Hyperbolic Graph Neural Networks Under the Microscope: The Role of Geometry–Task Alignment

Dionisia Naddeo[*]
University of Rome Tor Vergata

Jonas Linkerhägner[*]
University of Basel

Nicola Toschi
University of Rome Tor Vergata
Harvard Medical School

Geri Skenderi [†]
HASH

Veronica Lachi
UiT The Arctic University of Norway

Abstract

Many complex networks exhibit hierarchical, tree-like structures, making hyperbolic space a natural candidate wherein to learn representations of them. Based on this observation, Hyperbolic Graph Neural Networks (HGNNs) have been widely adopted as a principled choice for representation learning on tree-like graphs. In this work, we question this paradigm by proposing the additional condition of geometry–task alignment, i.e., whether the metric structure of the target follows that of the input graph. We start by empirically showing that HGNNs can recover low-distortion representations when supervision explicitly rewards metric preservation and proceed by theoretically analyzing a controlled regression setting in which a linear readout benefits from representations whose geometry is aligned with the target metric. Finally, by relying on both predictive performance and embedding distortion, we further show that HGNNs gain an advantage on link prediction, a naturally geometry-aligned task, whereas this advantage largely disappears on standard node classification benchmarks. Overall, our findings shift the focus from only asking *Is the graph hyperbolic?* to also questioning *Is the task aligned with hyperbolic geometry?*, showing that HGNNs consistently outperform Euclidean models under such alignment, while their advantage vanishes otherwise.

1 Introduction

Graph representation learning has gathered increasing attention over the last decade, due to the importance of solving downstream tasks such as node classification or link prediction [1]. Early embedding approaches such as DEEPWALK [2] and NODE2VEC [3] focus on learning low-dimensional node representations that reflect graph proximity in Euclidean space by relying on random-walk co-occurrence statistics. Therefore, the core premise behind the embedding paradigm is *geometric faithfulness*: a good graph embedding should preserve the relevant structure of the input graph.

This naturally raises the question of which geometry is appropriate to maintain this faithfulness. Many real-world networks exhibit strong hierarchical organization and tree-like growth patterns, including biological taxonomies [4], knowledge graphs [5], and social hierarchies [6]. From a metric viewpoint, tree-like graphs are closely connected to negatively curved spaces. Crucially, Sarkar [7] provides a guarantee: for a fixed low dimensionality, trees can be embedded in hyperbolic space with arbitrarily small distortion, while this cannot be done in Euclidean space. Motivated by this theory, hyperbolic representation learning has emerged as a powerful tool to encode latent hierarchies [8–12].

[*]Equal contribution.
[†]Main theory contributor.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In particular, Nickel and Kiela [13] introduced Poincaré embeddings, demonstrating that hyperbolic geometry yields low-distortion representations of hierarchical symbolic data and relational structures.

With the advent of Graph Neural Networks (GNNs) [14–17], graph representation learning shifted to end-to-end task-driven optimization via message passing. Building on the success of hyperbolic embeddings, Hyperbolic GNNs (HGNNs) extend this paradigm to hyperbolic spaces [18–21]. These works often motivate the use of hyperbolic architectures by linking their expected benefits to the hyperbolicity of the input graph, commonly measured through Gromov's δ-hyperbolicity [22] . For instance, Chami et al. [19] explicitly conjecture that "HGCN works better on graphs with small δ-hyperbolicity", a line of reasoning that has been adopted, explicitly or implicitly, in subsequent work [20, 23, 24]. This has fostered the belief that HGNNs should be preferred whenever the input graph is highly hyperbolic.

Despite their popularity in applications like brain network analysis, recommender systems, and knowledge graphs [6, 25, 26], the benefits of HGNNs remain contested. Recent work shows that their empirical advantages are difficult to assess on standard benchmarks, where tasks can often be solved from node features alone [27]. Naddeo et al. [28] further demonstrate that, on graph-level tasks, HGNNs do not consistently outperform Euclidean models, even on strongly hyperbolic graphs. These findings highlight a conceptual gap in the current narrative (see Appendix A).

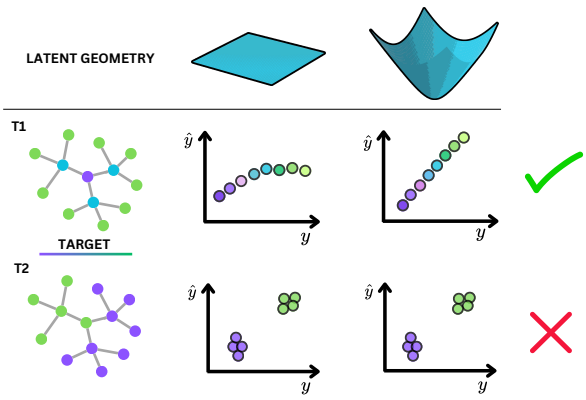


Figure 1: **Geometry–Task Alignment.** Different tasks induce different levels of alignment between the latent geometries and the predicted outcomes. The columns indicate latent spaces with increasing curvature magnitude. The rows plot labels y against predictions $\hat{y}$ for targets **T1**&**T2**. Increasing curvature improves performance for **T1**, whose targets align with the underlying input geometry, but not **T2**, despite identical inputs.

The original motivation for hyperbolic *embeddings* was fundamentally to minimize distortion [19]. In contrast, representations learned via supervision are task-dependent and do not necessarily optimize for metric faithfulness. Consequently, even if hyperbolic space is the right geometry for *the input*, it does not follow that hyperbolic message passing is beneficial for *the task at hand*. Specifically, while trees admit near-isometric hyperbolic embeddings [7], it has never been established that hyperbolic message passing under supervision actually recovers low-distortion node geometries, nor that such low distortion is systematically useful across different graph learning tasks.

In this work, we revisit the question of when *message-passing HGNNs* are *truly* warranted through the lens of *geometric alignment between input structure and task supervision* (Fig. 1). We argue that the "tree-likeness" of the input graph is not, by itself, a sufficient criterion, but we must consider an additional aspect, i.e., whether solving the downstream task requires to reflect (parts of) the hyperbolic input geometry. Indeed, even a maximally hyperbolic input, such as a tree, does not guarantee an advantage for HGNNs: the same graph can favor different representation geometries depending on the supervision signal. Our analysis therefore distinguishes the ability to learn metric-faithful embeddings from their usefulness for downstream prediction. Through controlled experiments on synthetic graphs, evaluations on real-world datasets, and theoretical analyses, we disentangle the benefits of learning hyperbolic representations in GNNs and when they translate into measurable performance gains. In summary, our main contributions are:

(i) We empirically verify that supervised HGNNs can recover low-distortion representations much better than their Euclidean counterparts when the task explicitly rewards metric preservation. Motivated by established theoretical results, we verify that this gap can be reduced as hidden dimensionality increases;

(ii) We introduce geometry–task alignment as a lens for understanding when hyperbolic inductive biases help. By formalizing this principle in a controlled node-regression setting and conducting a comprehensive re-evaluation of HGNNs under a fair benchmarking protocol, we show that

HGNN performance is governed not only by graph hyperbolicity, but also by the alignment between supervision and input geometry;

(iii) We derive actionable guidance for practitioners: HGNNs are well suited for link prediction on hyperbolic graphs, but they do not typically provide advantages for node classification. For the latter, we recommend careful benchmarking against Euclidean counterparts. Initial results indicate that HGNNs can benefit node regression when the target is geometry-aligned, albeit the lack of real-world hyperbolic benchmarks for this task remains an open research gap.

The code repository is available on GitHub.

2 Background

In this section, we introduce the key concepts from hyperbolic geometry and hyperbolic graph learning. Additional details can be found in Appendices A and B.

Riemannian Manifolds and Hyperbolic Space. A Riemannian manifold $(\mathcal{M}, g)$ is a smooth manifold equipped with a metric $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ at each point $x \in \mathcal{M}$, where $T_x\mathcal{M}$ denotes the tangent space at x. The metric g defines a (local) inner product thereby leading to notions of length and curvature, enabling the extension of Euclidean geometry to non-flat spaces. We consider hyperbolic spaces modeled as Riemannian manifolds with constant negative sectional curvature. This property leads to exponential volume growth (Appendix B.1) and makes it suitable for modeling tree-like data, where the number of nodes grows exponentially with depth.

Models of Hyperbolic Space. Hyperbolic space admits several equivalent representations, the most common being the Poincaré ball and the Lorentz (hyperboloid) model, which are isometric to one another. The Poincaré ball $\mathcal{B}_c^d$ models hyperbolic space as an open ball in $\mathbb{R}^d$:

$$\mathcal{B}_c^d = \left\{ \boldsymbol{x} \in \mathbb{R}^d \ : \ \|\boldsymbol{x}\|^2 < 1/c \right\}, \tag{1}$$

equipped with a conformal metric with factor $\lambda_c(\boldsymbol{x}) = \frac{2}{1-c\|\boldsymbol{x}\|^2}$. Although points lie inside a bounded Euclidean region, hyperbolic distances grow rapidly near the boundary. The Lorentz model $\mathcal{L}_c^d$ represents hyperbolic space as a d-dimensional hyperboloid embedded in $\mathbb{R}^{d+1}$:

$$\mathcal{L}_c^d = \left\{ \boldsymbol{x} \in \mathbb{R}^{d+1} \ : \ \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = -1/c, \ x_0 > 0 \right\}, \tag{2}$$

where $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{d} x_i y_i$, and the metric is induced from the ambient Minkowski space.

Geodesics, exponential map, and logarithmic map. In curved spaces, the notion of straight line is replaced by *geodesics*, i.e., locally shortest curves under the manifold metric. The *exponential map* $\exp_{\boldsymbol{x}} : T_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ maps a tangent vector to the manifold by following the geodesic starting at $\boldsymbol{x}$ with that initial velocity. Conversely, the *logarithmic map* $\log_{\boldsymbol{x}} : \mathcal{M} \to T_{\boldsymbol{x}}\mathcal{M}$ locally inverts the above map and sends a point on the manifold back to a tangent vector. These operations enable neural architectures to apply Euclidean computations by mapping back and forth into the manifold [19, 29].

HGNNs. HGNNs learn node representations in a negatively curved space and require aggregation and update operators that are consistent with hyperbolic geometry. Existing approaches can be grouped into two design families. *Tangent-space HGNNs*, such as HGCN [19], perform message passing in a Euclidean tangent space (typically at the origin) and then map the updated features back to the manifold via logarithmic and exponential maps. In contrast, *fully hyperbolic GNNs*, such as HyboNet [21], operate directly in hyperbolic space by aggregating neighborhoods with geometry-aware operators (Appendix B.2). In our experiments, we use HGCN in the Poincaré ball model with learnable curvature, HyboNet in the Lorentz model, and GCN [15] and GAT [17] as Euclidean baselines. Note that HyboNet is designed to work only on the Lorentz hyperboloid.Transformer-based approaches, such as SGFormer [30] and HypFormer [23], fall outside the scope of this work, which focuses on message-passing GNNs.

3 Can HGNNs learn low-distortion embeddings of trees?

HGNNs typically learn representations through supervision rather than by directly optimizing a metric embedding objective. This raises a basic question: *can a HGNN actually learn low-distortion*

representations under supervision? To answer it, we first formally define distortion, and then relate it to a controlled synthetic task which we can learn to solve, thereby directly answering the question. It will be useful for our subsequent analyses to view a GNN as a composition of functions $f_\theta = g_\psi \circ h_\phi$, where h_ϕ is an *embedding* function and g_ψ is a task-dependent *solver*. Concretely, $h_\phi : \mathcal{G} \to \mathcal{M} \subseteq \mathbb{R}^d$ maps nodes (and their graph context) into a (Riemannian) representation manifold $\mathcal{M}$, while the solver $g_\psi : \mathcal{M} \to \mathcal{Y}$ maps latent representations to an output space $\mathcal{Y}$ determined by the task. For example, $\mathcal{Y} = \mathbb{R}$ in node regression or $\mathcal{Y} = \{1, \dots, C\}$ in node classification. For simplicity, we will drop the weight subscripts when referring to the model functions. The distortion of the embedding map h is meant to capture the metric faithfulness of an embedding by considering it as a map between metric spaces. Let the distance measure on the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the shortest-path distance $d_\mathcal{G}(\cdot, \cdot)$ and $d_\mathcal{M}(\cdot, \cdot)$ be geodesic distance on the manifold $\mathcal{M}$.

Definition 3.1 (Embedding distortion [7])**.** Let the embedding $h : (\mathcal{G}, d_\mathcal{G}) \to (\mathcal{M}, d_\mathcal{M})$ be a map between metric spaces that is injective onto its image. Define the contraction and expansion factors:

$$\delta_c(h) = \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{d_\mathcal{G}(i,j)}{d_\mathcal{M}(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j))} \quad \text{and} \quad \delta_e(h) = \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{d_\mathcal{M}(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j))}{d_\mathcal{G}(i,j)}. \tag{3}$$

The distortion of h is defined as $\delta(h) = \delta_c(h)\delta_e(h)$.

