many more example applications. Due to this success, there has been significant work done in developing methods that can embed data in Hyperbolic space [18, 26, 27, 34, 38], and Hyperbolic neural networks [5, 9, 17, 36]. There are also a significant number of works that develop Hyperbolic graph neural networks [4, 19, 46, 47].

While Hyperbolic space is well-suited for representing hierarchies, certain datasets exhibit non-constant curvature manifolds. Notably, graph datasets often contain regions with flat and negative curvature, as demonstrated in [39]. Additionally, [10] illustrates that better representations can be learned in spaces with mixed curvature. Hence, since the data displays multiple curvatures, we believe that processing the data using mixed curvature models can be beneficial. Recent research explores spaces with varying curvature or combinations of geometries as alternative frameworks for integrating into neural networks [6, 10, 20, 21, 37, 42, 48].

In this paper, we take a closer look to try and understand the effect this geometry has on the model's performance. Specifically, we are interested in whether the geometry results in a global improvement independent of other factors such as architecture type, downstream task, or dataset. To do so, we define an intermediary between Euclidean and Hyperbolic networks that we call Mixed Geometry neural networks (MGNNs). We then evaluate the effect of the geometry by considering three different model architectures for each geometry and evaluating these nine models (three geometries time three architectures) on two tasks and nine different datasets,

Contributions Our main contributions are

1. We present a new model that incorporates Euclidean and Hyperbolic geometry, known as the Mixed geometry model.
2. We show that the effect of the geometry is very dependent on the type of task. Specifically, node classification and link prediction have very different trends.
3. We show that the Mixed models have the best performance for node classification and that Hyperbolic models have the best performance for link prediction.
4. We show that for node classification, the model's architecture has a larger impact on the performance than the geometry and that for link prediction, the geometry has a larger impact.
5. We train over 50,000 neural networks and provide their training, test, and validation statistics for each epoch of the training for each model. Hence we provide data that can be used to further analyze the impact of the geometry on the performance of neural networks.

The rest of the paper is organized as follows. Section 2 presents the background on Hyperbolic models. Section 3 presents the new Mixed geometry models. Section 4 presents the datasets and models used in the paper. Section 5 presents the node classification experiment, and Section 6 the link prediction experiment.

2 Hyperbolic Geometry Background

We briefly introduce the Hyperboloid or Lorentzian model of the Hyperbolic manifolds. The Hyperboloid model $\mathbb{H}^k$ of the Hyperbolic manifold is defined as

$$\mathbb{H}^k = \left\{ x \in \mathbb{R}^{k+1} : x_0 > 0, x_0^2 - \sum_{i=1}^{k} x_i^2 = 1 \right\}. \tag{1}$$

Here distances are given via $d(x, y) = \operatorname{arccosh}\left(-\sum_{i=1}^{n} x_i y_i + x_{n+1} y_{n+1} \right)$. For the Hyperboloid model, we have explicit formulas for the Exponential and Logarithmic maps. Specifically, the Exponential map is given by $\exp_x(v) = \cosh(\|v\|_{\mathbb{H}})x + \sinh(\|v\|_{\mathbb{H}}) \frac{v}{\|v\|_{\mathbb{H}}}$, and the Logarithmic map is given by $\log_x(y) = \frac{\operatorname{arccosh}(\beta)}{\sqrt{\beta^2 - 1}}(y - \beta x)$, where $\beta = -x_0 y_0 + \sum_{i=1}^{k} x_i y_i$ and $\|x\|_{\mathbb{H}} = -x_0^2 + \sum_{i=1}^{k} x_i^2$.

2.1 Using Tangent Space.

We are ready to describe the standard Hyperbolic graph neural network layer. Let F be any graph neural network layer on Euclidean space. Then the corresponding Hyperbolic version is traditionally given as follows $HF(x) = \log(F(\exp(x)))$ where $\exp, \log$ are the Exponential and Logarithmic maps. In some cases, only the feature mapping is done in the tangent space, and then the aggregation is done in Hyperbolic using the Frechet mean. In most cases, this is computationally expensive. This broadly includes works such as [4, 19, 46, 47].

2.2 Fully Hyperbolic Models

[5] introduced a fully Hyperbolic framework that adopts the Lorentz model as its feature space. Since the manifold is embedded into Euclidean space, given a vector $x \in \mathbb{H}^n \subset \mathbb{R}^{n+1}$ we can apply

$W \in \mathbb{R}^{n+1\times n+1}$ to x to get a vector Wx in $\mathbb{R}^{n+1}$. However, Wx need not lie on the Lorentzian manifold. The family of linear transformations W that map the Lorentz model to itself is called the Lorentz transformations. [24] showed that these transformations can be decomposed into a Lorentz rotation and a Lorentz boost. Lorentz rotations characterize rotation in spacial coordinates, whereas the Lorentz boosts describe relative motion with constant velocity.

Definition 1 (Lorentz Rotation)**.** The Lorentz rotation matrices are given by $\mathbf{R} = \begin{pmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \tilde{\mathbf{R}} \end{pmatrix}$, where $\tilde{\mathbf{R}}^\top \tilde{\mathbf{R}} = \mathbf{I}$ and $\det(\tilde{\mathbf{R}}) = 1$.

Definition 2 (Lorentz Boost)**.** Given a velocity $\mathbf{v} \in \mathbb{R}^n$, $\|\mathbf{v}\| < 1$ and $\gamma = \frac{1}{\sqrt{1-\|\mathbf{v}\|^2}}$, the Lorentz boost matrices are given by $\mathbf{B} = \begin{pmatrix} \gamma & -\gamma\mathbf{v}^\top \\ -\gamma\mathbf{v} & \mathbf{I} + \frac{\gamma^2}{1+\gamma}\mathbf{v}\mathbf{v}^\top \end{pmatrix}$.

Moreover, [5] proves that the conventional Hyperbolic linear layer that relies on logarithmic maps and exponential maps can only model a special form of "pseudo-rotation" but cannot model the Lorentz boost. Hence they propose the following Hyperbolic linear layer that covers both Lorentz rotation and Lorentz boost. Given a matrix $M = \begin{bmatrix} v \\ W \end{bmatrix}$ and an input x, the output is given by $z = \begin{bmatrix} \frac{\sqrt{\|Wx\|^2+1}}{v^T x} v \\ W \end{bmatrix} x$. They show that if x resides in the Hyperbolic manifold, then z must also reside in the Hyperbolic manifold. These models are trained using Riemannian Adam [15].

3 Mixed Geometry Networks

In this paper, we define Mixed geometry networks. In many prior works mixed geometry refers to product manifolds, or manifolds with non constant curvature. However, in this paper, we shall refer to Mixed geometry networks that use layers that combine Hyperbolic and Euclidean features. This paper simplifies the above layer to define a linear layer HL that maps $\mathbb{R}^d \to \mathbb{H}^k$. Given a matrix $W \in \mathbb{R}^{k\times d}$ and a vector $x \in \mathbb{R}^d$, define $\hat{x} := Wx \in \mathbb{R}^k$. Then the output $z := HL(x; W) \in \mathbb{H}^k \subset \mathbb{R}^{k+1}$, is given by $z = \begin{bmatrix} z_0 \\ \hat{x} \end{bmatrix}$, where

$$z_0 := \sqrt{1 + \|\hat{x}\|^2}. \tag{2}$$

This layer differs slightly from the one presented in [5] as it does not involve the vector v. Further, we claim that this maps not only the Lorentz manifold to itself but that it maps all of $\mathbb{R}^d$ to the Lorentz manifold. This can be seen by the following proposition.

