Dynamic Curvature Optimization for Hyperbolic GCNs

Valentina Simon
Department of Statistics & Data Science
Yale University
valentina.simon@yale.edu

Arjan Kohli
Department of Statistics & Data Science
Yale University
arjan.kohli@yale.edu

1 Introduction

Hyperbolic Graph Convolutional Networks (GCNs) have emerged as powerful tools for modeling hierarchical and complex relational structures in graph data. Unlike their Euclidean counterparts, hyperbolic GCNs leverage the naturally occurring negative curvature of hyperbolic geometry to embed hierarchies more efficiently, enabling lower-distortion representations of tree-like graphs and large-scale knowledge structures. In comparison to state-of-the-art GCNs, HGCNs have shown an error reduction of up to 63.1% in ROC AUC for link prediction, demonstrating the value of hyperbolic embedding structures [2]. Despite these advantages, however, most existing hyperbolic GCN architectures either fix curvature globally or adjust it per layer, limiting the interpretability and flexibility of the learned embedding geometry.

This work addresses the problem of dynamically tuning a single, global curvature parameter for hyperbolic GCNs. By treating curvature as a learnable model parameter rather than a static hyperparameter or a layer-specific variable, we allow the network to continuously adapt the global geometric space to the intrinsic structure of the data. This direct, unified approach enables clearer insight into how curvature correlates with the hierarchical properties of a given dataset, providing a more interpretable mapping between the geometry of the embedding space and the underlying graph topology.

To achieve this, we integrate a trainable curvature parameter into a hyperbolic GCN and apply three distinct Riemannian optimization methods, examining their effects on both embedding quality and curvature convergence. We evaluate our model on four datasets that exhibit a range of Gromov δ-hyperbolicities, thereby providing a controlled setting to investigate the relationship between the learned curvature and the intrinsic hyperbolicity of the data. Compared to existing approaches that adjust curvature per layer, our single-curvature strategy simplifies the curvature-hyperbolicity relationship, offering a more direct interpretation of how global geometry aligns with hierarchical structure.

Our results demonstrate that empowering the model to learn a single global curvature parameter leads to performance improvements in tasks such as link prediction, often reducing the dimensionality requirements and enhancing representational efficiency. This unified curvature optimization not only bolsters predictive accuracy but also yields deeper insights into the geometric foundations of hyperbolic embeddings. By coupling curvature learning directly to data properties, we advance the capabilities of hyperbolic representation learning and further illuminate the intricate interplay between geometry and structure in graph data.

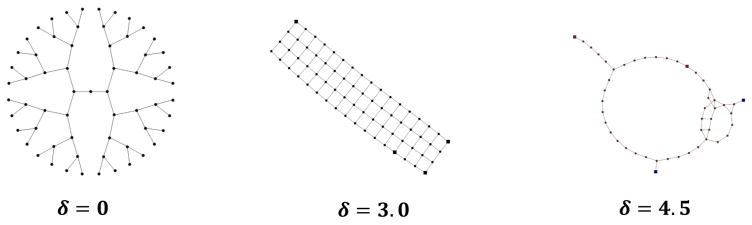


Figure 1: δ-hyperbolicity for different graph structures. A lower score is indicative of a more hyperbolic graph (ie. hierarchical structure). Trees have a score of zero.

2 Related Work

The introduction of hyperbolic embeddings for capturing hierarchical data can be traced back to Nickel and Kiela [6], who proposed Poincaré embeddings to learn latent hierarchical representations more compactly than Euclidean embeddings. Subsequent works [7, 8] have extended hyperbolic representations to various tasks, including learning hyperbolic word embeddings and modeling complex hierarchical ontologies. These advances established the foundational intuition that negatively curved spaces naturally mirror tree-like structures and thus reduce distortion when embedding hierarchical or scale-free graphs.