It is immediate to see that $\delta(h) \geq 1$. Moreover, we have that for all $i \neq j$: $\delta_c(h)^{-1} d_\mathcal{G}(i,j) \leq d_\mathcal{M}(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)) \leq \delta_e(h)\, d_\mathcal{G}(i,j)$, so h is *bi-Lipschitz onto its image* with constants given by its distortion. When $\delta(h) = 1$ we can write $\delta_c(h) = \delta_e(h)^{-1}$ and therefore $d_\mathcal{M}(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)) = \tau\, d_\mathcal{G}(i,j)$ for some $\tau > 0$. An *embedding without distortion is thereby a global similarity*, i.e., it globally scales pairwise distances in $\mathcal{G}$ by a factor τ. The stricter case of a (global) isometry is obtained by requiring $\delta_e(h) = \delta_c(h) = 1$. We are now ready to ask our first key question regarding hyperbolic message passing over trees: *even if low-distortion hyperbolic embeddings exist [7], can a HGNN actually learn them under supervision?*

3.1 Pairwise Distance Prediction: training explicitly for metric preservation

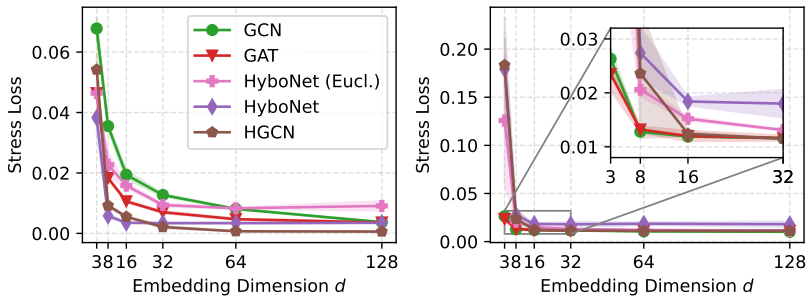
To address this question, we introduce a controlled synthetic task named the Pairwise Distance Prediction (PDP) task. Given a finite, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we consider all node pairs $\mathcal{P} \subseteq \{(i,j) \in \mathcal{V} \times \mathcal{V} : i \neq j\}$, divide them in training, validation and test sets and define regression targets $D_{ij} := d_\mathcal{G}(i,j),\ (i,j) \in \mathcal{P}$. We train the encoder h_ϕ to map nodes to a latent manifold $\boldsymbol{z}_i = h_\phi(\boldsymbol{x}_i) \in \mathcal{M}$, where $\boldsymbol{x}_i$ are the features of node i and $\mathcal{M}$ is either $\mathbb{R}^d$ or $\mathcal{B}_c^d / \mathcal{L}_c^d$. For each pair we compute the corresponding latent distance $\hat{d}_{ij} = d_\mathcal{M}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ and use it to predict D_{ij} through a scalar decoder $\hat{D}_{ij} = g_\psi(\hat{d}_{ij}) = a\, \hat{d}_{ij} + b,\ (i,j) \in \mathcal{P}$, where $\psi = (a, b)$. This isolates the role of the *geometry of the embedding space*: the model succeeds only if shortest-path distances can be expressed up to affine rescaling as geodesic distances in $\mathcal{M}$.

Stress loss. We optimize the parameters ϕ, ψ by minimizing a distortion-aware objective inspired by metric multidimensional scaling [31]:

$$\mathcal{L}_{\text{stress}}(\phi, \psi) = \frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \frac{\left(\hat{D}_{ij} - D_{ij}\right)^2}{D_{ij}^2 + \varepsilon}, \tag{4}$$

where $\varepsilon > 0$ is a small constant for numerical stability. The normalization by D_{ij}^2 prevents optimization from being dominated by large-distance pairs. Importantly, this loss provides a straightforward empirical proxy to operationalize distance preservation by measuring whether graph distances are easily recoverable from latent distances. If this loss is further normalized by the stress obtained from a feature-only baseline that calculates distances directly in the input feature space, we refer to it as *Normalized Stress Loss*.

Datasets. To evaluate the PDP task, we construct two synthetic graphs with comparable size but different intrinsic geometry: a *tree* and a *grid*. These are canonical settings where hyperbolic and Euclidean latent spaces are respectively well matched. The *tree* is a balanced b-ary tree with branching factor $b = 5$ and depth $l = 4$ ($|\mathcal{V}| = 781$), whereas the *grid* is a 28×28 2D lattice ($|\mathcal{V}| = 784$). Node features are sparse (10%) Gaussian vectors in $\mathbb{R}^{100}$.



(a) Tree (b) Grid

Figure 2: Test Stress Loss (↓) vs. embedding dimension for the PDP task. Markers show mean over 10 seeds; shaded regions indicate standard deviation.

Results. Averaged results over 10 random seeds on the *tree* and *grid* graphs are reported in Fig. 2, with detailed values provided in Appendix F.1. For details on tuning and evaluation, see Appendix E.4. Inspired by Nash's embedding theorems [32], we vary the embedding dimension d to assess whether Euclidean models can increasingly approximate a low-distortion embedding of the input graph. To isolate the role of geometry, we further compare HyboNet with a Euclidean counterpart that shares the same message-passing architecture and model capacity, differing only in the embedding space. On the *tree*, hyperbolic models achieve substantially lower stress than Euclidean baselines at low dimensions, confirming that negative curvature provides an effective inductive bias. HyboNet performs best at $d = 3$, while HGCN becomes competitive at higher dimensions. As d increases, Euclidean models progressively close the gap, and by $d = 128$ all methods achieve comparable stress. On the *grid*, the trend reverses: Euclidean models consistently achieve lower stress across all dimensions, indicating that flat geometry better matches grid distances. Notably, HGCN outperforms HyboNet in this setting, suggesting that its learnable curvature allows it to adapt toward an effectively Euclidean representation regime. This interpretation is supported by the learned final-layer curvature magnitudes, which are close to zero on the grid but remain substantially nonzero on the tree (Appendix F.1, Table 7).

We complement the stress-based evaluation with a qualitative analysis that directly compares pairwise distances in data and embedding space. Specifically, we plot average latent distances conditioned on true graph distances, fit a linear model, and report the resulting R^2 score. Full results are provided in Appendix F.1, including a target-permutation control where labels are randomly shuffled across nodes, preserving their marginal distribution while breaking their relation to graph distances.

To test whether the observed advantage depends on geometry–task alignment, we repeat the PDP experiments after randomly permuting target distances across node pairs [33]. This preserves their marginal distribution while breaking their correspondence with the graph metric. On the tree, all models obtain similar test stress, approximately 0.15–0.16, at both $d = 3$ and $d = 128$. On the grid, test stress is also broadly similar across models, ranging from approximately 0.53 to 0.58. Increasing dimension does not consistently improve performance under permuted supervision. In particular, the low-dimensional hyperbolic advantage on the tree disappears despite its unchanged 0-hyperbolicity, supporting the interpretation that the benefit depends on geometry-aligned supervision rather than input hyperbolicity alone. Full results are reported in Appendix F.1, Table 8.

Additional PDP experiments (Appendix F.1.1) at fixed hidden dimension $d = 8$ vary tree depth ($\ell = 2$–6, with $b = 3$) and branching factor ($b = 2$–6, with $\ell = 4$). A hyperbolic model achieves the lowest mean test stress in every configuration, showing that the main findings generalize across balanced trees with different depths and branching factors.

Key Takeaway

When supervision demands metric faithfulness, message passing can exploit the geometric inductive bias of the latent space. Hyperbolic architectures can minimize distortion more effectively on hyperbolic inputs, especially in lower dimensions where model capacity is limited.

4 From Metric Preservation to Task-Aligned Geometry

Section 3 established an important result: *when supervision explicitly demands metric faithfulness*, hyperbolic message passing can recover optimal representations of tree-like graphs in terms of distortion. This dependence on supervision is clear from the control experiment in Appendix F.1: keeping a *tree* as input but randomly shuffling the targets breaks the link between labels and geometry, and hyperbolic models no longer outperform Euclidean ones. This leads us to the key claim of our study: *contrary to popular belief, input hyperbolicity alone is not sufficient to justify HGNNs*. Instead, a hyperbolic inductive bias is beneficial precisely when the task at hand is aligned in some way with the geometric structure of the input. This raises the central question of this section: *when does a task require preserving (parts of) the input geometry?* To provide a first step toward answering this question, we turn to a simple formal model of geometry–task alignment. Consider the problem of univariate *node regression* (NR), where the label space has metric function $d_{\mathcal{Y}}(y_i, y_j) = |y_i - y_j|$. The goal is to formalize the intuition that the labels inherit the input geometry if they share similar metric characteristics to the input graph. More specifically, we propose the following characterizations, in decreasing order of strength:

Suppose that there exist constants $\alpha, \beta > 0$ such that for all $i, j \in \mathcal{V},\ i \neq j$: ***(i) [Global]*** $\alpha\, d_{\mathcal{G}}(i,j) \ \leq d_{\mathcal{Y}}(y_i, y_j) \ \leq \beta\, d_{\mathcal{G}}(i,j)$; ***(ii) [Monotone Global]*** there exists a strictly increasing function $\omega : \mathbb{R} \to \mathbb{R}$ for which $\alpha\, d_{\mathcal{G}}(i,j) \ \leq d_{\mathcal{Y}}(\omega(y_i), \omega(y_j)) \leq \beta\, d_{\mathcal{G}}(i,j)$; ***(iii) [Local]*** nodes in an r-ball $d_{\mathcal{G}}(i,j) \leq r$ respect $\alpha\, d_{\mathcal{G}}(i,j) \ \leq d_{\mathcal{Y}}(y_i, y_j) \leq \beta\, d_{\mathcal{G}}(i,j) \leq \beta\, r$.

Consider now a supervised learning problem with n_t training labels $\mathbf{y} = \{y_i\}_{i=1}^{n_t} \in \mathbb{R}^{n_t}$ and features $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n} \in \mathbb{R}^{n \times d'}$, which live upon a simple, finite, and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}),\ |\mathcal{V}| = n$. The problem consists in learning a model $f_\theta(\mathbf{x_i}, \mathcal{G}) = \hat{y}_i \in \mathbb{R}$ such that the risk $\frac{1}{2}(y_i - \hat{y}_i)^2$ is minimal over training samples. Unless noted otherwise, the representation manifolds are assumed to be Riemannian submanifolds of a Euclidean ambient space s.t. $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ correspond to the dot-product and l^2 norm, respectively. The proofs of all statements are deferred to Appendix C.

Theorem 4.1. *Let $\mathcal{G} \coloneqq (\mathcal{V}, \mathcal{E})$ be the input graph, $f = g \circ h$ be the model, with h an embedding as in Def. 3.1, and $g(\boldsymbol{z}_i = h(\boldsymbol{x}_i) | \boldsymbol{w}, b) : \mathcal{M} \to \mathbb{R} \coloneqq \langle \boldsymbol{w}, \boldsymbol{z}_i \rangle + b = \hat{y}_i$ a linear solver. Define:*

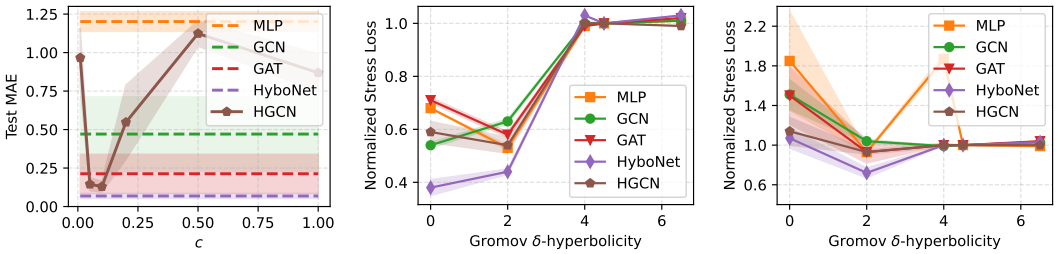
$$\xi(h) = \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|\mathbf{z}_i - \mathbf{z}_j\|}{d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j)}, \quad \zeta(h) = \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|\mathbf{z}_i - \mathbf{z}_j\|}{d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j)}, \quad \rho(h, \boldsymbol{w}) = \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{|\langle w, \mathbf{z}_i - \mathbf{z}_j \rangle|}{\|\boldsymbol{w}\| \|\mathbf{z}_i - \mathbf{z}_j\|}. \quad (5)$$

Then, if $\rho(h, \boldsymbol{w}) > 0$, f is bi-Lipschitz onto its image with upper constant $\beta = \|\boldsymbol{w}\| \zeta(h) \delta_e(h)$ and lower constant $\alpha = \frac{\|\boldsymbol{w}\| \xi(h) \rho(h, \boldsymbol{w})}{\delta_c(h)}$, such that $\alpha d_{\mathcal{G}}(i,j) \leq |\hat{y}_i - \hat{y}_j| \leq \beta d_{\mathcal{G}}(i,j)$.