Proposition 3. *For any vector $x \in \mathbb{R}^d$, and any matrix $W \in \mathbb{R}^{k\times d}$ the output $z = HL(x; W)$ lives in the Hyperboloid manifold $\mathbb{H}^k$.*

Proof. Using definition (2), we have $z_0^2 - \sum_{i=1}^k z_i^2 = 1$, satisfying (1). This implies $z \in \mathbb{H}^k$. □

The advantage of this is now clear. Specifically, unlike [5], we do not need to figure out how to apply the aggregation step for GNNs, and the non-linear activation for general neural networks in a way that preserves the manifold. That is, given $z = HL(x^{(0)}; W)$, we can apply a non-linearity f coordinate wise to z to get $x^{(1)}$ the input to the next layer. This is because we no longer require that the input to HL lies on the manifold. Further, this implies that we can use layers with different geometries for different layers. Thus, we have defined a mixed geometry model.

3.1 Training the model

Due to Proposition 3, we note that *any* W will suffice. Hence, we do not need to use Riemannian gradient descent but can use regular gradient descent or other optimization techniques. This contrasts with [5], where we believe they do not need Riemannian gradient descent but use it anyway. This difference is crucial for a variety of reasons. It has been shown that Hyperbolic embeddings and Hyperbolic neural networks can have numerical instabilities [23, 32, 45]. Part of this is due to the exponential and logarithmic maps. In our framework, similar to [5], we do not need these maps to parameterize the network. However, since [5] uses Riemannian gradient descent. However, since we do Euclidean gradient descent or Adam [12], we do not need them during the optimization process.

3.2 Graph Neural Network Architectures

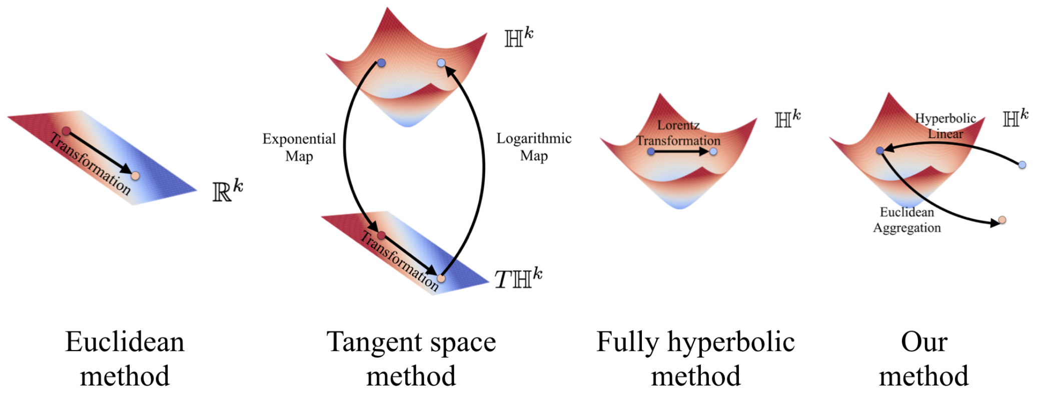


Figure 1: Visual comparison of linear layers for different methods, including the Euclidean method, tangent space method, fully Hyperbolic method, and our Mixed geometry method.

We now define the Mixed geometric version of three different types of graph neural network architectures. Specifically, Graph Convolutional Neural Network (GCN) [13], GraphConv (GC) [25], and Graph Attention Network version 2 (which we shall abbreviate as GAT) [2]. Figure 1 illustrates the difference between existing methods and our method regarding space transformations. We provide the exact update equations below.

GCN A standard GCN layer is given by $X^{(\ell+1)} = \hat{L} \cdot X^{(\ell)} \cdot W$ where $\hat{L}$ is the unnormalized graph Laplacian, $X^{(\ell)}$ are the input features to layer ℓ, and W is the weight matrix with the learnable parameters. For the Mixed geometry version, we modify the update rule to

$$X^{(\ell+1)} = \hat{L} \cdot HL(X^{(\ell)}; W).$$

Where W is the weight matrix with the learnable parameters. We note that the aggregation step, the multiplication by the unnormalized Laplacian, occurs in Euclidean space.

GC A standard GraphConv layer is given by $x_i^{(\ell+1)} = W_1 x_i^{(\ell)} + W_2 \sum\limits_{j \in \mathcal{N}(i)} e_{j,i} x_j^{(\ell)}$, where $e_{j,i}$ is the edge weight of the edge from j to i and $\mathcal{N}(i)$ is the neighborhood of node i. Like before, W_1, W_2 are the matrices with the learnable parameters, and $x_i^{(\ell)}$ is the feature vector for the ith node. For the Mixed geometry version, we modify the update rule to

$$x_i^{(\ell+1)} = HL(x_i^{(\ell)}; W_1) + HL\left(\sum_{j \in \mathcal{N}(i)} e_{j,i} x_j^{(\ell)}; W_2\right).$$

We note again that the aggregation happens in Euclidean space.

GAT A standard Graph Attention layer is given by $x_i^{(\ell+1)} = \alpha_{i,i} W x_i^{(\ell)} + \sum\limits_{j \in \mathcal{N}(i)} \alpha_{i,j} W x_j^{(\ell)}$, where $\alpha_{i,j} = \dfrac{\exp(\text{LeakyReLU}(a^T[W x_i^{(\ell)} \| W x_j^{(\ell)}]))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\text{LeakyReLU}(a^T[W x_i^{(\ell)} \| W x_k^{(\ell)}]))}$ are the attention coefficients. Here the matrix W and the vector a are the learnable parameters. For the Mixed geometry version, we modify the update rule to

$$x_i^{(\ell+1)} = \alpha_{i,i} HL(x_i^{(\ell)}; W) + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} HL(x_j^{(\ell)}; W),$$

where $\alpha_{i,j} = \dfrac{\exp(\text{LeakyReLU}(HL(a^T; [HL(x_i^{(\ell)}; W) \| HL(x_j^{(\ell)}; W)])))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\text{LeakyReLU}(HL(a^T; [HL(x_i^{(\ell)}; W) \| HL(x_k^{(\ell)}; W)])))}$. Note here we again have the attention aggregation happens in Euclidean space.

4 Experiments

The main goal of the experiment is to understand the effect of the geometry on the performance of the model and to understand the dependence on the downstream task, the dataset, as well as the architecture of the model[1]. Specifically, we have two different tasks, node classification, and link prediction, three different architectures, GCN, GC, and GAT for each geometry, Euclidean, Hyperbolic, and Mixed, and nine different datasets of varying homophily. For each (geometry, architecture, dataset) triple, we search over 36 different hyperparameter configurations to find the best neural network. We split the data into train, test, and validation sets to choose the best model for each dataset. We train each combination of model, data, and hyperparameters for ten trials for 100 epochs (200 for link prediction) for each trial. After each epoch, we compute the model's accuracy on the validation set. We then use validation accuracy to pick the stopping epoch for each trial. We then average over the ten trials to pick the best hyperparameter configuration for testing.