Using these insights, Chami et al. [2] introduced Hyperbolic Graph Convolutional Networks (HGCNs), employing Riemannian geometry-based operations (e.g., exponential and logarithmic maps) to directly incorporate curvature into GNNs. This approach was among the first to show that incorporating hyperbolic geometry at the layer level leads to improved representations of data with hierarchical or power-law degree distributions. However, their per-layer curvature tuning increased complexity and made it challenging to interpret how curvature related to the dataset's global structure. Similarly, Ganea et al. [8] explored hyperbolic neural networks, focusing on fundamental building blocks such as gyrovector additions and Möbius transformations, providing a mathematical toolkit for constructing hyperbolic analogs of standard neural network layers.

Zhang et al. [5] integrated attention mechanisms into hyperbolic space, leading to Hyperbolic Graph Attention Networks (HGATs). By combining the expressive power of attention with the compactness of hyperbolic embeddings, these models aimed to capture complex relational patterns more efficiently. Existing work tends to fix curvature values or vary them per layer, limiting clarity on the interplay between global curvature and hierarchical properties of the data. Recent efforts have further refined curvature modeling strategies to better align with Gromov hyperbolic structures [9].

Beyond conventional GCNs, Bose and Das [1] applied hyperbolic geometry to graph transformers (Hype-GT), using hyperbolic positional encodings to represent global structural information. Transformers inherently capture long-range dependencies, and integrating hyperbolic signals further improves their capacity to represent hierarchical relationships. Hyperbolic embeddings have also influenced other domains. Yang et al. [4] introduced hyperbolic fine-tuning strategies for large language models, illustrating that the benefits of hyperbolic geometry—such as more faithful hierarchical representations—extend beyond graph-centric tasks. Still, adapting these insights to GNN-based problems requires careful consideration of manifold constraints and data-specific characteristics.

3 Methods

Building on previous efforts, our work advances hyperbolic representation learning along three key dimensions.

First, while early Hyperbolic Graph Convolutional Networks (HGCNs) [2] and Hyperbolic Graph Attention Networks (HGATs) [5,6] demonstrated the benefits of embedding graph data in negatively curved spaces, they typically either fixed curvature or allowed it to vary per layer. This layer-specific tuning can introduce complexity and obscure the relationship between global graph structure and the model's geometric parameters. In contrast, we simplify and clarify this relationship by introducing a single, globally trainable curvature parameter. By doing so, we isolate curvature as a global characteristic of the embedding space, making it easier to interpret and correlate directly with graph hyperbolicity measures.

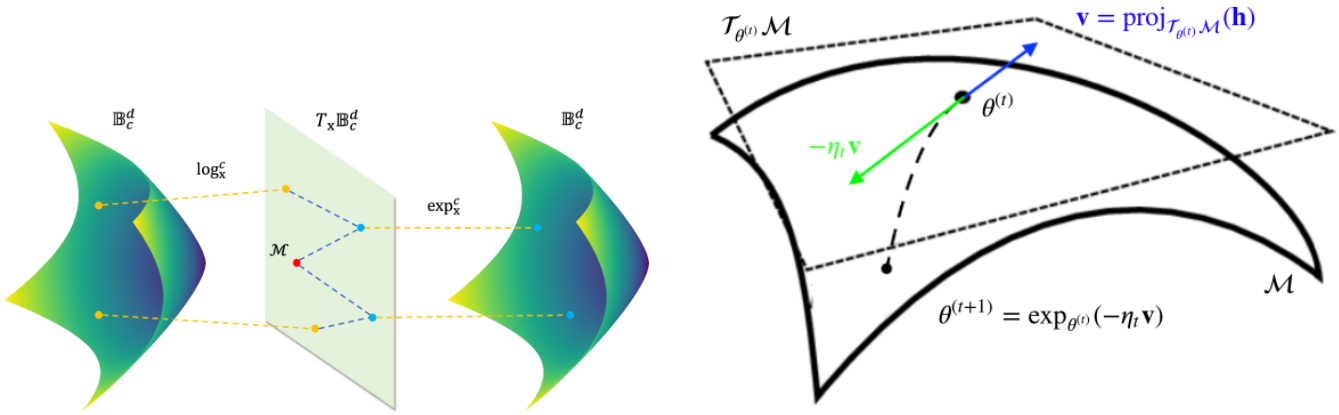
Second, while prior works primarily explored a limited range of optimization methods or relied on standard Riemannian optimizers, we systematically investigate three distinct approaches to curvature optimization. This comprehensive comparison reveals how different optimization techniques affect the stability of curvature learning, leading to improved insights into best practices for training hyperbolic GNNs.