The above result should not be simply interpreted as saying that that a model class consisting of a linear solver is able to effectively learn geometry aligned labels controlled by distortion. The relevant question is whether the resulting geometry agrees with the ground-truth labels. Indeed, the theorem controls distances between predictions, therefore if the prediction error is uniformly bounded as $|y_i - \hat{y}_i| \leq \varepsilon$, then for all $i \neq j$ we trivially have that $\alpha d_{\mathcal{G}}(i,j) - 2\varepsilon \leq |y_i - y_j| \leq \beta d_{\mathcal{G}}(i,j) + 2\varepsilon$. Thus, *a low-error solution whose embedding has low distortion implies that the labels are approximately aligned with the input geometry*. Conversely, if the labels violate such metric relations, as in the previous permutation experiments, preserving graph distances is not meaningful for the supervised task and the model must either incur prediction error or learn a distorted representation. By analyzing this simple regression setting with a linear readout we can observe a concrete case of alignment, since the label space carries a well-defined distance function. On the other hand, Thm 4.1 does not directly apply to Lorentz-model architectures such as HyboNet, whose ambient space is Minkowski rather than Euclidean. We therefore will simply treat HyboNet as an empirical extension for detailed comparison. Having clarified the presence of geometry–task alignment, we now turn our attention to model behavior using another concept of distortion:

Definition 4.2 (Model distortion)**.** Let $\mathcal{G}$ be the input graph, f be the model, and β, α the upper and lower constants following Thm. 4.1. The model distortion is defined as the condition number $k(f) = \frac{\beta}{\alpha} = \frac{\zeta(h)\delta(h)}{\xi(h)\rho(h, \boldsymbol{w})}$.

Intuitively, model distortion is a task-level analogue of embedding distortion: it measures how the graph metric is transformed by the complete learning model. We can then derive the following results:



(a) HGCN curvature (b) LP stress (c) NC stress

Figure 3: Left: NR MAE (↓) of HGCN with fixed curvature c on the tree for $d = 128$, showing a curvature–performance trade-off. Middle/right: Normalized Stress loss (↓) for predicting shortest-path distances using embeddings learned for LP and NC. Hyperbolic models help for LP at small δ, while gains for NC are marginal.

Proposition 4.3 (Euclidean embeddings)**.** *Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. Furthermore, let h be an embedding of the graph into Euclidean space $\mathbb{R}^d$. The model distortion of f is $k(f) = \frac{\delta(h)}{\rho(h,\boldsymbol{w})}$.*

In the most practically employed case of learning Euclidean embeddings, the model distortion and thus the geometry–task relationship is directly proportional to the distortion of the embedding function. Assuming non-orthogonality between any chord in representation space and the weight vector of g, the model distortion is in fact no less than the embedding distortion, since $0 < \rho(h, \boldsymbol{w}) \leq 1$, therefore the alignment tightness is directly controlled by $\delta(h)$. This fact is precisely the intuition behind most papers that propose the use of HGNNs for tree-like graphs [19]. The following propositions show that despite being intuitive, this motivation does not necessarily hold in hyperbolic space.

Proposition 4.4 (Similarities and isometries)**.** *Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. If $\delta(h) = 1$, the model distortion of f is $k(f) = \frac{\zeta(h)}{\xi(h)\rho(h,\boldsymbol{w})}$.*

Proposition 4.5 (Poincaré ball embeddings)**.** *Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. Let h be an embedding function that embeds into the Poincaré ball $\mathcal{B}_1^d$ and define $\eta := \max_{i \in \mathcal{V}} \|h(\mathbf{x}_i)\|$. Then, the model distortion of f is upper bounded by $k(f) \leq \frac{\delta(h)}{(1-\eta^2)\rho(h,\boldsymbol{w})}$.*

Prop. 4.5 applies specifically to solvers that operate on ambient Poincaré coordinates. It captures a boundary-compression effect where as embeddings approach the boundary, hyperbolic distances grow much faster than the Euclidean coordinates seen by the solver. Practical tangent-space readout usually use $\log_0$ as the coordinate map, therefore the above result should not be interpreted as a direct characterization of the full readout. Nevertheless, it serves in highlighting the general principle that the choice of solver coordinates affects model distortion and alignment, ultimately revealing the notion of geometry-task alignment. Figure 3a empirically illustrates a related curvature–performance trade-off for HGCN. Note that similar phenomena have been observed empirically for hyperbolic generative models [34], where typically the curvature of the latent space is set close to 0.

4.1 Experiments on Node Regression

To approximate a setting where we can verify our theoretical results, we propose a simple task. Extract an anchor node $a \in \mathcal{V}$ and define a label for each node $i \in \mathcal{V} \setminus \{a\}$ as $y_i = \beta^* d_{\mathcal{G}}(i, a)$, for $\beta^* > 0$. Using the reverse triangle inequality and excluding a from the labeled set, we obtain a problem where $0 \leq |y_i - y_j| = \beta^* |d_{\mathcal{G}}(i, a) - d_{\mathcal{G}}(a, j)| \leq \beta^* d_{\mathcal{G}}(i, j)$, which provides a local geometry-alignment probe: it tests whether the model can recover distance-to-anchor structure. Concretely, we use a tree with depth $l = 4$ and branching factors $b = (100, 2, 2, 2)$ with sparse (10%) Gaussian features in $\mathbb{R}^{100}$. The anchor is the root node, therefore the regression error reflects geometric alignment by testing whether the model captures the exponential expansion of the underlying space. As a sanity check, we also report results for a target-permutation baseline under the control column. To assess alignment on a non-hyperbolic structure, we consider the same grid graph as in Section 3.1, using a

Table 1: **NR** MAE (↓) on the synthetic tree and grid dataset for dimension 3 and ≤ 128, and control on the tree.

	Tree			Grid	
Model	3	≤ 128	Control	3	$\leq$128
MLP	1.24±0.06	1.20±0.06	1.20±0.07	8.36±1.67	9.40±2.67
GCN	0.51±0.24	0.47±0.24	1.19±0.07	0.45±0.10	0.46±0.04
GAT	0.63±0.39	0.21±0.13	1.21±0.05	**0.41±0.14**	0.29±0.06
HyboNet	**0.17±0.02**	**0.07±0.02**	1.23±0.06	1.85±0.55	0.45±0.08
HGCN	0.25±0.05	0.08±0.02	**1.16±0.06**	0.44±0.07	**0.27±0.06**

Table 2: Link prediction (LP) and node classification(NC) results on real-world datasets. We also report Gromov δ-hyperbolicity (lower is more hyperbolic). The last row shows the Absolute Gain of the best hyperbolic model over the best Euclidean GNN model.

	Link Prediction (ROC AUC ↑)					Node Classification (Macro F1 ↑)				
Model	Disease $\delta=0$	Airport $\delta=2$	Cora $\delta=4$	Pubmed $\delta=4.5$	Citeseer $\delta=6.5$	Disease* $\delta=0$	Airport $\delta=2$	Cora $\delta=4$	Pubmed $\delta=4.5$	Citeseer $\delta=6.5$
MLP	98.43±0.74	97.66±0.12	90.33±0.94	94.52±0.26	91.45±0.68	48.62±5.06	88.79±1.97	55.63±1.50	70.45±1.12	54.29±1.14
GCN	91.67±1.79	96.58±0.35	92.98±1.01	95.03±0.19	96.02±0.50	89.50±4.82	92.46±0.86	80.09±1.01	78.49±1.47	64.95±1.50
GAT	93.25±1.59	95.67±0.33	93.50±0.68	96.77±0.18	93.32±0.92	85.71±5.50	88.33±5.73	80.71±1.22	78.44±1.88	65.16±1.48
HyboNet	97.99±0.72	98.15±0.17	93.70±0.58	97.08±0.09	91.53±1.40	90.48±5.12	90.87±1.10	78.72±1.01	78.33±1.97	62.98±1.48
HGCN	95.16±2.50	97.66±0.15	93.88±1.09	96.86±0.11	94.46±0.61	90.51±3.45	90.92±2.04	78.34±0.95	78.21±1.47	63.29±2.07
Abs. Gain	**+4.74**	**+1.57**	**+0.38**	**+0.31**	**-1.56**	**+1.01**	**-1.54**	**-1.99**	**-0.16**	**-1.87**

central node as anchor. We evaluate the same models as in Section 3, along with an MLP. Models are tested in two configurations: a constrained setting with dimension 3, and the best configuration among $\{16, 32, 64, 128\}$ (denoted by ≤ 128). Performance is measured using Mean Average Error (MAE) over 10 splits.

Results. The results in Table 1 show that all GNNs outperform the MLP on both datasets, confirming the importance of graph structure. On the tree, the fully hyperbolic HyboNet achieves the best performance, likely due to operating entirely in hyperbolic space. HGCN performs slightly worse, which we attribute to a learned curvature parameter that puts the linear decoder in a tough situation, in similar spirit to Prop. 4.5. This trade-off is illustrated in Figure 3a: at fixed and relatively small curvature ($c = 0.1$), HGCN outperforms all methods except HyboNet. In accordance with the broader message of the theory, we see that both representation geometry and solver coordinates affect alignment. On the other hand, Euclidean models work better on the grid. HGCN is able to keep up due to its learnable curvature parameter, while HyboNet performs worst among all GNN methods at small capacity. Taking into account the control experiment, where all methods perform the same, these results confirm that it is not sufficient for general tasks to consider δ-hyperbolicity, but also the preservation of geometry that the tasks requires. In Appendix F.3, we explore a preliminary diagnostic of target smoothness relative to graph distance. It distinguishes observed from randomized labels more clearly on Tree and Grid than on Airport (Table 21). Finally, our results suggest that node regression is a natural setting where hyperbolic advantages *should* emerge, yet it remains largely unexplored due to the lack of suitable benchmarks, which we highlight as an important direction for future work.

Key Takeaway

When NR is geometry-aligned on tree-like graphs, fully hyperbolic methods outperform others. Methods that rely on tangent space computations might degrade in performance if the representation manifold becomes highly curved, so this parameter must be controlled.

5 Real-World Tasks: Node Classification and Link Prediction

We assess whether HGNNs outperform Euclidean models through a unified evaluation on the standard HGNN benchmarks, covering node classification (NC) and link prediction (LP) on Disease, Airport, Cora, Pubmed, and Citeseer. We report mean ± std. over 10 splits and use a controlled protocol with matched hyperparameter-search budgets, validation-based model selection, and comparable decoders across geometries (Appendix E.4). This comprehensive re-evaluation avoids copied liter-

ature numbers, fixed-capacity assumptions, single-split tuning artifacts, and undertuned Euclidean baselines, enabling a direct assessment of representation geometry. We further recompute Gromov δ-hyperbolicity exactly with SageMath [35], rather than by approximation (Appendix D).

5.1 Node Classification

We evaluate two complementary aspects on standard datasets: predictive performance (Macro F1, Accuracy) and the geometry of learned representations. After end-to-end classification training, we quantify how faithfully node embeddings preserve the input graph metric using stress loss (Section 3). Experimental details are in Appendix E. Compared to prior work, we also fix data-splitting bugs in Disease (marked by *) and data leakage in Airport.

Results. The results are reported in the right half of Table 2. Across all benchmarks, including the most tree-like ones (Disease and Airport), HGNNs do not provide significant improvements over Euclidean counterparts, as also reflected by the Absolute Gain values.This behavior can be understood through the lens of geometry–task alignment. In NC, the label space is discrete and permutation-invariant, and therefore does not induce a continuous notion of distance between classes. As a result, training objectives such as cross-entropy primarily encourage class separability, often leading to representation collapse [36], rather than preserving the input graph metric. This is reflected in Fig. 3c, where none of the methods recover shortest-path distances, regardless of the hyperbolicity of the input graph. Importantly, this does not imply that NC is inherently misaligned with geometry. Rather, alignment in NC is more contingent on the structure of the supervision signal: when labels encode a meaningful metric structure (e.g., hierarchies or ordered relationships), geometry-aligned regimes can emerge (an example is provided in F.4 in Table 27). However, such conditions are typically absent in standard benchmarks, making alignment weaker and less systematic than in LP.

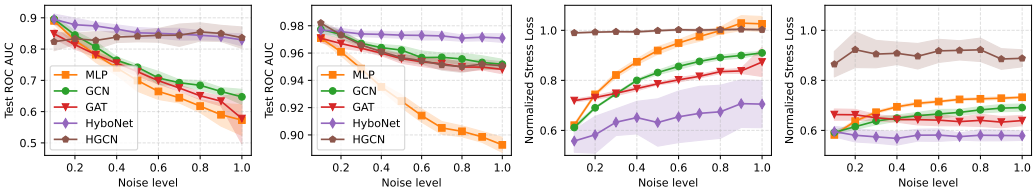
Key Takeaway

Input hyperbolicity alone is not sufficient for geometry–task alignment in NC. When the supervision signal lacks structured relationships between labels, training objectives prioritize class separation over distance preservation, limiting the advantages of hyperbolic representations.