4.1 Datasets, Models, and Hyperparameters

We use nine datasets spanning different fields and topics. Specifically, we use citation networks {Cora, Citeseer, PubMed} [44]. Here nodes represent papers, and edges represent citation links. We use airport networks {USA, Brazil, Europe} [33]. Here nodes denote the airports, and edges denote air traffic. Finally, we use website connection network {Cornell, Texas, Wisconsin} [29]. Here nodes correspond to web pages, and edges correspond to hyperlinks between them. The statistics for the different datasets can be seen in the Appendix.

We use nine GNNs with different model architectures and geometries in our experiments. Specifically, the three Euclidean models are GCN [13], GC [25], and GAT [41]. The three Mixed models are described in Section 2.2. The three Hyperbolic models are LGCN [47], the Hyperbolic version of GCN. HGNN [19], which is the Hyperbolic version of GC. Finally, HyboNet [5], which is the Hyperbolic version of attention networks. Note that both LGCN and HGNN adopt the tangent space method, while HyboNet adopts the fully Hyperbolic method.

Both of our experiments will have the same base architecture. We start by feeding the data into an input embedding layer. In the case of Euclidean GNNs, this involves a Euclidean linear layer, while for Mixed GNNs, it's a Hyperbolic linear layer defined in (2). For Hyperbolic models, we use their corresponding embedding layer. Following this, we have $k \in \{2, 3, 4\}$[2] layers of the different GNN architectures with width $h \in \{32, 64, 128, 256\}$. We use ReLU as the activation function. Lastly, for node classification, we proceed through a Euclidean linear layer that maps the output of prior layers to the number of classes for all three geometries. For link prediction, after the last message passing layer, for Euclidean GNNs, we compute the Euclidean inner product between the node features for each edge. For Hyperbolic, we compute the Hyperbolic inner product. Finally, for the Mixed geometry GNNs, we compute the Euclidean inner product ignoring the first coordinate. We ignore the first coordinate as we have factored through the Hyperbolic manifold. Hence it has a special meaning. Finally, we apply a sigmoid function to derive the probabilities of edge existence. The models are then trained using either Adam or Riemannian Adam with a learning rate of $\eta \in \{0.0002, 0.001, 0.005\}$.

4.2 Node Classification

The first question we attempt to answer is whether the architecture or the geometry has a bigger impact on the classification accuracy.

4.2.1 Best Configurations

Table 1 shows the test accuracies for the configuration of hyperparameters with the best average validation error averaged over the ten trials. In order to determine which geometry resulted in the best performance for each model, we computed the average accuracy across the nine datasets. This is reported in Table 2. Here we see that the Mixed geometry models have the highest mean test accuracies for each model architecture. To further validate this, we instead average over the architecture type instead of averaging over the datasets. These results are shown in Table 3. Here we see that the type of dataset affects the results as well. Concretely, we see that for datasets with high

[1]All code and data can be found at Github Link

[2]For Pubmed we could run $k = 2$ due to limited computational resources.

Dataset	Euclidean GCN	Euclidean GC	Euclidean GAT	Mixed GCN	Mixed GC	Mixed GAT	Hyperbolic GCN	Hyperbolic GC	Hyperbolic GAT
Airports Brazil	24 ± 2	42 ± 6	43 ± 3	20 ± 2	39 ± 4	41 ± 3	30 ± 10	40 ± 9	17 ± 2
Airports Europe	43 ± 8	42 ± 4	41 ± 4	46 ± 4	47 ± 2	42 ± 4	44 ± 1	49 ± 3	40 ± 1
Airports USA	46 ± 1	52 ± 1	46 ± 2	47 ± 1	49 ± 1	45 ± 4	47 ± 1	49 ± 3	45 ± 0
Citeseer	65 ± 1	64 ± 1	65 ± 1	66 ± 1	66 ± 1	64 ± 2	68 ± 1	68 ± 1	65 ± 2
Cora	79 ± 1	78 ± 1	79 ± 1	78 ± 1	80 ± 1	78 ± 1	79 ± 1	79 ± 1	80 ± 1
Pubmed	74 ± 2	77 ± 1	74 ± 1	76 ± 1	75 ± 1	74 ± 1	77 ± 1	77 ± 1	77 ± 1
Cornell	45 ± 2	52 ± 2	48 ± 2	47 ± 3	45 ± 3	46 ± 3	42 ± 3	43 ± 2	43 ± 2
Texas	61 ± 2	67 ± 4	61 ± 3	65 ± 2	65 ± 5	64 ± 3	56 ± 2	51 ± 2	56 ± 2
Wisconsin	51 ± 1	61 ± 3	51 ± 3	52 ± 1	69 ± 3	55 ± 2	51 ± 1	52 ± 1	47 ± 3

Table 1: Average accuracy of the best configuration for each model on each dataset along with the standard deviation. Each entry is with respect to ten trials. The darker cell color refers to better accuracy.

	Best Configuration Euclidean	Best Configuration Mixed	Best Configuration Hyperbolic	All Configurations Euclidean	All Configurations Mixed	All Configurations Hyperbolic
GCN	54.31 ± 1.76	$\underline{55.26 \pm 1.83}$	54.97 ± 1.69	48.99 ± 0.32	$\underline{52.05 \pm 0.31}$	$\mathbf{50.75 \pm 0.29}$
GC	$\mathbf{59.32 \pm 1.40}$	$\underline{\mathbf{59.45 \pm 1.48}}$	$\mathbf{56.44 \pm 1.48}$	$\mathbf{54.75 \pm 0.24}$	$\underline{\mathbf{55.66 \pm 0.25}}$	45.84 ± 0.29
GAT	56.27 ± 1.41	$\underline{56.69 \pm 1.44}$	52.19 ± 1.99	46.94 ± 0.34	48.09 ± 0.32	$\underline{50.15 \pm 0.29}$

Table 2: Average accuracy for each architecture geometry pair and the standard error. For the best configuration, we pick the best configuration using the validation error and then average across all ten trials for all nine datasets. For all configurations, we further average over all 36 configurations as well. The best number in each row is underlined, and the best in each column is in bold.

node homophily, such as Citeseer, Cora, and Pubmed, we have that the models with the Hyperbolic geometry perform best. On the other hand, for datasets with low node homophily, such as Cornell, Texas, and Wisconsin, the Mixed geometry models have the best performance. For datasets in the middle, such as the Airport datasets, the models with different geometries perform roughly similarly. Moreover, we see that for the larger datasets such as Cora, Citeseet, and Pubmed, the different models have very similar performance, within 1%. However, for the smaller datasets, we have a much more significant difference, up to 11% for Texas, between the best and worst performing models.