Finally, prior studies often focused on performance gains without thoroughly examining how curvature correlates with the underlying properties of the datasets. By evaluating our method on multiple graphs with varying Gromov δ-hyperbolicities, we more concretely connect learned curvature values to the intrinsic hierarchical structure of the data. In this way, our work not only enhances representational fidelity but also contributes to a deeper theoretical understanding of how geometry and graph topology interact, paving the way for more interpretable, robust, and targeted hyperbolic GNN architectures.

3.1 Hyperbolic GCN

Model Architecture: Using the Geopt library, we implement a hyperbolic Graph Convolutional Network (HGCN) following the general framework introduced by Chami et al. [2]. Our architecture uses the Poincaré model of hyperbolic space for embedding node features and performing message passing. Unlike the original HGCN, which learns a distinct curvature parameter per layer, we treat curvature as a single global parameter shared across all layers. This design choice is intended to provide a more direct interpretation of how curvature, as a scalar hyperbolic metric parameter, relates to the global geometry of the graph representation and the underlying dataset properties. By maintaining a single curvature value, we reduce the complexity of the model and enable clearer analyses of the curvature-hyperbolicity relationship.

Curvature as a Tunable Parameter: We introduce a trainable curvature parameter c that can be updated during backpropagation. To investigate the relationship between curvature and Gromov δ-hyperbolicity, we evaluate our model on four publicly available graph datasets of different hyperbolicities and observe the learned curvature. Each dataset is preprocessed following standard protocols, including normalization of node features and use of given train/validation/test splits. Curvature updates affect the geodesic distances and exponential/logarithmic mappings in the Poincaré model, influencing how node features are aggregated and embedded.



(a) Hyperbolic GCN Schematic

(b) Riemannian Optimization

Figure 2: Visual explanation of Hyperbolic space and impact of curvature in GCN performance. Stochastic gradient descent pictured on a Riemannian manifold.

3.2 Optimization

We apply three distinct optimization methods for updating the curvature parameter and the network weights: (1) Riemannian Adam, (2) Riemannian SGD, and (3) mixed-precision Riemannian optimizer. Each optimizer operates on the Poincaré manifold for node embeddings and on the Euclidean manifold for curvature, using appropriate retractions and exponential maps to ensure constraints on curvature values are preserved (e.g., ensuring negative curvature: c < 0). Hyperparameters (learning rate, weight decay, and patience for early stopping) are selected via grid search on the validation set for each dataset.

3.2.1 Riemannian manifold optimization

In a Riemannian manifold $(\mathcal{M}, g)$, the key operations are:

- **Riemannian Gradient**: $\nabla_{\mathcal{M}} f(x)$ projects the Euclidean gradient onto the tangent space $T_x\mathcal{M}$.
- **Exponential Map**: $\text{Exp}_x(v) : T_x\mathcal{M} \to \mathcal{M}$ maps a tangent vector v to the manifold.
- **Retraction**: $R_x(v)$ approximates the exponential map for computational efficiency.

3.2.2 Riemannian SGD Optimizer

Let $x_t \in \mathcal{M}$ denote the current parameters on the manifold. At iteration t, Riemannian SGD proceeds as follows:

Initialization: Set $t = 0$.

Update Steps:

1. Compute the **Riemannian gradient**:
$$g_t = \nabla_{\mathcal{M}} f(x_t). \tag{1}$$
2. Compute the **tangent space update vector**:
$$u_t = -\alpha g_t, \tag{2}$$
where α is the learning rate.
3. Update the parameters using the **exponential map**:
$$x_{t+1} = \text{Exp}_{x_t}(u_t), \tag{3}$$

3.2.3 Riemannian Adam Optimizer

Let $x_t \in \mathcal{M}$ denote the current parameters on the manifold. At iteration t, the Riemannian Adam optimizer proceeds as follows:

Initialization: Set $m_0 = 0$, $v_0 = 0$, and $t = 0$.