5.2 Link Prediction

For LP we consider the usual predictive performance metrics (ROC AUC, AP) and also the stress loss of the final embedding. As in prior work, all models use a distance-based decoder on the final embeddings to focus on the effects of latent geometry. Motivated by [27], which highlights that many commonly used LP benchmarks can be solved using node features alone via a simple MLP, we additionally consider controlled settings to disentangle feature-driven performance from structural advantages. Specifically, we progressively corrupt node features in highly δ-hyperbolic datasets (Disease and Airport) by injecting Gaussian noise, thereby degrading feature information. Complementary ablations on Disease examine how the hyperbolic advantage depends on feature quality with a fixed distance-based decoder, and on decoder choice with clean features. These ablations, together with evaluations on the synthetic $Tree1111_{\gamma}$ dataset of [27], are reported in Appendix F.2. Appendix E contains all the details on the experimental setup. Finally, we provide a simple theoretical motivation for why LP is indeed geometry-aligned when using distance-based decoders in Appendix C.1, in the hope that this can serve as a starting point for future research.

Results. The LP results on real-world datasets are shown in the left half of Table 2. HGNNs outperform Euclidean counterparts on the most tree-like datasets, Disease and Airport, whereas all GNNs obtain similar ROC AUC on the other benchmarks; the absolute gain is thus substantial on low-δ graphs and negligible otherwise. This suggests that LP is a *geometry-aligned* task: edge prediction relies on local proximity and benefits from latent spaces that preserve tree-like graph structure. A particularly noteworthy observation is that concurrent work in hyperbolic benchmarking (Fig. 5 of Aliakbarisani et al. [37]) has also indirectly observed this phenomenon, for which we provide a concrete explanation for the first time. Stress analysis corroborates this interpretation: message-passing models reduce distortion relative to feature-only baselines, with HyboNet consistently achieving the lowest stress on Disease and Airport. Although MLP can match or exceed GNNs on



(a) Disease ROC AUC (b) Airport ROC AUC (c) Disease Stress (d) Airport Stress

Figure 4: **LP** ROC AUC (↑) and normalized Stress Loss (↓) under feature noise on Disease and Airport. HyboNet achieves strong predictive performance while better preserving geometry.

these datasets, likely because standard LP splits fragment tree-like graphs and make strong node features sufficient [38], corrupting features with Gaussian noise reveals the structural advantage. As feature informativeness decreases (Fig. 4), GNNs outperform MLP, while hyperbolic models, especially HyboNet, maintain higher ROC AUC and lower distortion. The stress analysis shows the same trend (Fig. 4c/4d): noise worsens metric preservation for all methods, but HyboNet remains the most stable and lowest-distortion model. An SNR-based separability score further explains the different MLP behavior on Disease and Airport, showing that Airport retains substantially more informative feature structure under noise (Appendix F.2).

Key Takeaway

With a distance-based decoder, LP is geometry-aligned, since it relies on relative node proximity. Consequently, recovering the graph metric is sufficient to solve the task and graph δ-hyperbolicity can act as a practical proxy for identifying when HGNNs are effective.

6 Conclusion

We revisit when hyperbolic inductive biases benefit GNNs and show that input hyperbolicity alone is insufficient: gains arise when the task is aligned with the graph metric. HGNNs are effective for distance-based LP on tree-like graphs, where geometry and task naturally match, but do not consistently improve standard NC benchmarks, where class separability often matters more than geometric faithfulness. We also identify node regression as a promising but underexplored setting, currently limited by the lack of suitable hyperbolic benchmarks. Overall, our results provide practical guidance and position geometry–task alignment as a principled criterion for model selection.

Limitations and Future Work. Our theory currently covers geometry–task alignment for node regression with linear decoders. The extention it to distance-based decoders, especially for LP, is left to future work. To facilitate this, we provide an initial step in Appendix C.1. It also does not yet yield a quantitative heuristic for selecting hyperbolic models on arbitrary datasets, which we aim to develop. Finally, existing benchmarks remain limited: Disease and Airport are partly saturated, and node-regression benchmarks with clear hyperbolic structure are missing. We therefore rely on synthetic datasets to isolate geometric effects under controlled conditions, while broader real-world validation will require the introduction of new benchmarks.

Acknowledgments

This work was supported by the BRAINFEDERATION FISA-2022-00899, by NEXTGENERATIONEU (NGEU) and funded by the Italian Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), project MNESYS (PE0000006) (to NT)- A Multiscale integrated approach to the study of the nervous system in health and disease (DN. 1553 11.10.2022); the CROSSBRAIN project (European Union's European Innovation Council under grant agreement No. 101070908; The BRAINSTORM PROJECT under Grant Agreement number 101099355. The work is partially funded by the Norwegian Research Council through the Centre of Excellence Integreat - The Norwegian Centre for knowledge-driven machine learning (project 332645).

References

[1] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55, 2024.

[2] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[3] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[4] Hirotaka Matsumoto, Takahiro Mimori, and Tsukasa Fukunaga. Novel metric for hyperbolic phylogenetic tree embeddings. *Biology Methods and Protocols*, 6(1):bpab006, 2021.

[5] Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. Hyperbolic knowledge graph embeddings for knowledge base completion. In *European Semantic Web Conference*, pages 199–214. Springer, 2020.

[6] Yonghui Yang, Le Wu, Kun Zhang, Richang Hong, Hailin Zhou, Zhiqiang Zhang, Jun Zhou, and Meng Wang. Hyperbolic graph learning for social recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8488–8501, 2023.

[7] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International symposium on graph drawing*, pages 355–366. Springer, 2011.

[8] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International conference on machine learning*, pages 3779–3788. PMLR, 2018.

[9] Maksim Kitsak, Ivan Voitalov, and Dmitri Krioukov. Link prediction with hyperbolic geometry. *Physical Review Research*, 2(4):043113, 2020.

[10] Xiao Wang, Yiding Zhang, and Chuan Shi. Hyperbolic heterogeneous information network embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5337–5344, 2019.

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

[12] Zhe Pan and Peng Wang. Hyperbolic hierarchy-aware knowledge graph embedding for link prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2941–2948, 2021.

[13] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[14] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[16] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[18] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[19] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[20] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. Lorentzian graph convolutional networks. In *Proceedings of the web conference 2021*, pages 1249–1261, 2021.

[21] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5672–5686, 2022.

[22] Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

[23] Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. Hypformer: Exploring efficient transformer fully in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3770–3781, 2024.

[24] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. Hyperbolic graph neural networks: A review of methods and applications. *arXiv preprint arXiv:2202.13852*, 2022.

[25] Cole Baker, Isabel Suárez-Méndez, Grace Smith, Elisabeth B Marsh, Michael Funke, John C Mosher, Fernando Maestú, Mengjia Xu, and Dimitrios Pantazis. Hyperbolic graph embedding of meg brain networks to study brain alterations in individuals with subjective cognitive decline. *IEEE journal of biomedical and health informatics*, 28(12):7357–7368, 2024.

[26] Yancong Li, Xiaoming Zhang, Ying Cui, and Shuai Ma. Hyperbolic graph neural network for temporal knowledge graph completion. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8474–8486, 2024.

[27] Isay Katsman and Anna Gilbert. Shedding light on problems with hyperbolic graph learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. Reproducibility Certification.

[28] Dionisia Naddeo, Tiago Azevedo, and Nicola Toschi. Do we need curved spaces? a critical look at hyperbolic graph learning in graph classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 649–664. Springer, 2025.

[29] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31, 2018.

[30] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in neural information processing systems*, 36:64753–64773, 2023.

[31] Stefan Canzar, Van Hoan Do, Slobodan Jelić, Sören Laue, Domagoj Matijević, and Tomislav Prusina. Metric multidimensional scaling for large single-cell datasets using neural networks. *Algorithms for molecular biology*, 19(1):21, 2024.

[32] John Nash. The imbedding problem for riemannian manifolds. *Annals of mathematics*, 63(1): 20–63, 1956.

[33] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy8gdB9xx`.

[34] Lingfeng Wen, Xuan Tang, Mingjie Ouyang, Xiangxiang Shen, Jian Yang, Daxin Zhu, Mingsong Chen, and Xian Wei. Hyperbolic graph diffusion model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15823–15831, 2024.

[35] William Stein and David Joyner. Sage: system for algebra and geometry experimentation. *SIGSAM Bull.*, 39(2):61–64, June 2005. ISSN 0163-5824. doi: 10.1145/1101884.1101889.

[36] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117 (40):24652–24663, 2020.

[37] Roya Aliakbarisani, Robert Jankowski, M. Angeles Serrano, and Marian Boguña. Hypbench: Hyperbolic benchmark for graph neural network performance. In *Non-Euclidean Foundation Models: Advancing AI Beyond Euclidean Frameworks*, 2025. URL `https://openreview.net/forum?id=Mem1jlsEfp`.

[38] Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[39] Victor Chepoi, Feodor F Dragan, Bertrand Estellon, Michel Habib, Yann Vaxes, and Yang Xiang. Additive spanners and distance and routing labeling schemes for hyperbolic graphs. *Algorithmica*, 62(3):713–732, 2012.

[40] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 82(3):036106, 2010.

[41] Fragkiskos Papadopoulos, Dmitri Krioukov, Marián Boguná, and Amin Vahdat. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *2010 Proceedings IEEE Infocom*, pages 1–9. IEEE, 2010.

[42] Rishi Sonthalia and Anna Gilbert. Tree! i am no tree! i am a low dimensional hyperbolic embedding. *Advances in Neural Information Processing Systems*, 33:845–856, 2020.

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

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

[45] Isay Katsman, Eric Chen, Sidhanth Holalkere, Anna Asch, Aaron Lou, Ser Nam Lim, and Christopher M De Sa. Riemannian residual neural networks. *Advances in Neural Information Processing Systems*, 36:63502–63514, 2023.

[46] Riccardo Benedetti and Carlo Petronio. *Lectures on hyperbolic geometry*. Springer Science & Business Media, 1992.

[47] Bruno Martelli. An introduction to geometric topology. *arXiv preprint arXiv:1610.02592*, 2016.

[48] John G Ratcliffe. Hyperbolic n-manifolds. In *Foundations of Hyperbolic Manifolds*, pages 506–596. Springer, 2019.

[49] John M Lee. *Introduction to Riemannian manifolds*, volume 2. Springer, 2018.

[50] John M Lee. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.

[51] Mario Bonk and Oded Schramm. Embeddings of gromov hyperbolic spaces. In *Selected Works of Oded Schramm*, pages 243–284. Springer, 2011.

[52] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[53] Derek Lim, Felix Matthew Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Prasad Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=DfGu8WwT0d`.

[54] Nathann Cohen, David Coudert, and Aurélien Lancin. *Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs*. PhD thesis, INRIA, 2012.

[55] David Coudert, André Nusser, and Laurent Viennot. Hyperbolicity computation through dominating sets. In *ALENEX 2022-SIAM Symposium on Algorithm Engineering and Experiments*, pages 78–90, 2022.

[56] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008. ISSN 0738-4602. doi: 10.1609/aimag.v29i3.2157.

[57] Casey S Greene, Arjun Krishnan, Aaron K Wong, Emanuela Ricciotti, Rene A Zelaya, Daniel S Himmelstein, Ran Zhang, Boris M Hartmann, Elena Zaslavsky, Stuart C Sealfon, et al. Understanding multicellular function and disease with human tissue-specific networks. *Nature genetics*, 47(6):569–576, 2015.

[58] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1325–1334, 2020.

[59] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

[60] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 43–52, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336215. doi: 10.1145/2766462.2767755.

[61] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 05 2021. ISSN 2051-1329. doi: 10.1093/comnet/cnab014.

[62] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 807–816, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557108.

[63] Leonardo FR Ribeiro, Pedro HP 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*, pages 385–394, 2017.

[64] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[65] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

[66] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

A Related Works

Trees constitute a fundamental class of data structures. From a metric perspective, it is well established that trees are prototypical examples of negatively curved spaces. In his seminal work on hyperbolic groups, Gromov introduced the notion of δ-hyperbolicity [22], formalizing hyperbolicity as a coarse, quasi-isometry–invariant property that depends only on large-scale geometry rather than local details or the choice of generating set (See Appendix D for computational details). In this sense, trees are 0-hyperbolic and share with hyperbolic spaces key global properties, including exponential volume growth.