	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
Euclidean	$\mathbf{36 \pm 2}$	42 ± 1	$\mathbf{48 \pm 1}$	65 ± 0	79 ± 0	75 ± 0	$\mathbf{48 \pm 1}$	63 ± 1	54 ± 1
Mixed	34 ± 2	$\mathbf{45 \pm 1}$	47 ± 0	65 ± 0	79 ± 0	75 ± 0	46 ± 1	$\mathbf{65 \pm 1}$	$\mathbf{58 \pm 1}$
Hyperbolic	29 ± 2	44 ± 1	47 ± 0	$\mathbf{67 \pm 0}$	$\mathbf{80 \pm 0}$	$\mathbf{77 \pm 0}$	43 ± 0	54 ± 1	50 ± 1

Table 3: Average accuracy and standard error for the different geometry models for each dataset. The average is for the best configuration. Hence we average over all ten trials and three architecture types. The best value in each column is highlighted in bold.

Next, we explore which architecture has the best performance. Table 2 shows that GraphConv (GC) performs best among the three architectures. We further explore this by averaging across the model's geometry. These accuracies can be seen in Table 4. This further confirms that GraphConv performs the best. Here, in contrast to the geometry case, we see that the homophily of the dataset does not have as much of an effect.

Finally, to answer our question, it can be seen that the difference across architecture types is greater than the difference between the different geometries for the same architecture type. Hence we see that **for node classification, the architecture type has a larger impact than the geometry of the network.**

	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
GCN	25 ± 1	44 ± 1	47 ± 0	$\mathbf{66 \pm 0}$	$\mathbf{79 \pm 0}$	$\mathbf{76 \pm 0}$	45 ± 1	$\mathbf{61 \pm 1}$	51 ± 0
GC	$\mathbf{40 \pm 1}$	$\mathbf{46 \pm 1}$	$\mathbf{50 \pm 0}$	$\mathbf{66 \pm 0}$	$\mathbf{79 \pm 0}$	$\mathbf{76 \pm 0}$	$\mathbf{47 \pm 1}$	$\mathbf{61 \pm 1}$	$\mathbf{61 \pm 1}$
GAT	34 ± 2	41 ± 1	46 ± 0	65 ± 0	$\mathbf{79 \pm 0}$	75 ± 0	45 ± 1	60 ± 1	51 ± 1

Table 4: Average accuracy and standard error for the different architecture models for each dataset. The average is for the best configuration. Hence we average over all ten trials and three geometry types. The best value in each column is highlighted in bold.

4.2.2 Stability Across Hyperparameters

Previously, we looked at the accuracy of the models for the best configuration. However, we can also look at the performance across all configurations. Hence, we average over the configurations instead of picking the best configurations. These results can be seen in Table 2. Here we see the same trend from before. That is, the Mixed geometry models have the best average performance, and that GraphConv has the best performance as well. Further, doing a similar cross-sectional analysis, where we average across the architecture, Table 5 shows that the Mixed geometry networks have the best performance. Unlike the best configuration case, there is less of a correlation between homophily and geometry. Averaging across the geometry, Table 6 shows that GraphConv (GC) has the best performance. Finally, we also see that *the Mixed models have the highest stability across hyperparameters*. This is in the sense that the performance for the Mixed models changed by the smallest when switching from the best configuration to the average across all configurations.

	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
Euclidean	28 ± 0	36 ± 0	42 ± 0	58 ± 0	72 ± 0	72 ± 0	$\mathbf{46 \pm 0}$	60 ± 0	54 ± 0
Mixed	$\mathbf{29 \pm 0}$	$\mathbf{39 \pm 0}$	$\mathbf{45 \pm 0}$	$\mathbf{60 \pm 0}$	$\mathbf{74 \pm 0}$	$\mathbf{73 \pm 0}$	$\mathbf{46 \pm 0}$	$\mathbf{62 \pm 0}$	$\mathbf{56 \pm 0}$
Hyperbolic	24 ± 0	42 ± 0	43 ± 0	55 ± 0	67 ± 1	63 ± 1	43 ± 0	52 ± 0	47 ± 0

Table 5: Average accuracy and standard error for the different geometry models for each dataset. The average is taken over all configurations. Hence we average over all ten trials, three architecture types, and thirty-six configurations. The best value in each column is highlighted in bold.

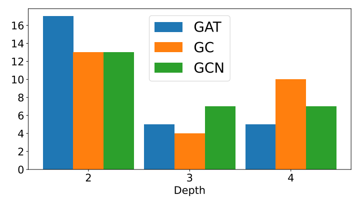
	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
GCN	21 ± 0	36 ± 0	42 ± 0	$\mathbf{60 \pm 0}$	72 ± 0	$\mathbf{72 \pm 0}$	44 ± 0	58 ± 0	51 ± 0
GC	$\mathbf{31 \pm 0}$	$\mathbf{43 \pm 0}$	$\mathbf{46 \pm 0}$	55 ± 0	68 ± 0	64 ± 1	$\mathbf{47 \pm 0}$	$\mathbf{60 \pm 0}$	$\mathbf{55 \pm 0}$
GAT	29 ± 0	38 ± 0	42 ± 0	58 ± 0	$\mathbf{74 \pm 0}$	$\mathbf{72 \pm 0}$	44 ± 0	56 ± 0	51 ± 0

Table 6: Average accuracy and standard error for the different architecture models for each dataset. The average is taken over all configurations. Hence we average over all ten trials, three geometry types, and thirty-six configurations. The best value in each column is highlighted in bold.

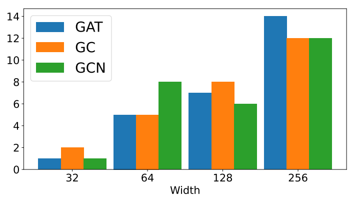
4.2.3 Best Hyper-parameters

Understanding the relationship between the best hyperparameters and the model geometry and architecture is also interesting. Figure 2 shows that the most common depth is depth 2. Thus, we see that shallow networks seem to perform better. However, we see that GraphConv performs well for deeper networks more often than the other two. On the other hand, we see that wider networks perform better than narrower networks. This seems to be true across model architectures. Finally, we see that higher learning rates are also preferred.

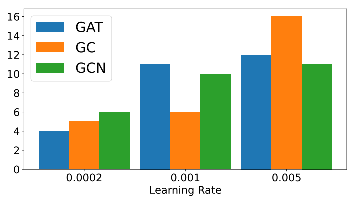
We see something different when we look at the trends for the model's geometry. Here we see that while Euclidean models again prefer shallower networks, the Mixed geometry and Hyperbolic models have a near-equal preference between depth 2 and depth 4 models. This suggests that Mixed geometry and Hyperbolic models might help alleviate the problem of oversmoothing. However, regarding the width, we again see that all three geometries prefer wider networks. Finally, the Mixed geometry model performs better for smaller learning rates, while the Hyperbolic and Euclidean models require



(a) Depth

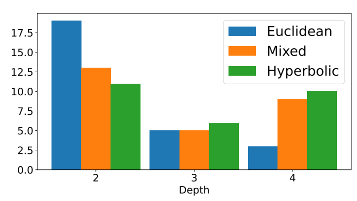


(b) Width

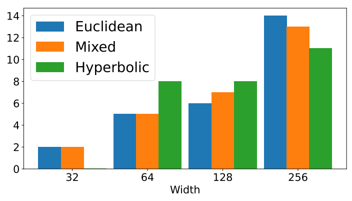


(c) Learning Rate

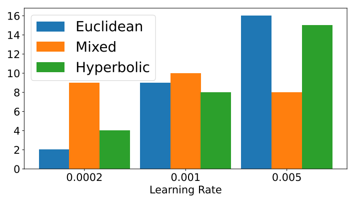
Figure 2: Figure showing the distribution of the best hyperparameters for each architecture type.