Algorithm:

1. Compute the **Riemannian gradient**:
$$g_t = \nabla_{\mathcal{M}} f(x_t). \tag{4}$$
2. Update the **biased first and second moment estimates**:
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{5}$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t \odot g_t, \tag{6}$$
where $\odot$ denotes element-wise multiplication.
3. Apply **bias correction**:
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{7}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{8}$$
4. Compute the **tangent space update**:
$$u_t = -\alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{9}$$
where α is the learning rate and ϵ is a small constant.
5. Update the parameters using the **exponential map**:
$$x_{t+1} = \text{Exp}_{x_t}(u_t), \tag{10}$$

3.2.4 Mixed Precision Riemannian Adam Optimizer

Mixed precision optimization uses lower-precision (e.g., 16-bit floating point) computations during training to reduce memory usage and improve computational efficiency. Gradient scaling ensures numerical stability by temporarily scaling gradients before applying optimization steps. The **Mixed Precision Riemannian Adam Optimizer** integrates gradient scaling into the Riemannian Adam framework.

Let $x_t \in \mathcal{M}$ denote the parameters on the manifold at iteration t. The mixed precision algorithm proceeds as follows:

Initialization: Set $m_0 = 0$, $v_0 = 0$, $t = 0$, and initialize the gradient scale factor scale.

Algorithm:

1. Compute the **scaled Riemannian gradient**:
$$g_t = \text{scale} \cdot \nabla_{\mathcal{M}} f(x_t). \tag{11}$$
2. Update the **biased moment estimates**:
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{12}$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t). \tag{13}$$
3. Apply **bias correction**:
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{14}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{15}$$
4. Compute the **scaled tangent space update**:
$$u_t = -\alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{16}$$
5. Unscale the update by dividing by scale:
$$u_t' = \frac{u_t}{\text{scale}}. \tag{17}$$
6. Update the parameters using the **exponential map**:
$$x_{t+1} = \text{Exp}_{x_t}(u_t'). \tag{18}$$

Gradient Scale Adjustment: After applying the update, we adjusted the gradient scale dynamically based on overflow checks to maintain numerical stability during training.

This mixed precision Riemannian Adam method leverages the advantages of both Riemannian optimization and mixed precision training. By scaling and unscaling gradients, it maintains stability while optimizing over a Riemannian manifold.

3.2.5 Application to Hyperbolic GCNs

For hyperbolic GCNs, the above steps are adapted to Poincaré ball:

- The tangent space projection uses the Riemannian metric $g_x = \frac{2}{1-\|x\|^2}$.
- The exponential map is defined as:

$$\text{Exp}_x(v) = \tanh\left(\frac{\|v\|}{1-\|x\|^2}\right)\frac{v}{\|v\|} + x. \tag{19}$$

Evaluation Metrics and Analysis: We use standard link prediction accuracy on the test sets to gauge the predictive performance of the model. To assess the relationship between curvature and hyperbolicity, we record the final learned curvature values across runs and examine their correlations with the measured δ-hyperbolicity of each dataset. This analysis is complemented by ablation studies comparing our single-curvature approach to a per-layer curvature approach, as in [2]. We discuss how a single global curvature might enhance interpretability and simplify the curvature-hyperbolicity mapping, potentially offering more stable curvature learning and allowing for clearer diagnostic insights into the model's geometric inductive bias.

4 Experiments

4.1 Datasets

We used a total of four datasets with varying δ-hyperbolicity. WordNet is a comprehensive lexical database for the English language, organizing words into synsets that capture synonyms and various semantic relationships such as hypernymy and meronymy. PubMed is a vast repository of biomedical literature, encompassing millions of citations and abstracts from life sciences and medical journals. FB15K is a widely used subset of the Freebase knowledge graph, containing approximately 15,000 entities and 1,345 relation types, which is instrumental for tasks such as knowledge graph completion, link prediction, and semantic reasoning. The Diseases dataset encompasses hierarchical information about various diseases, including classifications, symptoms, genetic associations, and treatment options.