Building on this perspective, Chepoi et al. [39] showed that any n-node δ-hyperbolic graph admits a tree approximation with additive error $O(\delta \log n)$, demonstrating that tree metrics provide faithful approximations of distances in hyperbolic graphs. In the special case of trees themselves, Sarkar [7] further strengthened this connection by providing a constructive embedding result: any weighted tree can be embedded into the two-dimensional hyperbolic plane with arbitrarily small distortion, preserving both topology and metric structure.

Real-world networks are rarely trees; however, many are approximately tree-like at large scales. This includes networks with scale-free degree distributions, where the probability $P(k)$ that a node has degree k follows a power law $P(k) \sim k^{-\gamma}$, with exponent γ typically between 2 and 3. Such networks often display strong heterogeneity, high clustering, and efficient navigability.

Krioukov et al. [40] proposed a geometric framework in which these structural features arise naturally from an underlying hyperbolic geometry. In this model, degree heterogeneity is directly controlled by the negative curvature of the space, while clustering emerges as a consequence of its metric properties. Within the same framework, Papadopoulos et al. Papadopoulos et al. [41] showed that scale-free network topologies naturally emerge from hyperbolic spaces and support efficient greedy routing based solely on geometric distances. Together, these results suggest that hyperbolic geometry provides a unifying explanation for the large-scale, tree-like organization observed in many real-world networks.

Rather than embedding a known tree, a complementary line of work aims to recover latent hierarchical structure directly from data by learning representations in hyperbolic space. Motivated by the exponential growth properties of hyperbolic geometry, these approaches leverage curvature as an inductive bias for modeling hierarchical organization. A seminal contribution in this direction was introduced by Nickel and Kiela [13], who proposed Poincaré embeddings for learning continuous representations of symbolic data with latent hierarchies in an unsupervised setting, demonstrating substantial improvements over Euclidean embeddings. Building on this idea, Sonthalia and Gilbert [42] proposed to first infer an approximate tree structure from the data and subsequently embed it into a low-dimensional hyperbolic space. Overall, these works bridge classical geometric embedding theory and modern representation learning, positioning hyperbolic space as a powerful tool for hierarchy induction rather than explicit metric reconstruction.

While hyperbolic space provides a powerful inductive bias for modeling hierarchical structure, learning in negatively curved spaces requires optimization techniques that go beyond standard Euclidean machinery. In particular, gradient-based learning must be generalized to Riemannian manifolds, where updates are performed along geodesics and parameters are constrained to remain on the manifold. Becigneul and Ganea [43] addressed this challenge by extending popular adaptive optimization methods, including Adam and AdaGrad, to the Riemannian setting, providing both practical algorithms and theoretical guarantees. In parallel, the choice of the specific hyperbolic model was shown to have significant implications for numerical stability and computational efficiency. Nickel and Kiela [8] demonstrated that learning embeddings in the Lorentz (hyperboloid) model is substantially more efficient than in the Poincaré ball, improving optimization behavior while preserving the representational advantages of hyperbolic geometry.

Building on advances in Riemannian optimization, hyperbolic geometry has been progressively integrated into standard neural architectures, enabling the transition from static embedding methods to fully trainable models. In this setting, core neural operations such as linear transformations, non-linearities, attention mechanisms, and residual connections are reformulated to operate directly in hyperbolic space while respecting its geometric constraints. Ganea et al. [29] provided a foundational framework for hyperbolic neural networks by generalizing key components of deep learning to negatively curved manifolds. Subsequent work further expanded this paradigm, proposing unified

formulations of neural network layers in the Poincaré ball [44] and extending residual architectures to general Riemannian manifolds in a principled manner [45]. Collectively, these contributions established hyperbolic neural networks as a viable and scalable modeling paradigm for learning hierarchical representations in an end-to-end fashion.

An active line of research has focused on extending GNNs to hyperbolic spaces in order to combine message passing with the representational advantages of negative curvature. Two seminal works published in parallel in 2019 introduced the first hyperbolic GNN architectures [18, 19]. These approaches generalize Euclidean message passing by mapping node representations between hyperbolic space and local Euclidean tangent spaces via logarithmic and exponential maps. In particular, Chami et al. [19] proposed an hyperbolic graph convolutional network with learnable curvature and attention mechanisms defined in hyperbolic space. Subsequent work sought to reduce reliance on tangent-space operations in order to better preserve geometric structure. Zhang et al. [20] introduced a Lorentzian graph convolutional network that performs neighborhood aggregation directly in the hyperbolic manifold by computing centroids under the squared Lorentzian distance. Along similar lines, Chen et al. [21] proposed a fully hyperbolic neural network operating entirely in the Lorentz model, demonstrating strong empirical performance on a range of benchmarks.

Despite the growing body of work on HGNNs, the benefits of hyperbolic message passing in supervised settings remain under active debate. Recently, Katsman and Gilbert [27] provided a systematic reassessment of hyperbolic graph learning, highlighting several methodological issues related to baseline selection, modeling assumptions, and evaluation protocols. Notably, they showed that when Euclidean models with comparable capacity are carefully trained under the same experimental conditions, they often match or outperform existing hyperbolic graph representation learning methods, even on datasets previously characterized as highly hyperbolic according to Gromov hyperbolicity, including tree-structured graphs. Naddeo et al. Naddeo et al. [28] investigate the interplay between node features and representation geometry in graph classification, showing that hyperbolic models do not necessarily outperform Euclidean ones even on tree-structured graphs. However, their analysis does not examine whether such advantages depend on the alignment between graph geometry and task supervision. Our work extends these reassessments by considering the interplay between representation geometry and task supervision, providing missing details on how the learning process might or might not require a geometric inductive bias that preserves the input metric structure.

B Extended Background

B.1 Hyperbolic Geometry

In this section we provide more details on the exponential growth of (a model of) hyperbolic space and also the architectural mechanisms of HGNNs. For a more rigorous and detailed treatment of hyperbolic geometry and hyperbolic manifolds, we refer the reader to [46–48], while [49, 50] are fantastic resources for the theory of Riemannian manifolds. For a general overview of hyperbolic graph learning and applications, please consult the review of Yang et al. [24].

The Hyperboloid Model. To illustrate exponential volume growth, we consider (w.l.o.g) the hyperbolic space $\mathcal{L}_c^d$ with unit negative curvature which we will simply denote by $\mathcal{L}^d$.

Definition B.1 (Lorentzian scalar product)**.** Let $u = (u_0, u_1, \dots, u_d)$ and $v = (v_0, v_1, \dots, v_d)$ be vectors in $\mathbb{R}^{d+1}$. The *Lorentzian scalar product* on $\mathbb{R}^{d+1}$ is defined as

$$\langle u, v \rangle_{1,d} = -u_0 v_0 + \sum_{i=1}^{d} u_i v_i.$$

This bilinear form induces a pseudo-metric of signature $(1, d)$ on $\mathbb{R}^{d+1}$. We can now define an inner product space by equipping $\mathbb{R}^{d+1}$ with the Lorentzian scalar product, called the $(d+1)$-dimensional Minkowski space

$$\mathbb{R}^{1,d} = (\mathbb{R}^{d+1}, \langle \cdot, \cdot \rangle_{1,d}).$$

For any $x \in \mathbb{R}^{1,d}$ we define its *orthogonal complement* as

$$x^{\perp} = \{y \in \mathbb{R}^{1,d} : \langle x, y \rangle_{1,d} = 0\}.$$

We can now define the hyperboloid as the set of all points of norm -1 in the $(d+1)$-dimensional Minkowski space. This set is a two-sheeted hyperboloid, and by restricting to the upper sheet to get the Lorentz model

$$\mathcal{L}^d = \big\{x = (x_0, \ldots, x_d) \in \mathbb{R}^{1,d} : -x_0^2 + x_1^2 + \cdots + x_d^2 = -1,\ x_0 > 0\big\}.$$

It is possible to show that $\mathcal{L}^d$ is a differentiable manifold and, for every $x \in \mathcal{L}^d$, the tangent space $T_x\mathcal{L}^d$ coincides with $x^{\perp}$. For intuition about this fact, consider that the hyperboloid is an anti-sphere (a sphere of imaginary radius).

To define the metric on this manifold, we can use the notion of induced metric, also known as the first fundamental form. This simply consists in defining the Riemannian metric as the restriction of the ambient inner product to the tangent space:

$$g_{\boldsymbol{x}} : T_{\boldsymbol{x}}\mathcal{L}^d \times T_{\boldsymbol{x}}\mathcal{L}^d \to \mathbb{R} \text{ s.t. } g_{\boldsymbol{x}}(u, v) = \langle u, v \rangle_{1,d},\ u, v \in T_{\boldsymbol{x}}\mathcal{L}^d$$

The induced metric provides the natural Riemannian metric which allows us to compute lengths, angles, and volumes in hyperbolic space. From this point forward, when we talk about $\mathcal{L}^d$, we actually refer to its Riemannian form, i.e., the tuple $(\mathcal{L}^d, g)$, where g is the induced metric described above. This is the *hyperboloid model* of hyperbolic space.

Exponential Growth of Volume. We compute the volume of a geodesic ball and show that it grows exponentially with the radius. To this end, we consider the parametrization

$$x(r, w) = (\cosh r, \sinh r\, w),$$

where $r \geq 0$ is the hyperbolic radius and $w \in S^{d-1} \subset \mathbb{R}^d$ provides the angular direction. It is immediate to check that the image of this parametrization lies in $\mathcal{L}^d$. For a clear exposition, let $\theta = \{\theta_1, \ldots, \theta_{d-1}\} \in S^{d-1}$ be the local coordinates on the sphere, which implies we have a coordinate function $w = w(\theta)$. To compute the induced metric, we can just compute the metric components by considering the tangent basis in above coordinates

$$\partial_r x = (\sinh r, \cosh r\, w), \qquad \partial_\theta x = (0, \sinh r\, \partial_\theta w).$$

The coefficients of the first fundamental form in the coordinates (r, w) are given, in Gauss notation, by

$$E = \langle \partial_r x, \partial_r x \rangle_{1,d}, \qquad F = \langle \partial_r x, \partial_\theta x \rangle_{1,d}, \qquad G = \langle \partial_\theta x, \partial_\theta x \rangle_{1,d}.$$

A direct computation yields

$$\begin{aligned} E &= -\sinh^2 r + \cosh^2 r = 1, \\ F &= 0, \\ G &= \sinh^2 r\, \|\partial_\theta w\|^2. \end{aligned}$$

Note that in the angular component of the metric G, we have the induced metric on the unit sphere $\|\partial_\theta w\|^2 = g_{S^{d-1}}$. We can now write the metric in matrix notation as

$$g = \begin{pmatrix} 1 & 0 \\ 0 & \sinh^2(r)\, g_{S^{d-1}} \end{pmatrix},$$

To obtain the Riemannian volume element, we simply scale the typical volume form by the factors induced by the metric, i.e., it's determinant

$$\det g = \sinh^{2d-2}(r)\ \det(g_{S^{d-1}}),$$

and therefore in the coordinates (r, θ) the Riemannian volume element becomes

$$\begin{aligned} \sqrt{\det g} &= \sinh^{d-1}(r)\, \sqrt{\det g_{S^{d-1}}}. \\ dV &= \sqrt{\det g}\, dr\, d\theta_1 \cdots d\theta_{d-1}. \end{aligned}$$

Since

$$\sqrt{\det g} = \sinh^{d-1}(r)\,\sqrt{\det g_{S^{d-1}}},$$

and the angular part can be written as

$$\sqrt{\det g_{S^{d-1}}}\,d\theta_1 \cdots d\theta_{d-1} = d\Omega_{d-1},$$

i.e., the standard volume element of the unit sphere, we obtain

$$dV = \sinh^{d-1}(r)\,dr\,d\Omega_{d-1}.$$

Integrating the volume element over a geodesic ball of radius R from the origin we obtain

$$\mathrm{Vol}(B_R) = \int_{B_R} dV = \int_{S^{d-1}} \int_0^R \sinh^{d-1}(r)\,dr\,d\Omega_{d-1}.$$

Since the angular integral yields the area of the unit sphere S^{d-1}, denoted by $\omega_{d-1} = \mathrm{Area}(S^{d-1})$, we arrive at

$$\mathrm{Vol}(B_R) = \omega_{d-1} \int_0^R \sinh^{d-1}(r)\,dr.$$

As $r \to \infty$, by the exponential definition of $\sinh$ we have the asymptotic behavior

$$\sinh r \sim \tfrac{1}{2}e^r,$$

and consequently

$$\sinh^{d-1}(r) \sim 2^{-(d-1)}e^{(d-1)r},$$

which allows us to conclude that

$$\mathrm{Vol}(B_R) \sim \frac{\omega_{d-1}}{(d-1)\,2^{d-1}}e^{(d-1)R}$$

Therefore the volume of geodesic balls in $\mathcal{L}^d$ grows exponentially with rate $(d-1)$ in the radius R.

B.2 Hyperbolic Graph Learning

In this section, we review the GNN message-passing mechanism and provide an overview of standard hyperbolic GNN frameworks.