(a) Depth



(b) Width



(c) Learning Rate

Figure 3: Figure showing the distribution of the best hyperparameters for each geometry type.

bigger learning rates. This is surprising as the common thought was that Hyperbolic models are less numerically stable. Hence we might expect them to need smaller learning rates.

Node Classification Take-Aways To summarize the main takeaways from the node classification experiment.

1. The model's architecture has a bigger impact than the geometry.
2. The model's geometry is still important, with the Mixed geometry model performing the best. The results are stronger for datasets with low homophily.
3. Shallow and wide models seem to have the best performance.
4. The Mixed geometry model requires smaller learning rates.
5. The Mixed models were the most stable over different hyperparameter configurations.

4.3 Link Prediction

The first question we answer is whether the data geometry or the model's architecture has a more significant impact on the performance. For node classification, we saw that the architecture has a more significant impact, however for link prediction, as seen in Table 7 and Table 8 we have that the geometry has a more significant impact on the performance than the architecture. Here unlike for the node classification experiment, we see that the Hyperbolic models have the best performance. We again validate this by looking at the average over the architecture as seen in Table 9 and by averaging over the geometry in Table 10. We again look at the stability across all configurations, and the same trends are reflected in Table 12 and Table 13. These tables can be found in the Appendix. Further, the analysis for the best hyperparameter configuration can also be found in the Appendix.

This is a surprising result that the task on the data gives us different trends. In principle, the downstream task should affect the geometry and architecture of the neural network. However, it is still interesting that the relative importance of the two quantities change by changing the task. *For link prediction, having hyperbolic geometry for the node features is beneficial.* We know from prior work that it is easier to represent sparse graphs in Hyperbolic space [38]. Hence it is possible that the graph's structure is reflected in the node feature embeddings due to this. *On the other hand, for the node classification problem, we have that both the initial node features and the graph is important. Hence the Mixed geometry might do better.*

To summarize the main takeaways from the link prediction experiment.

1. The model's architecture has a *smaller* impact than the geometry.
2. The model's geometry is important, with Hyperbolic geometry performing the best.

Dataset	Euclidean GCN	Euclidean GC	Euclidean GAT	Mixed GCN	Mixed GC	Mixed GAT	Hyperbolic GCN	Hyperbolic GC	Hyperbolic GAT
Airports Brazil	84 ± 0	86 ± 1	84 ± 1	70 ± 12	83 ± 2	71 ± 7	93 ± 0	84 ± 1	91 ± 0
Airports Europe	84 ± 0	85 ± 0	82 ± 1	66 ± 8	84 ± 0	77 ± 2	91 ± 0	83 ± 0	87 ± 0
Airports USA	91 ± 1	93 ± 0	87 ± 1	73 ± 13	92 ± 1	73 ± 7	96 ± 0	93 ± 0	95 ± 0
Citeseer	94 ± 1	96 ± 0	89 ± 0	74 ± 9	79 ± 2	75 ± 8	99 ± 0	99 ± 0	99 ± 0
Cora	96 ± 0	94 ± 0	94 ± 0	72 ± 8	79 ± 4	71 ± 11	99 ± 0	98 ± 0	99 ± 0
Pubmed	93 ± 3	94 ± 0	88 ± 0	80 ± 6	87 ± 2	78 ± 2	99 ± 0	96 ± 0	99 ± 0
Cornell	68 ± 4	82 ± 5	82 ± 3	60 ± 9	84 ± 2	57 ± 4	81 ± 3	62 ± 4	88 ± 4
Texas	74 ± 4	78 ± 2	75 ± 5	61 ± 10	73 ± 2	58 ± 7	75 ± 3	58 ± 4	79 ± 2
Wisconsin	83 ± 1	81 ± 2	75 ± 1	52 ± 16	74 ± 4	55 ± 7	80 ± 3	78 ± 2	86 ± 1

Table 7: Average AUC of the best configuration for each model on each dataset along with the standard deviation for link prediction. Each entry is with respect to ten trials. The darker the cell color refers to better AUCs.

	Best Configuration Euclidean	Best Configuration Mixed	Best Configuration Hyperbolic	All configurations Euclidean	All configurations Mixed	All configurations Hyperbolic
GCN	85.2 ± 0.97	67.62 ± 1.34	90.28 ± 0.95	76.21 ± 0.31	52.78 ± 0.3	88.09 ± 0.17
GC	$\mathbf{87.65 \pm 0.69}$	$\mathbf{81.46 \pm 0.65}$	83.69 ± 1.51	$\mathbf{80.92 \pm 0.13}$	$\mathbf{72.29 \pm 0.38}$	74.04 ± 0.46
GAT	83.92 ± 0.67	68.35 ± 1.12	$\mathbf{91.52 \pm 0.73}$	73.58 ± 0.39	53.52 ± 0.31	$\mathbf{90.67 \pm 0.13}$

Table 8: Average AUC and standard error for each arch. geometry pair for link prediction. For the best configuration, we pick the best configuration using the validation error and then average across all ten trials for all nine datasets. For all configurations, we further average over all 36 configurations as well. The best number in each row is underlined, and the best in each column is in bold.

3. The Mixed geometry model still requires smaller learning rates.
4. The Hyperbolic model was the most stable over different hyperparameter configurations.

	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
Euclidean	85 ± 0	84 ± 0	90 ± 0	93 ± 1	95 ± 0	92 ± 1	$\mathbf{77 \pm 1}$	$\mathbf{76 \pm 1}$	80 ± 1
Mixed	75 ± 2	76 ± 2	79 ± 2	76 ± 1	74 ± 2	82 ± 1	67 ± 2	64 ± 2	61 ± 2
Hyperbolic	$\mathbf{89 \pm 1}$	$\mathbf{87 \pm 1}$	$\mathbf{95 \pm 0}$	$\mathbf{99 \pm 0}$	$\mathbf{99 \pm 0}$	$\mathbf{98 \pm 0}$	$\mathbf{77 \pm 2}$	71 ± 2	$\mathbf{82 \pm 1}$

Table 9: Average AUC and standard error for the different geometry models for each dataset for link prediction. The average is for the best configuration. Hence we average over all ten trials and three architecture types. The best value in each column is highlighted in bold.

	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
GCN	82 ± 2	80 ± 2	87 ± 2	89 ± 2	89 ± 2	91 ± 2	69 ± 2	70 ± 2	72 ± 3
GC	$\mathbf{84 \pm 0}$	$\mathbf{84 \pm 0}$	$\mathbf{92 \pm 0}$	$\mathbf{91 \pm 2}$	$\mathbf{90 \pm 2}$	$\mathbf{92 \pm 1}$	$\mathbf{76 \pm 2}$	70 ± 2	$\mathbf{78 \pm 1}$
GAT	82 ± 2	82 ± 1	85 ± 2	87 ± 2	88 ± 3	89 ± 2	$\mathbf{76 \pm 3}$	$\mathbf{71 \pm 2}$	72 ± 2

Table 10: Average AUC and standard error for the different architecture models for link prediction. The average is for the best configuration. Hence we average over all ten trials and three architecture types. The best value in each column is highlighted in bold.