Table 1: Statistics for all datasets used. Calculated hyperbolicity from a subgraph of 20,000 nodes to manage computational complexity and memory requirements.

Dataset	**δ-hyperbolicity**	**Number of Entities**	**Number of Relations**
Wordnet	5.5	40945	19
Pubmed	3.5	19717	3
FB15K	1.5	14541	237
Diseases	0	2665	2

4.2 Results

Training Details: For training on all datasets, we set the hidden dimension to 128 and chose a starting curvature parameter of c = 1.0 for the hyperbolic manifold. Experiments are run using a fixed random seed of 42 to ensure reproducibility and allow fair comparisons. Data preprocessing included converting triple-based graph data into numerical head, tail, and relation indices, constructing adjacency matrices for GCN message passing, and ensuring all embeddings are initialized on the Poincaré ball manifold.

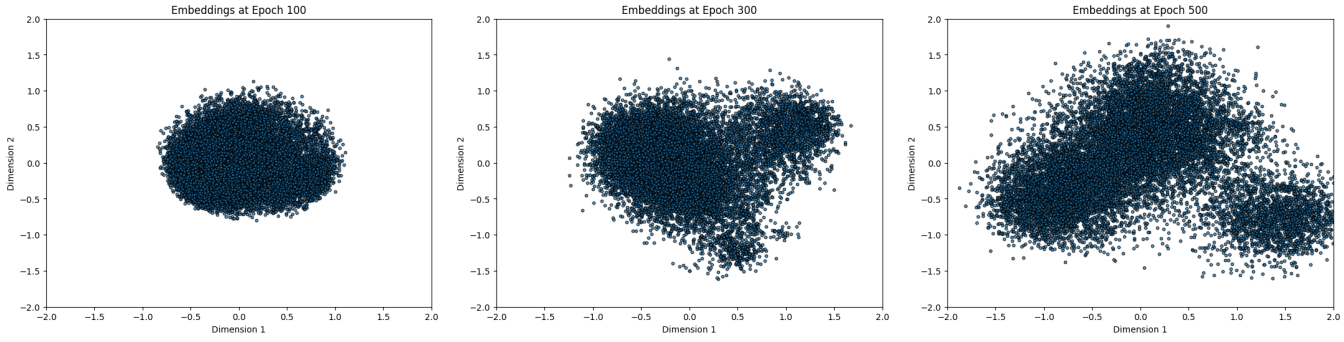


Figure 3: PCA of node embeddings at 100, 300, and 500 epochs. Structure is observed as model converges to a curvature and more node separation is visible in low dimensions.

Optimizers and Learning Rates: We used (1) Riemannian Adam, (2) Riemannian SGD, and (3) mixed-precision Riemannian optimizer. We adjusted learning rate such that we saw convergence to optimal test accuracy for each experiment.

Table 2: Experiment Learning Rates

Dataset	**Riemannian Adam**	**Riemannian SGD**	**Mixed-Precision Riemannian**
Wordnet	1×10^{-2}	1×10^{-1}	1×10^{-1}
Diseases	1×10^{-4}	1×10^{-3}	1×10^{-4}
FB15K	1×10^{-2}	1×10^{-1}	1×10^{-3}
Pubmed	1×10^{-4}	5×10^{-2}	1×10^{-2}

Table 3: HGCN Results for various datasets and configurations. Note that starting c is always 1 and is negative curvature.

Dataset	Mean Best Test Acc.	Mean Curvature
Diseases		
SGD	0.9142 ± 0.0068	0.9953 ± 0.0064
ADAM	0.9113 ± 0.0123	0.9504 ± 0.0045
Mixed-Precision	0.9246 ± 0.0088	0.9535 ± 0.0013
Wordnet		
SGD	0.4533 ± 0.0016	0.2827 ± 0.0153
ADAM	0.7678 ± 0.0146	0.0186 ± 0.0041
Mixed-Precision	0.7628 ± 0.0061	0.0016 ± 0.0001
Pubmed		
SGD	0.8576 ± 0.0004	0.8681 ± 0.0523
ADAM	0.8611 ± 0.0021	0.8188 ± 0.0108
Mixed-Precision	0.8648 ± 0.0041	0.7403 ± 0.0263
FB15K		
SGD	0.3906 ± 0.0064	0.0013 ± 0.00002
ADAM	0.9434 ± 0.0003	0.1320 ± 0.0251
Mixed-Precision	0.9422 ± 0.0031	0.1299 ± 0.0015

Analysis: We present the HGCN curvature optimization results in Table 2, calculating the mean best test accuracy and mean curvature for three runs along with the standard deviation.