Graph Neural Networks. GNNs learn node representations through iterative neighborhood aggregation, providing an inductive alternative to transductive graph embeddings. Given initial node features $\boldsymbol{x}_v^{(0)} \in \mathbb{R}^{d'}$, representations are updated layer by layer via message passing:

$$\boldsymbol{m}_v^{(\ell)} = \mathrm{AGG}^{(\ell)}\Big(\{\boldsymbol{x}_u^{(\ell-1)} : u \in \mathcal{N}(v)\}\Big), \tag{6}$$

$$\boldsymbol{x}_v^{(\ell)} = \mathrm{UPD}^{(\ell)}\Big(\boldsymbol{x}_v^{(\ell-1)}, \boldsymbol{m}_v^{(\ell)}\Big), \tag{7}$$

where $\mathcal{N}(v)$ denotes the neighbors of node v, and AGG is a permutation-invariant function (e.g., sum, mean, or max). This formulation encompasses a broad class of Euclidean GNN architectures, including GCNs and GATs, which differ in their specific choices of aggregation and update functions.

HGNNs. Hyperbolic Graph Neural Networks (HGNNs) extend standard message passing to non-Euclidean geometries by learning node representations on hyperbolic manifolds. In this setting, node representations at layer ℓ are denoted by $\boldsymbol{z}_v^{(\ell)} \in \mathcal{H}_c^d$, where $\mathcal{H}_c^d$ is a model of d-dimensional hyperbolic space with curvature $-c$ (e.g.,$\mathcal{B}_c^d$ or $\mathcal{L}_c^d$). The goal of HGNNs is to perform aggregation and transformation operations that are consistent with the underlying hyperbolic geometry. Existing approaches can be broadly grouped into two design families.

Tangent-space HGNNs perform message passing in Euclidean tangent spaces. Node representations are first mapped from the manifold to a tangent space via a logarithmic map, updated using standard GNN operations, and then mapped back to hyperbolic space via an exponential map. A representative

example is HGCN [19], which operates in the tangent space at a reference point, typically the origin $\boldsymbol{o}$:

$$\boldsymbol{x}_v^{(\ell-1)} = \log_{\boldsymbol{o}}\left(\boldsymbol{z}_v^{(\ell-1)}\right) \in T_{\boldsymbol{o}}\mathcal{H}_c^d, \tag{8}$$

$$\boldsymbol{x}_v^{(\ell)} = \mathrm{UPD}^{(\ell)}\left(\boldsymbol{x}_v^{(\ell-1)}, \boldsymbol{m}_v^{(\ell)}\right), \tag{9}$$

$$\boldsymbol{z}_v^{(\ell)} = \exp_{\boldsymbol{o}}\left(\boldsymbol{x}_v^{(\ell)}\right) \in \mathcal{H}_c^d, \tag{10}$$

where the curvature c is in fact a learnable hyperparameter.

Fully HGNNs instead aim to perform aggregation, transformation, and non-linear activation directly in hyperbolic space, avoiding explicit tangent-space projections. We consider HyboNet [21] as a representative of this class. HyboNet operates in the Lorentz model and combines linear transformation and non-linearity within a single hyperbolic layer. Given aggregated features $\boldsymbol{x}_v^{(\ell)}$, obtained via an attention-weighted center of mass in Lorentz space, the update is defined as

$$\boldsymbol{z}_v^{(\ell)} = \begin{bmatrix} \sqrt{\|\phi(\boldsymbol{W}\boldsymbol{x}_v^{(\ell)}, \boldsymbol{v})\|^2 - \frac{1}{c}} \\ \frac{\phi(\boldsymbol{W}\boldsymbol{x}_v^{(\ell)}, \boldsymbol{v})}{\|\phi(\boldsymbol{W}\boldsymbol{x}_v^{(\ell)}, \boldsymbol{v})\|} \end{bmatrix}, \tag{11}$$

where $\boldsymbol{W}$ and $\boldsymbol{v}$ are learnable parameters and $\phi(\cdot)$ denotes a non-linear transformation. This formulation ensures that updated representations remain on the Lorentz hyperboloid without having to use rely on the tangent space formalism.

C Proofs

For the sake of clarity and exposition, we report here both the statements and their respective proofs. We remind the reader that in all the statements below, the representation manifolds are assumed to be Riemannian submanifolds in a Euclidean ambient space, therefore $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ correspond to the dot-product and Euclidean norm, respectively. We would like to note that this setting excludes the Lorentz model, but we can make considerations about the Poincaré model.

Theorem 4.1. Let $\mathcal{G} \coloneqq (\mathcal{V}, \mathcal{E})$ be the input graph, $f = g \circ h$ be the model, with h an embedding as in Def. 3.1, and $g(\boldsymbol{z}_i = h(\boldsymbol{x}_i)|\boldsymbol{w}, b) : \mathcal{M} \to \mathbb{R} \coloneqq \langle \boldsymbol{w}, \boldsymbol{z}_i \rangle + b = \hat{y}_i$ a linear solver. Define:

$$\xi(h) = \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|\mathbf{z}_i - \mathbf{z}_j\|}{d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j)},\ \zeta(h) = \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|\mathbf{z}_i - \mathbf{z}_j\|}{d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j)},\ \rho(h, \boldsymbol{w}) = \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{|\langle w, \mathbf{z}_i - \mathbf{z}_j \rangle|}{\|\boldsymbol{w}\|\|\mathbf{z}_i - \mathbf{z}_j\|}. \tag{12}$$

Then, if $\rho(h, \boldsymbol{w}) > 0$, f is bi-Lipschitz onto its image with upper constant $\beta = \|\boldsymbol{w}\|\zeta(h)\delta_e(h)$ and lower constant $\alpha = \frac{\|\boldsymbol{w}\|\xi(h)\rho(h,\boldsymbol{w})}{\delta_e(h)}$, such that $\alpha d_{\mathcal{G}}(i,j) \leq |\hat{y}_i - \hat{y_j}| \leq \beta d_{\mathcal{G}}(i,j)$.

Proof. For any two nodes i, j where $i \neq j$ we have:

$$\frac{|\hat{y}_i - \hat{y_j}|}{d_{\mathcal{G}}(i,j)} = \frac{|\hat{y}_i - \hat{y_j}|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \frac{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))}{d_{\mathcal{G}}(i,j)}. \tag{13}$$

By definition of the expansion factor of the embedding distortion (Eq. 3), we have that:

$$\frac{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))}{d_{\mathcal{G}}(i,j)} \leq \delta_e(h). \tag{14}$$

Furthermore:

$$\frac{|\hat{y}_i - \hat{y_j}|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} = \frac{|\langle w, h(\mathbf{x}_i) - h(\mathbf{x}_j) \rangle|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \leq \frac{\|\boldsymbol{w}\|\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \leq \|\boldsymbol{w}\|\zeta(h), \tag{15}$$

where the first inequality is due to Cauchy-Schwarz.

Putting together all of the above we get the upper bound:

$$\frac{|\hat{y}_i - \hat{y_j}|}{d_{\mathcal{G}}(i,j)} \leq \|\boldsymbol{w}\|\zeta(h)\delta_e(h). \tag{16}$$

Following a similar line of reasoning we can write:

$$\frac{|\hat{y_i} - \hat{y_j}|}{d_{\mathcal{G}}(i,j)} = \frac{|\hat{y_i} - \hat{y_j}|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \frac{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))}{d_{\mathcal{G}}(i,j)} \geq \frac{|\hat{y_i} - \hat{y_j}|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \frac{1}{\delta_c(h)}. \tag{17}$$

By definition of the Euclidean inner product we have:

$$|\hat{y_i} - \hat{y_j}| = |\langle w, h(\mathbf{x}_i) - h(\mathbf{x}_j)\rangle| = \|\boldsymbol{w}\| \, \|h(\mathbf{x}_i) - h(\mathbf{x}_j)\| \, |\cos(\theta_{ij})| \tag{18}$$

where $\theta_{ij} = \angle(w, h(\mathbf{x}_i) - h(\mathbf{x}_j))$ and as such $\rho(h, \boldsymbol{w}) = \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} |\cos(\theta_{ij})|$. Knowing this we can write:

$$\frac{|\langle w, h(\mathbf{x}_i) - h(\mathbf{x}_j)\rangle|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} = \frac{\|\boldsymbol{w}\| \, \|h(\mathbf{x}_i) - h(\mathbf{x}_j)\| \, |\cos(\theta_{ij})|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \geq \|\boldsymbol{w}\| \xi(h) \rho(h, \boldsymbol{w}), \tag{19}$$

such that we can conclude:

$$\frac{|\hat{y_i} - \hat{y_j}|}{d_{\mathcal{G}}(i,j)} \geq \frac{\|\boldsymbol{w}\| \xi(h) \rho(h, \boldsymbol{w})}{\delta_c(h)} \tag{20}$$

and in succession:

$$\frac{\|\boldsymbol{w}\| \xi(h) \rho(h, \boldsymbol{w})}{\delta_c(h)} d_{\mathcal{G}}(i,j) \leq |\hat{y_i} - \hat{y_j}| \leq \|\boldsymbol{w}\| \zeta(h) \delta_e(h) d_{\mathcal{G}}(i,j), \tag{21}$$

which completes the proof. □

Proposition 4.3. Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. Furthermore, let h be an embedding of the graph into Euclidean space $\mathbb{R}^d$. The model distortion of f is:

$$k(f) = \frac{\delta(h)}{\rho(h, \boldsymbol{w})}. \tag{22}$$

Proof. Given that h maps into the ambient Euclidean space, we immediately get that $\xi(h) = \zeta(h) = 1$ and from the definition of model distortion we obtain

$$k(f) = \frac{\delta(h)}{\rho(h, \boldsymbol{w})}. \tag{23}$$

□

Proposition 4.4. Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. If $\delta(h) = 1$, the model distortion of f is:

$$k(f) = \frac{\zeta(h)}{\xi(h) \rho(h, \boldsymbol{w})}. \tag{24}$$

Proof. This result is a direct consequence of the fact that the embedding has no distortion due to it being a similarity/isometry, which implies that $\delta(h) = \delta_c(h)\delta_e(h) = 1$, and from the definition of model distortion we obtain

$$k(f) = \frac{\zeta(h)}{\xi(h) \rho(h, \boldsymbol{w})}. \tag{25}$$

□

We will now present two Lemmas that hold more general value, which will be used in the proof of Prop. 4.5.

Lemma C.1 (Submanifolds with induced Euclidean metric)**.** *Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. Furthermore, let h be an embedding of the graph into a Riemannian submanifold of with the induced geodesic distance, i.e., $h : (\mathcal{G}, d_{\mathcal{G}}) \to (\mathcal{M}, d_{\mathcal{M}})$ and*

$$d_{\mathcal{M}}(a,b) = \inf_{\gamma} \int_0^1 \|\gamma'(t)\| \, dt,$$

such that $\gamma(0) = a, \gamma(1) = b$. Then the model distortion of f is $k(f) = \frac{\zeta(h)\delta(h)}{\xi(h)\rho(h,\boldsymbol{w})} \leq \frac{\delta(h)}{\xi(h)\rho(h,\boldsymbol{w})}$.