5 Conclusion and Future Work

In conclusion, we have introduced a Mixed geometry graph neural network in this paper. After that, we explored the effect of the model's geometry on the model's performance for different tasks, different model architectures, and different datasets. We conduct extensive experiments and find strong and interesting trends.

References

[1] Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. "Sustaining the internet with hyperbolic mapping". In: *Nature communications* 1.1 (2010), p. 62. 1

[2] Shaked Brody, Uri Alon, and Eran Yahav. "How attentive are graph attention networks?" In: *arXiv preprint arXiv:2105.14491* (2021). 4

[3] Michael M Bronstein et al. "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478* (2021). 1

[4] Ines Chami et al. "Hyperbolic Graph Convolutional Neural Networks". In: *Advances in neural information processing systems* 32 (2019), pp. 4869–4880. 1, 2

[5] Weize Chen et al. "Fully Hyperbolic Neural Networks". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5672–5686. DOI: 10.18653/v1/2022.acl-long.389. URL: https://aclanthology.org/2022.acl-long.389. 1–3, 5

[6] Xinyue Cui and Rishi Sonthalia. "Hyperbolic and Mixed Geometry Graph Neural Networks". In: *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*. 2022. 2

[7] Karan Desai et al. "Hyperbolic image-text representations". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 7694–7731. 1

[8] Bhuwan Dhingra et al. "Embedding text in hyperbolic spaces". In: *arXiv preprint arXiv:1806.04313* (2018). 1

[9] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. "Hyperbolic neural networks". In: *Advances in neural information processing systems* 31 (2018). 1

[10] Albert Gu et al. "Learning Mixed-Curvature Representations in Product Spaces". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HJxeWnCcF7. 1, 2

[11] Valentin Khrulkov et al. "Hyperbolic image embeddings". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6418–6428. 1

[12] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106. 3

[13] Thomas Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ArXiv* abs/1609.02907 (2017). 4, 5

[14] Anna Klimovskaia et al. "Poincaré maps for analyzing complex hierarchies in single-cell data". In: *Nature communications* 11.1 (2020), p. 2966. 1

[15] Max Kochurov, Rasul Karimov, and Sergei Kozlukov. "Geoopt: Riemannian Optimization in PyTorch". In: *ArXiv* abs/2005.02819 (2020). URL: https://api.semanticscholar.org/CorpusID:218516676. 3

[16] Dmitri Krioukov et al. "Curvature and temperature of complex networks". In: *Physical Review E* 80.3 (2009), p. 035101. 1

[17] Keegan Lensink, Bas Peters, and Eldad Haber. "Fully hyperbolic convolutional neural networks". In: *Research in the Mathematical Sciences* 9.4 (2022), p. 60. 1

[18] Ya-Wei Eileen Lin et al. "Hyperbolic Diffusion Embedding and Distance for Hierarchical Representation Learning". In: *ArXiv* abs/2305.18962 (2023). 1

[19] Qi Liu, Maximilian Nickel, and Douwe Kiela. "Hyperbolic Graph Neural Networks". In: *NeurIPS*. 2019. 1, 2, 5

[20] Federico Lopez et al. "Symmetric Spaces for Graph Embeddings: A Finsler-Riemannian Approach". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 7090–7101. URL: https://proceedings.mlr.press/v139/lopez21a.html. 2

[21] Federico López et al. "Vector-valued distance and gyrocalculus on the space of symmetric positive definite matrices". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18350–18366. 2

[22] Pascal Mettes et al. "Hyperbolic Deep Learning in Computer Vision: A Survey". In: *arXiv preprint arXiv:2305.06611* (2023). 1

[23] Gal Mishne et al. "The Numerical Stability of Hyperbolic Representation Learning". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 24925–24949. 3

[24] Valter Moretti. "The interplay of the polar decomposition theorem and the Lorentz group". In: *arXiv preprint math-ph/0211047* (2002). 3

[25] Christopher Morris et al. "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks". In: *ArXiv* abs/1810.02244 (2019). 4, 5

[26] Maximilian Nickel and Douwe Kiela. "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry". In: *ICML*. 2018. 1

[27] Maximilian Nickel and Douwe Kiela. "Poincaré Embeddings for Learning Hierarchical Representations". In: *NIPS*. 2017. 1

[28] Yann Ollivier. "Ricci curvature of Markov chains on metric spaces". In: *Journal of Functional Analysis* 256.3 (2009), pp. 810–864. 1

[29] Hongbin Pei et al. "Geom-GCN: Geometric Graph Convolutional Networks". In: *ArXiv* abs/2002.05287 (2020). URL: https://api.semanticscholar.org/CorpusID:210843644. 5

[30] Hongbin Pei et al. "Geom-gcn: Geometric graph convolutional networks". In: *arXiv preprint arXiv:2002.05287* (2020). 12

[31] Wei Peng et al. "Hyperbolic Deep Neural Networks: A Survey". In: *IEEE transactions on pattern analysis and machine intelligence* PP (2021). 1

[32] Eric Qu and Dongmian Zou. "Autoencoding Hyperbolic Representation for Adversarial Generation". In: *arXiv preprint arXiv:2201.12825* (2022). 3

[33] Leonardo F. R. Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. "struc2vec: Learning Node Representations from Structural Identity". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017). URL: https://api.semanticscholar.org/CorpusID:3948366. 5

[34] Frederic Sala et al. "Representation Tradeoffs for Hyperbolic Embeddings". In: *Proceedings of machine learning research* 80 (2018), pp. 4460–4469. 1

[35] Yuval Shavitt and Tomer Tankel. "Hyperbolic Embedding of Internet Graph for Distance Estimation and Overlay Construction". In: *IEEE/ACM Trans. Netw.* 16.1 (Feb. 2008), pp. 25–36. ISSN: 1063-6692. DOI: 10.1109/TNET.2007.899021. URL: https://doi.org/10.1109/TNET.2007.899021. 1

[36] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. "Hyperbolic neural networks++". In: *arXiv preprint arXiv:2006.08210* (2020). 1

[37] Rishi Sonthalia, Anna C Gilbert, and Matthew Durham. "CubeRep: Learning Relations Between Different Views of Data". In: *Topological, Algebraic and Geometric Learning Workshops 2022*. PMLR. 2022, pp. 298–303. 2

[38] Rishi Sonthalia and Anna C. Gilbert. "Tree! I am no Tree! I am a Low Dimensional Hyperbolic Embedding". In: *NeurIPS*. 2020. 1, 8

[39] Domenico Tortorella and Alessio Micheli. "Is Rewiring Actually Helpful in Graph Neural Networks?" In: *ArXiv* abs/2305.19717 (2023). URL: https://api.semanticscholar.org/CorpusID:258987350. 1