There seems to be a relationship between hierarchical data and accuracy of our model. Datasets with lower δ-hyperbolicity (Diseases and FB15K) yielded higher accuracy than those with higher δ-hyperbolicity (Pubmed and Wordnet). Across the four datasets, the training curves (Figure 4) reveal how the HGCN with a single learnable curvature parameter adapts differently depending on the underlying structure and δ-hyperbolicity of the data. More hierarchical datasets, such as Diseases (δ=0), show more stable convergence with improved validation accuracy. Here, the hyperbolic geometry effectively captures the tree-like structure, enabling faster and more pronounced gains. On the other hand, Wordnet (δ=5.5), with higher δ-hyperbolicity, is more "Euclidean-like" and less naturally suited to hyperbolic embeddings. In this setting, the training still reduces loss, but improvements in accuracy are less dramatic. Intermediate datasets such as Pubmed (δ=3.5) and FB15k (δ=1.5) lie between these extremes, displaying moderate improvements in accuracy and steady convergence of curvature. This suggests that a single learned curvature parameter can provide a beneficial global geometric bias. The findings in table 2 emphasize the stability and robustness of the learned geometry when trained with different optimizers and precision settings. For example, although mixed-precision training consistently leads to competitive test accuracies across the various datasets, the resulting curvature values often show reduced variance, indicating a more tightly controlled adaptation of the hyperbolic space to the underlying data structure.