Proof. Note that under the induced metric, the length functional of a curve on the manifold $\gamma : [0,1] \to \mathcal{M}$ with endpoints $\gamma(0) = h(\mathbf{x}_i), \gamma(1) = h(\mathbf{x}_j)$ satisfies:

$$L(\gamma) = \int_0^1 \|\gamma'(t)\| \, dt \geq \left\| \int_0^1 \gamma'(t) \, dt \right\| = \|\gamma(1) - \gamma(0)\| = \|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|, \tag{26}$$

due to the triangle inequality and the fundamental theorem of calculus. The geodesic distance, represented as the minimal possible length of any curve satisfying the length functional, must also respect the above inequality, which implies that:

$$\zeta(h) = \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|h(\mathbf{x_i}) - h(\mathbf{x_j})\|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \leq 1. \tag{27}$$

The model distortion then reads

$$k(f) = \frac{\beta}{\alpha} = \frac{\|\boldsymbol{w}\| \zeta(h) \delta_e(h)}{\frac{\|\boldsymbol{w}\| \xi(h) \rho(h, \boldsymbol{w})}{\delta_c(h)}} = \frac{\zeta(h)\delta(h)}{\xi(h)\rho(h, \boldsymbol{w})} \leq \frac{\delta(h)}{\xi(h)\rho(h, \boldsymbol{w})}. \tag{28}$$

□

Lemma C.2 (Submanifolds with conformal metric)**.** *Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. Furthermore, let h be an embedding of the graph into a Riemannian submanifold equipped with a conformal metric, i.e., $h : (\mathcal{G}, d_{\mathcal{G}}) \to (\mathcal{M}, d_{\mathcal{M}})$ where $\mathcal{M}$ has metric $g : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ such that for two tangent vectors $\boldsymbol{v}, \boldsymbol{w} \in T_x\mathcal{M}$, $\mathbf{g}_x(v, w) = \lambda(p)^2 \langle v, w \rangle$. Assume that the conformal factor λ is uniformly bounded such that $0 < \lambda_{min} \leq \lambda \leq \lambda_{max}$. Define the quantity*

$$\xi^{\mathcal{M}^{\mathbb{E}}}(h) = \min_{i,j \in \mathcal{V} i \neq j} \frac{\|h(\mathbf{x_i}) - h(\mathbf{x_j})\|}{d_{\mathcal{M}^{\mathbb{E}}}(h(\mathbf{x}_i), h(\mathbf{x}_j))}, \tag{29}$$

where $d_{\mathcal{M}^{\mathbb{E}}}(\cdot, \cdot)$ is the geodesic distance under the induced metric. Then, the model distortion of f is upper bounded by

$$k(f) \leq \frac{\lambda_{max}\delta(h)}{\lambda_{min}\xi^{\mathcal{M}^{\mathbb{E}}}(h)\rho(h, \boldsymbol{w})}. \tag{30}$$

Proof. The length functional of a curve $\gamma : [0,1] \to \mathcal{M}$ with endpoints $\gamma(0) = h(\mathbf{x_i}), \gamma(1) = h(\mathbf{x_j})$, on a manifold with metric $\mathbf{g}$ is defined as:

$$L_{\mathbf{g}}(\gamma) = \int_0^1 \sqrt{\mathbf{g}_{\gamma(t)}(\gamma'(t), \gamma'(t))} \, dt. \tag{31}$$

When the metric takes the conformal form described above, it reads as:

$$L_{\mathbf{g}}(\gamma) = \int_0^1 \lambda(\gamma(t)) \|\gamma'(t)\| \, dt, \tag{32}$$

which indeed confirms that distances under a conformal metric are (appropriately) scaled counterparts of the distances under the induced Euclidean metric (remember that conformal in this scenario means that angles are preserved, while distances are scaled). Given that the conformal factor is positive, using the bound $\lambda_{\text{min}} \leq \lambda \leq \lambda_{\text{max}}$ pointwise gives

$$\lambda_{\text{min}} L_{\mathbf{g}^{\mathbb{E}}}(\gamma) \leq L_{\mathbf{g}}(\gamma) \leq \lambda_{\text{max}} L_{\mathbf{g}^{\mathbb{E}}}(\gamma), \tag{33}$$

where $L_{\mathbf{g}^{\mathbb{E}}}$ is the length functional for a submanifold with the induced Euclidean metric, as described in the proof of Prop. C.1. The geodesic distance is the one corresponding to the curve that minimizes the length function, and thus taking the infimum over curves will respect monotonicity such that we can arrive at:

$$\lambda_{\text{min}} d_{\mathcal{M}^{\mathbb{E}}}(h(\mathbf{x_i}), h(\mathbf{x_j})) \leq d_{\mathcal{M}}(h(\mathbf{x_i}), h(\mathbf{x_j})) \leq \lambda_{\text{max}} d_{\mathcal{M}^{\mathbb{E}}}(h(\mathbf{x_i}), h(\mathbf{x_j})), \tag{34}$$

where $d_{\mathcal{M}^{\mathbb{E}}}(\cdot, \cdot)$ denotes the distance under the induced Euclidean metric. We can then write:

$$\zeta(h) = \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|h(\mathbf{x_i}) - h(\mathbf{x_j})\|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \leq \max_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|h(\mathbf{x_i}) - h(\mathbf{x_j})\|}{\lambda_{\text{min}} d_{\mathcal{M}^{\mathbb{E}}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \leq \frac{1}{\lambda_{\text{min}}}, \tag{35}$$

$$\xi(h) = \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|h(\mathbf{x_i}) - h(\mathbf{x_j})\|}{d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \geq \min_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \frac{\|h(\mathbf{x_i}) - h(\mathbf{x_j})\|}{\lambda_{\text{max}} d_{\mathcal{M}^{\mathbb{E}}}(h(\mathbf{x}_i), h(\mathbf{x}_j))} \geq \frac{1}{\lambda_{\text{max}}} \xi^{\mathcal{M}^{\mathbb{E}}}(h). \tag{36}$$

Using all this information, we can upper bound the model distortion according as redefining both the upper and lower constants

$$k(f) = \frac{\beta}{\alpha} = \frac{\|\boldsymbol{w}\|\zeta(h)\delta_e(h)}{\frac{\|\boldsymbol{w}\|\xi(h)\rho(h,\boldsymbol{w})}{\delta_c(h)}} \leq \frac{\lambda_{\max}\delta(h)}{\lambda_{\min}\xi^{\mathcal{M}^{\mathbb{E}}}(h)\rho(h,\boldsymbol{w})}. \tag{37}$$

Unsurprisingly, the result is similar to the case for submanifolds with induced metric in Prop. C.1, but rescaled based on the conformal bounds. □

Proposition 4.5. Let $\mathcal{G}$ be the input graph, f be the model, h the embedding function, and g the solver function, all as defined in Thm. 4.1. Let h be an embedding function that embeds into the Poincaré ball $\mathcal{B}_1^d$ and define $\eta := \max_{i\in\mathcal{V}} \|h(\mathbf{x}_i)\|$. Then, the model distortion of f is upper bounded by:

$$k(f) \leq \frac{\delta(h)}{(1-\eta^2)\rho(h,\boldsymbol{w})}. \tag{38}$$

Proof. We start by noting that, given the domain of definition of $\mathcal{B}_1^d$ we have:

$$\lambda_{\min} = \lim_{\|p\|\to 0} \lambda(p) = \lim_{\|p\|\to 0} \frac{2}{1-\|p\|^2} = 2. \tag{39}$$

Following Lemma C.2 we can see that for any piecewise C^1 curve $\gamma : [0,1] \to \mathcal{B}_1^d$ with $\gamma(0) = \boldsymbol{h}(\mathbf{x}_i)$ and $\gamma(1) = h(\mathbf{x}_j)$ the length functional satisfies:

$$L_{\mathbf{g}}(\gamma) = \int_0^1 \lambda(\gamma(t))\|\gamma'(t)\|\, dt \tag{40}$$

$$\geq 2\int_0^1 \|\gamma'(t)\|\, dt \tag{41}$$

$$\geq 2\,\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|, \tag{42}$$

where the last inequality follows because the length of any curve joining $h(\mathbf{x}_i)$ and $h(\mathbf{x}_j)$ must be at least the Euclidean chord length. Since this holds for every admissible curve γ, taking the infimum over curves gives $d_{\mathcal{B}}(h(\mathbf{x}_i), h(\mathbf{x}_j)) \geq 2\,\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|$.

To concretely bound the maximal conformal stretch factor we first proceed by making a direct comparison of the difference in the length of a chord connecting any two points within the ball and it's length under the manifold metric. Given the *convexity of the domain*, we can write any chord connecting points $h(\mathbf{x}_i)$ and $h(\mathbf{x}_j)$ as:

$$\gamma(t) = th(\mathbf{x_i}) + (1-t)h(\mathbf{x_j}), \tag{43}$$

for $t \in (0,1)$. We can then deduce the following two facts:

$$\|\gamma(t)\| \leq \|th(\mathbf{x_i})\| + \|(1-t)h(\mathbf{x_j})\| \leq \max\{\|h(\mathbf{x_i})\|, \|h(\mathbf{x_j})\|\} \tag{44}$$

$$\|\gamma'(t)\| = \|h(\mathbf{x_i}) - h(\mathbf{x_j})\|, \tag{45}$$

where in the first equation we use the triangle inequality and the second implies a constant speed. Letting $c_{ij} = \max\{\|h(\mathbf{x}_i)\|, \|h(\mathbf{x}_j)\|\}$, the first inequality above implies that $1 - \|\gamma(t)\|^2 \geq 1 - c_{ij}^2$ and therefore we obtain:

$$\frac{2}{1-\|\gamma(t)\|^2} \leq \frac{2}{1-c_{ij}^2} \tag{46}$$

We can now use the fact that $\eta = \max_i \|h(\mathbf{x}_i)\|$ and the formula for the length functional under the conformal metric, as presented in the Lemma Prop. C.2, such that:

$$L_{\mathbf{g}}(\gamma) = \int_0^1 \lambda(\gamma(t))\|\gamma'(t)\|\, dt \leq \int_0^1 \frac{2\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|}{1-c_{ij}^2}\, dt \tag{47}$$

$$= \frac{2\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|}{1-c_{ij}^2} \leq \frac{2\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|}{1-\eta^2}. \tag{48}$$

Since the geodesic distance is the infimum over all curves, we obtain the desired upper bound $d_{\mathcal{B}}(h(\mathbf{x}_i), h(\mathbf{x}_j)) \leq \frac{2}{1-\eta^2} \|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|$. Using the language of Lemma C.2 and Thm. 4.1, the above translates to $\zeta(h) \leq \frac{1}{2}$ and $\xi(h) \geq \frac{1-\eta^2}{2}$, which in turn implies that we can redefine the upper and lower constants as $\beta = \frac{1}{2}\|\boldsymbol{w}\|\delta_e(h)$ and $\alpha = \frac{(1-\eta^2)\|\boldsymbol{w}\|\rho(h,\boldsymbol{w})}{2\delta_c(h)}$. By the definition of model distortion we obtain:

$$k(f) \leq \frac{\delta(h)}{(1-\eta^2)\rho(h,\boldsymbol{w})}. \tag{49}$$

We conclude by noting that the final result is essentially identical for a given curvature magnitude $c > 0$ following the same arguments given that $\lambda_c(p) = \frac{2}{1-c\|p\|^2}$ on $\mathcal{B}_c^d = \{p : \|p\|^2 < 1/c\}$ yielding $k(f) \leq \frac{\delta(h)}{(1-c\eta^2)\rho(h,\boldsymbol{w})}$. □

C.1 An initial analysis of distance-based LP decoders

We consider the following setup a starting point for analysis on distance-based LP decoders. Consider a connected, simple, unweighted input graph $\mathcal{G}$ represented via an adjacency matrix $\mathbf{A}$ and $f = g \circ h$ be the model, with h the embedding function as defined in Def. 3.1, and g a distance-based decoder s.t.:

$$\widehat{\mathbf{A}}_{ij} = g(h(\mathbf{x_i}), h(\mathbf{x_j})) = \mathbf{1}\left[d_{\mathcal{M}}(h(i), h(j)) < r\right], \quad i \neq j,$$

What our problem then dictates for each pair $(i, j) \in \mathcal{V} \times \mathcal{V}, i \neq j$ can be separated into two cases: (i) If $\mathbf{A}_{ij} = 1$ we would like our model to satisfy $d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j)) < r \implies \delta_e(h) < r$. On the other hand for a non-edge $\mathbf{A}_{ij} = 0$ we want $d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j)) \geq r$. In this case, we also have that $d_{\mathcal{G}}(i, j) \geq 2 \implies d_{\mathcal{M}}(h(\mathbf{x}_i), h(\mathbf{x}_j)) \geq \frac{2}{\delta_c(h)} \geq r$. Since the edge case implies $\delta_e(h) < r$ we obtain that $\delta_e(h) < \frac{2}{\delta_c(h)}$, which directly leads us to a sufficient condition for the recovery of the input adjacency matrix: $\delta(h) < 2$. If this condition is satisfied then we are guaranteed the existence of a radius $r^* \in \left(\delta_e(h), \frac{2}{\delta_c(h)}\right]$ s.t. $\widehat{\mathbf{A}}_{ij} = \mathbf{A}_{ij}$.

We now proceed to show how this condition manifests in a practical scenario, which is a probabilistic distance-based decoder

$$g(\mathbf{z_i}, \mathbf{z_j}) = 2\sigma\left(-d_{\mathcal{M}}(h_i, h_j)\right),$$

where sigma is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. We consider the typical classification threshold $p_{ij} > 1/2$. Note that the above decoder correctly encodes the intuition that as $d_{\mathcal{M}} \to \inf$ then $p_{ij} \to 0$ and as $d_{\mathcal{M}} \to 0$ then $p_{ij} \to 1$. We can then write down the decision rule to find our decision radius r:

$$\frac{2}{1 + e^{d_{\mathcal{M}}(\mathbf{z_i}, \mathbf{z_j})}} > \frac{1}{2} \implies d_{\mathcal{M}}(h_i, h_j) < \log 3.$$

Following our previous line of reasoning (using the expansion and contraction factors from Definition 3.1 and the fact that for any edge $d_G(i, j) = 1$), a correct classification will happen when $\delta_e(h) < \log 3$). For a non-edge between distinct nodes we then obtain $d_{\mathcal{M}}(h_i, h_j) \geq \log 3$, and since $d_G(i, j) \geq 2$, it is classified negatively whenever:

$$\frac{2}{\delta_c(h)} \geq \log 3, \qquad \text{or equivalently} \qquad \delta_c(h) \leq \frac{2}{\log 3}.$$

Thus these two conditions are sufficient for exact link recovery over distinct pairs (edges). We can clearly see that the sufficient condition is in fact $\delta(h) < 2$.