[40] Nicolas Garcia Trillos and Melanie Weber. "Continuum Limits of Ollivier's Ricci Curvature on data clouds: pointwise consistency and global lower bounds". In: *arXiv preprint arXiv:2307.02378* (2023). 1

[41] Petar Velickovic et al. "Graph Attention Networks". In: *ArXiv* abs/1710.10903 (2017). URL: https://api.semanticscholar.org/CorpusID:3292002. 5

[42] Xiaoyu Xu et al. "Joint hyperbolic and Euclidean geometry contrastive graph neural networks". In: *Inf. Sci.* 609 (2022), pp. 799–815. 2

[43] Menglin Yang et al. "Hyperbolic Graph Neural Networks: A Review of Methods and Applications". In: *ArXiv* abs/2202.13852 (2022). 1

[44] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. "Revisiting Semi-Supervised Learning with Graph Embeddings". In: *ArXiv* abs/1603.08861 (2016). 5

[45] Tao Yu and Christopher De Sa. "Numerically Accurate Hyperbolic Embeddings Using Tiling-Based Models". In: *NeurIPS*. 2019. 3

[46] Yiding Zhang et al. “Hyperbolic Graph Attention Network”. In: *ArXiv* abs/1912.03046 (2021). 1, 2

[47] Yiding Zhang et al. “Lorentzian Graph Convolutional Networks”. In: *Proceedings of the Web Conference 2021* (2021). URL: `https://api.semanticscholar.org/CorpusID:233241168`. 1, 2, 5

[48] Weichen Zhao et al. “Modeling Graphs Beyond Hyperbolic: Graph Neural Networks in Symmetric Positive Definite Matrices”. In: *ArXiv* abs/2306.14064 (2023). 2

[49] Jiong Zhu et al. “Beyond homophily in graph neural networks: Current limitations and effective designs”. In: *Advances in neural information processing systems* 33 (2020), pp. 7793–7804. 12

APPENDIX

A Data Statistics

The statistics for the datasets, including node [49] and edge homophily [30], can be seen in Table 11.

Dataset	# Nodes	# Edges	Node Homophily	Edge Homophily
Cora	2,708	10,556	0.83	0.81
Citeseer	3,327	9,104	0.71	0.74
PubMed	11,717	88,648	0.79	0.80
Airports USA	1,190	13,599	0.34	0.70
Airports Brazil	131	1,038	0.28	0.47
Airports Europe	399	5,995	0.20	0.40
Cornell	183	298	0.11	0.13
Texas	183	325	0.07	0.11
Wisconsin	251	515	0.17	0.20

Table 11: Data Statistics

B Link Prediction Tables

In this section, we include further results from our link prediction experiments, similar to the node classification results explored in Section 4.2 of the main paper that were not included in the link prediction analysis in Section 4.3

B.1 AUC for all configurations

While in the main paper we discussed the results of analyzing the best configurations, here we explore results from aggregating across all configurations.

B.1.1 Aggregating Across Geometry

Here, we see that Hyperbolic models out-perform their Euclidean and Mixed geometry counterparts, just as we saw in section 4.4 of the main paper. We also note that AUC values for the citation network datasets are lower than the airports and publication datasets, applicable for all geometries. We notice a correlation with the varying levels of homophily of the datasets (see Appendix A).

	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
Euclidean	80 ± 0	79 ± 0	85 ± 0	86 ± 0	87 ± 0	86 ± 0	$\mathbf{74 \pm 0}$	$\mathbf{71 \pm 0}$	74 ± 0
Mixed	66 ± 0	68 ± 0	70 ± 0	65 ± 0	63 ± 0	68 ± 1	64 ± 0	60 ± 0	57 ± 0
Hyperbolic	$\mathbf{88 \pm 0}$	$\mathbf{83 \pm 0}$	$\mathbf{93 \pm 0}$	$\mathbf{98 \pm 0}$	$\mathbf{98 \pm 0}$	$\mathbf{88 \pm 1}$	$\mathbf{74 \pm 0}$	$\mathbf{71 \pm 0}$	$\mathbf{81 \pm 0}$

Table 12: Average AUC and standard error for the different geometry models for each dataset for link prediction. The average is taken over all configurations. Hence we average over all ten trials, three architecture types, and thirty-six configurations. The best value in each column is highlighted in bold.

B.1.2 Aggregating Across Architecture

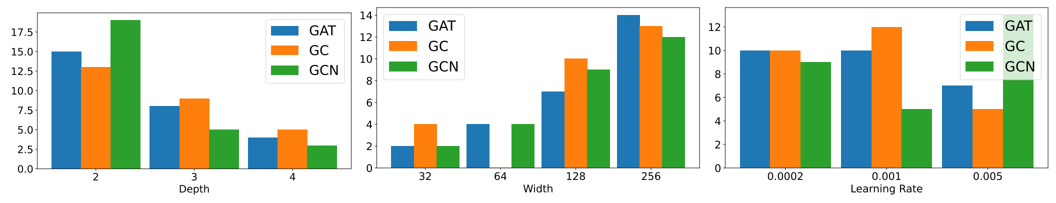
	Brazil	Europe	USA	Citeseer	Cora	Pubmed	Cornell	Texas	Wisconsin
GCN	74 ± 0	73 ± 0	80 ± 0	83 ± 1	82 ± 1	$\mathbf{81 \pm 1}$	68 ± 0	65 ± 0	69 ± 0
GC	$\mathbf{83 \pm 0}$	$\mathbf{81 \pm 0}$	$\mathbf{89 \pm 0}$	$\mathbf{85 \pm 0}$	$\mathbf{84 \pm 0}$	$\mathbf{81 \pm 1}$	$\mathbf{73 \pm 0}$	$\mathbf{69 \pm 0}$	$\mathbf{73 \pm 0}$
GAT	76 ± 0	76 ± 0	80 ± 0	81 ± 1	82 ± 1	$\mathbf{81 \pm 1}$	72 ± 0	68 ± 0	70 ± 0

Table 13: Average AUC and standard error for the different architecture models for link prediction. The average is taken over all configurations. Hence we average over all ten trials, three architecture types, and thirty-six configurations. The best value in each column is highlighted in bold.

We see that the models of the GraphConv (GC) outperform the other architectures. For the larger datasets, the difference in AUC values between architectures is lower, whereas for the smaller datasets the differences are more pronounced.

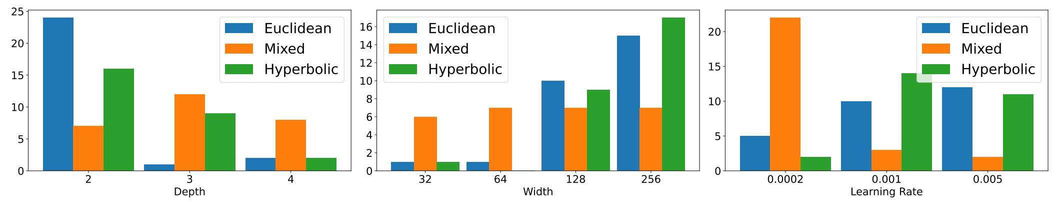
B.2 Best Hyper-parameters

Regarding the best hyperparameters, we see the same result that the mixed geometry model again prefers deeper networks. Hence again suggesting that the mixed geometry models might help alleviate oversmoothing. Here we see that the mixed no longer prefers a wider network and seems to have no width preference. Finally, as before, the mixed geometry networks need lower learning rates, with again the hyperbolic geometry needing the largest learning rates.