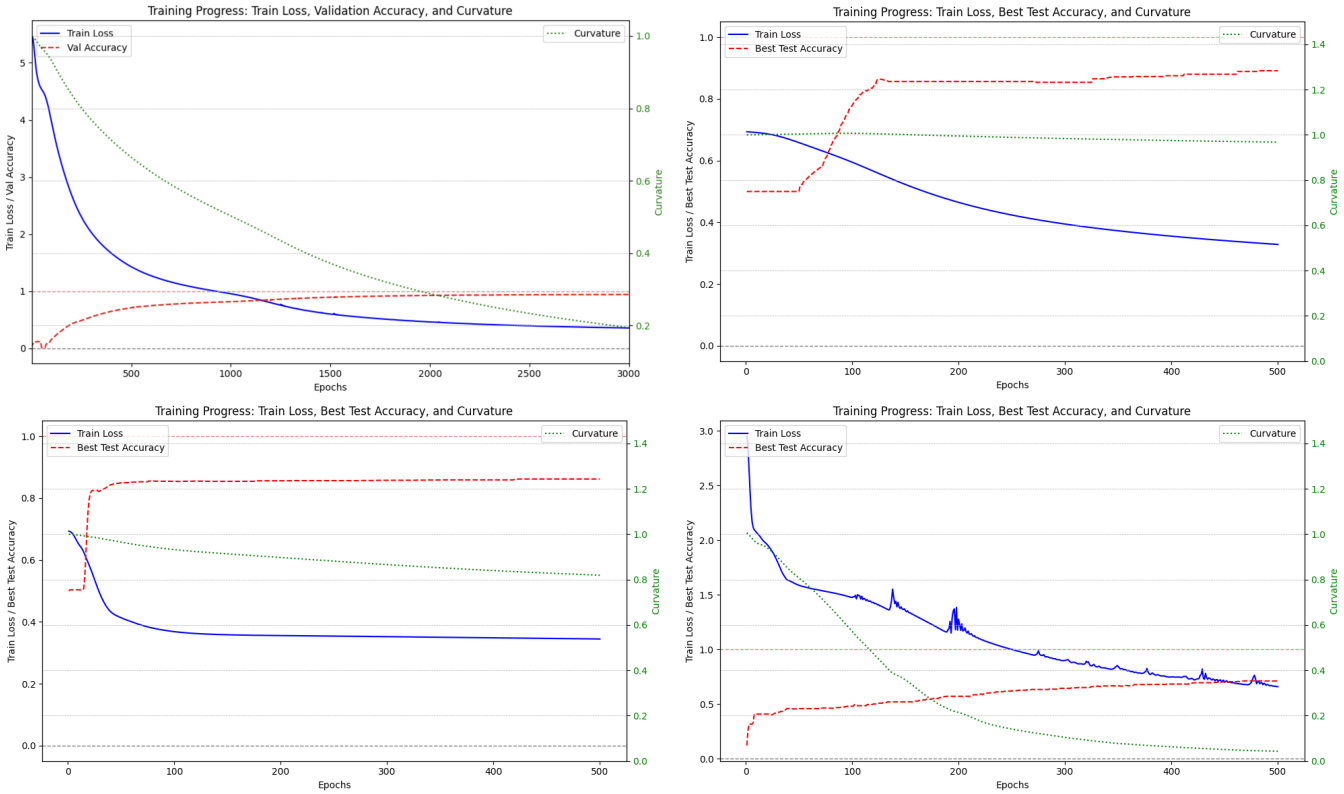


Figure 4: **Top Left: FB15K**, starting c = 1.0, lr = 1e-2, Adam optimizer. **Top Right: Disease**, starting c = 1.0, lr = 1e-4, Adam optimizer. **Bottom Left: Pubmed**, starting c = 1.0, lr=1e-3, Adam optimizer. **Bottom Right: Wordnet**, starting c = 1.0, lr = 1e-2, Adam optimizer. Training progress for all four datasets. Train loss in blue, best test accuracy in red . Curvature in green.

Of the optimizers, Riemannian Adam and Riemannian mixed-precision performed the best (with significantly higher test accuracies), with Riemannian SGD lagging behind them both. This is likely because Riemannian Adam and mixed-precision methods employ adaptive learning rates and variance-based updates, which better accommodate the curvature-related complexities of hyperbolic spaces. In contrast, Riemannian SGD's uniform updates may not sufficiently adapt to rapidly changing gradients, leading to slower or less stable convergence.

We also observe that the Mixed Precision Riemannian optimizer best followed our hypothesis of having higher learned curvatures for more hierarchical data. This could be because it offers a better balance between numerical stability and efficiency. By enabling more precise updates for sensitive parameters like curvature while maintaining computational efficiency, it might lead to better optimization dynamics and improved model performance in hyperbolic spaces.

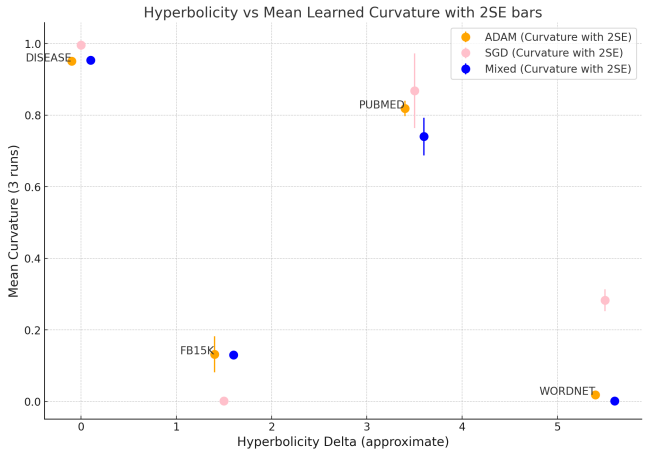


Figure 5: δ-hyperbolicity vs learned curvature for the graphs for each optimizer. A lower δ is indicative of a more hyperbolic graph.