(a) Depth **(b)** Width **(c)** Learning Rate

Figure 4: Figure showing the distribution of the best hyperparameters for each architecture type.



(a) Depth **(b)** Width **(c)** Learning Rate

Figure 5: Figure showing the distribution of the best hyperparameters for each geometry type.

C Run-time Experiments

In this section we use our node classification experiments to explore another dimension of analysis: run-time. We divide this into two parts: an evaluation of time-stamps and convergence epochs for the trials.

C.1 Time-stamps

In Section 3.1 of the main paper, we suggested that one of the benefits of our Mixed models was that fact that we do not need exponential and logarithmic maps to parameterize the network. We claim that this speeds up the training process as compared to those Hyperbolic counterparts where it is indeed necessary. We substantiate our claims by comparing time-stamps for the Mixed and Hyperbolic models of the GraphConv (GC) and Graph Convolutional Neural Network (GCN) architectures. We do not observe the time-stamps of the Attention Network models given that the Hyperbolic Attention architecture model uses the Tangent space. Note that all experiments are run on Google Colab with a T4 GPU and 50GB of RAM. We consider the average across 5 trials of the time the largest model (4 layers, 256 dimensions, learning rate of 5e-3) takes to run 100 epochs.

	GC		GCN	
Dataset	Mixed	Hyperbolic	Mixed	Hyperbolic
Airports Brazil	15.77	12.08	12.38	14.85
Airports Europe	16.09	17.22	12.75	14.87
Airports USA	16.13	37.24	12.98	15.01
Cora	17.77	72.28	15.1	17.0
CiteSeer	19.02	92.31	15.11	21.46
Cornell	16.43	17.2	12.93	14.55
Texas	16.08	17.2	12.57	14.95
Wisconsin	16.11	17.22	12.92	14.99

Table 14: Run-time of models for different experiments in seconds. Values obtained through averaging across 5 trials of configuration (4 layers, 256 dimensions, learning rate or 5e-3) over 100 epochs.

Here, we find that for the smallest dataset – Airprots Brazil – the Mixed geometry GC models took longer to run than their Hyperbolic counterparts. However, as the datasets size increased, we observe that Mixed geometry GC models ran faster. Specifically, for the largest that we used for our comparison – Cora and CiteSeer – we note 4x and 5x improvements on run-time respectively. On the other hand, comparing Mixed GCN models with Hyperbolic versions, we see that the former run faster for every dataset tested on. We note that for all the above experiments, the Euclidean models take about 6-9 seconds on average.

C.2 Stopping Epoch

For the purposes of our paper, we define the stopping epoch as the first instance of where the validation accuracy is maximized across training for a given experiment – a unique combination of task, dataset, model architecture, number of layers, number of dimensions, learning rate. For each combination of dataset and model, we calculate the converging epoch by averaging the stopping epoch across all trials of every configuration. The results are as follows:

	Airports			Publications			Citations		
Model	Brazil	Europe	USA	CiteSeer	Cora	PubMed	Cornell	Texas	Wisconsin
GAT	57.5	45.7	60.8	32.1	38.4	55.4	61.2	28.3	49.1
HGAT	70.7	49.8	59.5	23.9	29.0	24.2	66.9	52.7	77.5
HyboNet	26.6	36.2	47.6	46.0	46.6	45.2	4.9	21.5	30.0
GC	58.1	58.4	67.4	59.4	63.1	72.4	51.4	32.3	55.1
HGC	21.4	22.6	20.5	33.0	47.6	42.6	25.9	36.8	51.0
HGNN	48.5	59.3	55.3	60.9	56.3	61.2	17.3	56.8	41.0
GCN	53.2	54.8	65.1	40.3	54.6	61.1	16.1	51.1	37.0
HGCN	62.0	59.9	61.5	20.3	29.0	25.5	12.0	40.9	31.4
LGCN	37.8	48.8	54.3	59.2	62.0	66.6	22.1	38.0	48.7

Table 15: Stopping epochs per experiment. Calculated by averaging the stopping epoch across all trials for all configurations of a combination of dataset and model.

We note that for the largest datasets (Cora, CiteSeer, PubMed) – for which run-time is most consequential – we see that the Mixed models convergence significantly faster than the Euclidean and

	Euclidean			Mixed			Hyperbolic		
	GCN	GC	GAT	GCN	GC	GAT	GCN	GC	GAT
Stopping Epoch	48.1	38.1	48.6	57.5	33.5	50.7	47.61	50.5	33.8

Table 16: Stopping epochs per experiment. Calculated by averaging the stopping epoch across all trials for all configurations of a combination of dataset and model. Further averaged across the datasets to yield a singular value of comparison.

Hyperbolic counterparts for every architecture. We further aggregate across the datasets to compare across models at a high level. We see that for the GCN and GC models, the Mixed models converge noticeably faster whereas for the GAT models, the Hyperbolic models converge the fastest:

D Sensitivity to Hyper-parameters

In many cases, doing a comprehensive hyperparameter search is expensive and not feasible. In this case, we care about stability across hyperparameters. In other words, the propensity with which a model's accuracy deviates with changing hyperparameters. Here, we study these metrics by computing the standard deviation of the test accuracies across all 36 configurations for each combination of dataset and models. We finally report the mean by aggregating across all datasets.

	Airports			Publications			Citations			
Model	Brazil	Europe	USA	CiteSeer	Cora	PubMed	Cornell	Texas	Wisconsin	Mean
GAT	0.113	0.035	0.061	0.037	0.057	0.038	0.018	0.018	0.019	0.044
HGAT	0.045	0.023	0.022	0.035	0.028	0.012	0.008	0.023	0.018	0.023
HyboNet	0.022	0.027	0.021	0.055	0.066	0.023	0.023	0.017	0.025	0.031
GC	0.058	0.049	0.047	0.039	0.037	0.033	0.032	0.044	0.034	0.041
HGC	0.052	0.038	0.027	0.051	0.046	0.009	0.028	0.038	0.036	0.036
HGNN	0.057	0.019	0.028	0.149	0.192	0.120	0.003	0.013	0.018	0.066
GCN	0.032	0.081	0.068	0.078	0.121	0.078	0.008	0.021	0.029	0.057
HGCN	0.023	0.066	0.019	0.034	0.022	0.007	0.008	0.016	0.023	0.024
LGCN	0.045	0.048	0.048	0.111	0.157	0.065	0.005	0.011	0.039	0.058

Table 17: Standard deviation of accuracies for node classifications across configurations. For each combination of dataset and model, we compile the average test accuracy across trials for each configuration. We then report the standard deviation of these average test accuracy values across all configurations to obtain a metric signifying the sensitivity to hyper-parameters. We finally calculate the mean for each row (averaged across datasets) to compare between the model variants.

We observe that the Mixed models have the smallest standard deviation for each architecture. The margins of difference is highest when looking at the larger datasets – CiteSeer, Cora, and PubMed. We conclude that the Mixed models have the most stable accuracies across hyper-parameters.