Furthermore, the scatterplot showcases a clear relationship between δ-hyperbolicity and curvature. Diseases ($\delta = 0$), the most hierarchical of the datasets, had the highest curvature, and Wordnet ($\delta = 5.5$) the most euclidean of the datasets, had the lowest curvature. It is interesting that Pubmed had such high curvature and FB15K had such low curvature despite Pubmed being less intrinsically hierarchical than FB15K. Despite FB15K's broader and more explicitly hierarchical relational structure, its

higher relational diversity and complex link patterns may cause the model to settle on a lower overall curvature for stable embeddings. In contrast, Pubmed's simpler and more homogeneously connected network, although less hierarchical, can be effectively compressed into a more strongly curved space, reflecting its underlying distribution of node features and link densities rather than explicit hierarchical layering.

Comparison with Baselines: For a baseline, we compared our HGCN findings for the Diseases dataset (the most hierarchical of the datasets) with Riemannian Adam Optimizer to accuracy metrics for a standard Euclidean GCN with regular Adam Optimizer with a learning rate of 10^{-4}. We find that the best test accuracy over three runs for the standard GCN are 84.625%, 86.50%, and 85.75%. The mean best accuracy was 85.62% for the regular GCN with a standard deviation of 0.95%. Compared to our HGCN, which rendered a mean accuracy of 91.13% and standard deviation of 1.23%, the performance of the baseline GCN is clearly inferior. Therefore, even without layer-by-layer curvature optimization, the use of a global curvature parameter has a clear impact on the structural representations and node embeddings.

5 Conclusion

This report examines the relationship between optimal curvature and δ-hyperbolicity in HGCNs. We present three major findings:

(1) A HGCN with curvature optimized globally is superior to a GCN for highly hierarchal data

(2) In general, as our sample datasets became more hierarchical, or had smaller δ-hyperbolicity, the optimal curvature decreased. Conversely, as datasets became more "Euclidean-like," they tended to be more suited to a curvature closer to 0. Since this was a pilot study with 4 datasets, additional data analysis on a variety of hierarchal knowledge graphs (and a variety of tasks) would be valuable to report statistically significant results.

(3) The Adam Riemannian and Mixed Precision Optimizers are more effective at determining the optimal curvature than the SGD Riemannian Optimizer, likely due to their adaptive learning rates. This may have been because our HGCNs were trained on real world data, which are noisy, but is nevertheless a valuable insight into how unstable and complex hyperbolic gradients can be.

6 Reproducibility

Our GitHub can be found at: https://github.com/ScienceFair2018/HGCNs-Dynamic-Curvature-Training.git

References

[1] Bose, K., & Das, S. (2023, December 11). Hype-GT: Where graph transformers meet hyperbolic positional encodings. arXiv.org. https://arxiv.org/abs/2312.06576

[2] Chami, I., Ying, R., Ré, C., & Leskovec, J. (2019, October 28). Hyperbolic graph convolutional neural networks. arXiv.org. https://arxiv.org/abs/1910.12933

[3] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. 2017 IEEE International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv.2017.74

[4] Yang, M., Feng, A., Xiong, B., Liu, J., King, I., & Ying, R. (2024, October 5). Hyperbolic fine-tuning for large language models. arXiv.org. https://arxiv.org/abs/2410.04010

[5] Zhang, Y., Wang, X., Jiang, X., Shi, C., & Ye, Y. (2019a, December 6). Hyperbolic graph attention network. arXiv.org. https://arxiv.org/abs/1912.03046

[6] Nickel, M., & Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. Advances in Neural Information Processing Systems (NeurIPS).

[7] Nickel, M., & Kiela, D. (2018). Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. International Conference on Machine Learning (ICML).

[8] Ganea, O.-E., Bécigneul, G., & Hofmann, T. (2018). Hyperbolic neural networks. Advances in Neural Information Processing Systems (NeurIPS).

[9] Qiu, Q.,& Li, J. (2022). Hyperbolic Graph Representation via k-Curvature Gromov Hyperbolic Neural Graph Embedding. arXiv:2209.05635.

[10] Bonnabel, S. (2013). Stochastic gradient descent on Riemannian manifolds. IEEE Transactions on Automatic Control.

[11] Becigneul, G., Ganea, O.-E. (2019). Riemannian adaptive optimization methods. International Conference on Learning Representations (ICLR).