Unlike Theorem 4.1, this statement depends only on intrinsic latent distances; there is no ambient-coordinate or readout-direction term. It therefore applies to either Poincare- or Lorentz-model embeddings when the decoder uses the corresponding geodesic distance. Finally, we get that the predictive ability can rely entirely on the embedding distortion, which reveals why we consider LP to be inherently more geometry-aligned in the paper. For masked-edge link prediction, the previous statements are about alignment with the target graph metric. Whether that metric can be inferred from the observed training graph is a separate learnability question.

D Gromov δ-hyperbolicity

We recall the definition of δ-Gromov hyperbolicity based on the four-point condition, originally introduced by Gromov as a purely metric characterization of negative curvature [22, 51]. Trees are prototypical examples of hyperbolic metric spaces. Accordingly, the four-point condition quantifies the extent to which a metric space exhibits tree-like behavior.

Definition D.1 (Gromov hyperbolicity via the four-point condition)**.** A metric space (M, d) is said to be δ-hyperbolic, if for some $\delta \geq 0$, it satisfies the *four-point condition*: for any $x, y, z, t \in M$, define

$$S_1 = d(x, y) + d(z, t), \quad S_2 = d(x, z) + d(y, t), \quad S_3 = d(x, t) + d(y, z).$$

Then the two largest values among $\{S_1, S_2, S_3\}$ differ by at most 2δ. The *Gromov hyperbolicity constant* δ^* of (M, d) is the smallest $\delta^* \geq 0$ such that (M, d) is δ^*-hyperbolic.

This formulation is particularly appealing from a computational perspective, as the hyperbolicity of a finite metric space can be evaluated directly by checking the four-point condition over all quadruples of points.

Real-world datasets. We recomputed Gromov δ-hyperbolicity for all datasets used, using an exact algorithm from the SageMath library [35] (BCCM). This is different from previous papers that only provided results from an appoximate algorithm, that approximates δ by sampling quadruples of nodes. For that reason there are some differences in the values reported. In particular Chami et al. [19] reported $\delta = 1$ for Airport, $\delta = 3.5$ for PubMed and $\delta = 11$ for Cora using the approximate algorithm, while the exact algorithm yields $\delta = 2$ for Airport, $\delta = 4.5$ for PubMed and $\delta = 4$ for Cora.

Because of the importance of Gromov δ-hyperbolicity as a decision criterion in the context of HGNNs, we recomputed it for many well-known real-world datasets, which can be reviewed in Table 3. Again, all values are computed using SageMath [35] (BCCM algorithm) on the largest connected components. In case of multiple components, the smaller components achieved a smaller or equal delta (e.g., on Cora and Airport). Most real-world dataset exhibit small delta values, with the exception of Minesweeper which is a synthetic dataset on a 50×50 grid graph. The results on the heterophilic graphs span a range of $\delta = 2.5$ to $\delta = 49$, showing that heterophily alone does not identify a specific geometric regime and therefore does not directly test our hypothesis. Exact computation for larger graphs, like the OGBN graphs [52], or large heterophilic social networks such as Penn94 and Twitch-Gamers [53], is usually infeasible with an exact algorithm and in particular impossible in SageMath due to the limit of 65536 nodes and memory limitations. In these cases sampling-based methods are the only option and therefore no results are reported here. We would like to note however that computing or even only finding approximations to δ-hyperbolicity on large graphs is a research topic in discrete mathematics on its own [54, 55].

Table 3: Exact δ-hyperbolicity values of many well-known real-world datasets. Values are computed using SageMath [35] (BCCM algorithm) on the largest connected components. In case of multiple components, the smaller components achieved a smaller or equal delta (e.g. on Cora and Airport). On PPI all subgraphs achieved the same $\delta = 2$. Most real-world datasets exhibit small delta values, with the exception of Minesweeper which is a synthetic dataset on a 50×50 grid graph.

Category	**Dataset**	$\boldsymbol{\delta}$
Planetoid Citation Graphs [56]	Cora	4.0
	Citeseer	6.5
	Pubmed	4.5
Hyperbolic GNN Benchmarks [19, 57]	Disease	0.0
	Airport	2.0
	PPI	2.0
Small-scale Social Networks [58]	Facebook Page-Page	4.0
	LastFM Asia	3.5
Heterophilous Node Classification Benchmarks [59]	Roman-Empire	3.5
	Amazon-Ratings	11.0
	Minesweeper	49.0
	Questions	3.5
	Tolokers	2.5
Product Co-purchase Graphs [60]	Amazon Computers	2.5
	Amazon Photo	3.0
Wikipedia Networks [61]	Chameleon	2.5
	Squirrel	2.5
Actor Co-occurrence Network [62]	Actor	3.0
Air Transportation Networks [63]	USA	1.5
	Brazil	1.0
	Europe	1.0
University Web Pages (WebKB) [64]	Cornell	2.0
	Texas	1.5
	Wisconsin	2.0
Co-authorship Networks [65]	Coauthor-CS	4.0
	Coauthor-Physics	4.0

E Experimental Setup

Here, we summarize the evaluation protocol for each task and synthetic or real-world dataset.

E.1 Tasks and Real-world Datasets

Tasks On synthetic datasets, we consider all four tasks: (i) **Pairwise Distance Prediction (PDP)**, where the goal is to regress the shortest-path distance between pairs of nodes; (ii) **Node Regression**, where node-level continuous targets (distance to anchor node) are predicted; (iii) **Node Classification**, where node-level discrete targets (distance to anchor node as a class) are predicted; and (iv) **Link Prediction**, formulated as a binary classification task.

For LP tasks, in all experiments, we randomly split edges into $85/5/10\%$ for training, validation and test set, respectively. Negative edges are sampled uniformly using PyG's negative edge sampling utility.

Real-world Datasets. We use well-established benchmarks from the hyperbolic GNN literature [19, 20, 23], including Disease, Airport, Cora, Pubmed, and Citeseer, which exhibit varying levels of δ-hyperbolicity. They cover different kind of real-world networks:

- **Cora / Pubmed / Citeseer (citation networks)**
  Nodes represent papers with bag-of-words representations of text as features, edges are citations, and the node labels are the research topic of each paper.
- **Airport (flight network)**
  Nodes are airports, edges are airline routes. The features contain geographic information

(longitude, latitude and altitude) and the GDP of the airport's country. The node label is the country-level population (discretized into classes).

- **Disease LP/NC (Disease propagation trees)**
  Nodes are individuals in a tree generated via an SIR process. An SIR process is a Disease-spread model in which individuals transition from **s**usceptible to **i**nfected to **r**ecovered as the Disease propagates through a population. Node features indicate the susceptibility to Disease and node labels indicate infected vs. not infected.

On these datasets, we focus on **Link Prediction (LP)** and **Node Classification (NC)** tasks, following standard dataset splits and evaluation protocols used in prior work. For NC, we use a corrected $70/15/15\%$ split for both Airport and Disease, and standard splits [15] with 20 train examples per class, 500 overall for validation and the rest as test set for Cora, PubMed and Citeseer.

For each dataset, we generate 10 independent data splits on which we perform the experimental runs. Each splits also has a different random seed for initializing all random operations (like weight initialization). All reported results are averaged over these 10 splits, and means and standard deviations are reported.

Disease NC data split bug. There was a bug in the original data splitting function on Disease that resulted in a $71.6/4/24.3\%$ split instead of the reported $30/10/60\%$. Additionally the dataset is very unbalanced (of the overall nodes 20% are label 1 and 80% are label 0), but using the original splitting function this was not reflected on the train set, because the test and validation sets had a fixed $50\%/50\%$ ratio of the two labels. This led to even less training samples of the minority class, changing the training dynamics significantly compared to all other NC experiments. In our experiments we refer to the corrected version by Disease*, where we split the nodes using a standard $70/15/15\%$ split, keeping the ratio of the two classes the same in all splits. For full transparency we also report the results using the original splitting ratio with a slightly adapted $70/5/25\%$ splitting (compared to the original $71.6/4/24.3\%$ split) in Appendix F.4.

Airport data leakage. The use of the same splitting function that was used on Disease on the Airport dataset, resulted in data leakage as it was built for binary classification. Applying it to the 4-class classification of the Airport dataset resulted in leakage of validation and test labels to the train set. We fixed this problem by using PyG's standard random node splitting function and only report the results on the corrected splits.

E.2 Synthetic Datasets

For the **PDP task**, we use two synthetic graphs: a tree graph and a grid graph. For each split, node pairs are randomly partitioned using a $70/15/15\%$ split. In both cases, the nodes are equipped with 100-dimensional random (Gaussian) and sparse (10%) features. We chose these features for all our synthetic tasks, to isolate the effect of geometry, but to be still close to real-world features that are often high-dimensional but sparse.

For **Node Regression** and **Node Classification**, we use a synthetic tree with depth $h = 4$ and a varying branching factor of $b = 100, 2, 2, 2$. This means after root the tree has four levels, with increasing number of nodes: $100, 200, 400, 800$. The motivation for this was to work in a tree setting that is not extremely unbalanced w.r.t. to the numbers of nodes per level. The nodes are equipped with 100-dimensional random (Gaussian) and sparse (10%) features. For regression, the anchor node is set as the root, therefore the label is the distance to the root node (values 1 to 4). For classification these four values are the four classes. In both cases we use splits with 20 train examples per class, 500 overall for validation and the rest as test set as for Cora etc. Since only HGCN is not perfectly reproducible for NR with $d = 3$, we additionally run the experiments on the usual 10 splits, ten times and report the average over these in order to be as close as possible to full reproducibility.

For **Link Prediction**, we use the *Tree1111*$_\gamma$ synthetic dataset introduced in [27] with the standard $85/5/10\%$. In this dataset, the graph is a branching factor 10 tree with height 4 and the features are generated procedurally, level-by-level starting from $x^{(0)} \sim \mathcal{N}(0, I_{1000})$ and then $x^{(n)} = \gamma \cdot \mathrm{parent}(\mathrm{x^{(n)}}) + \nu$, where $\nu \sim \mathcal{N}(0, I_{1000})$. Now, when $\gamma = 0$, node features are i.i.d. samples from the base distribution and contain no graph information; therefore, any nontrivial performance must come from the model's ability to exploit the graph. As γ increases, features progressively "leak"

structural information. In this sense, it provides a controlled setting to evaluate whether and to what extent the graph structure itself is actually necessary to solve the task. Katsman and Gilbert [27] already showed results for the MLP on a range of γ-values between 0 and 1.0 and some advantage of using HyboNet for $\gamma = 0$. We provide results over all values of γ using all our GNN models and the MLP.

Additionally, as described in the main text, we experiment with versions of Disease and Airport, where we corrupt the features by Gaussian noise of varying standard deviation.

E.3 Models

Our experimental comparison includes:

- **HGNNs**, operating either fully in hyperbolic space, such as HyboNet [21], or using tangent-space-based methods, such as HGCN [19];
- **Euclidean GNNs**, operating in standard Euclidean space, including GCN [15], GAT [17], and a Euclidean variant of HyboNet [21] that we denote by HyboNet (Eucl.);
- **MLP** that does not consider the graph structure and thus operates solely on the node features (using Euclidean Neural Network operations).

For hyperbolic models, node features are first mapped from Euclidean space to hyperbolic space using the exponential map at the origin. All subsequent message-passing and prediction operations are performed within the chosen hyperbolic manifold. Euclidean counterparts follow identical architectural choices where applicable, differing only in the underlying geometry.

E.4 Hyperparameter Tuning and Control of Confounding Factors

To minimize confounding effects from architecture, decoder choice, optimization, and capacity, we retune all models, including Euclidean and hyperbolic baselines, under the same hyperparameter-search budget for each model–dataset–task combination, selecting configurations solely by validation performance. For the PDP experiments, where training is slower, we use a two-stage search consisting of 150 random configurations over a broad grid. We then select the top 10 configurations based on validation performance and perform a focused grid search around their hyperparameters to obtain a refined set of configurations. For LP, NC, and NR, we instead sample 1000 random hyperparameter configurations per model–dataset–task combination. These configurations are evaluated sequentially on two tuning splits using the task-specific validation criterion, and the selected configuration is then evaluated on the full set of 10 independent splits. The validation criterion is stress loss for PDP, the average of ROC-AUC and AP for LP, Macro-F1 for NC, and MAE for NR. The search space is shared across models whenever applicable, including learning rate, dropout, weight decay, activation function, normalization, gradient clipping, early stopping, hidden dimension, number of message-passing layers, and, when applicable, the use of attention mechanisms, with only model-specific parameters treated separately.

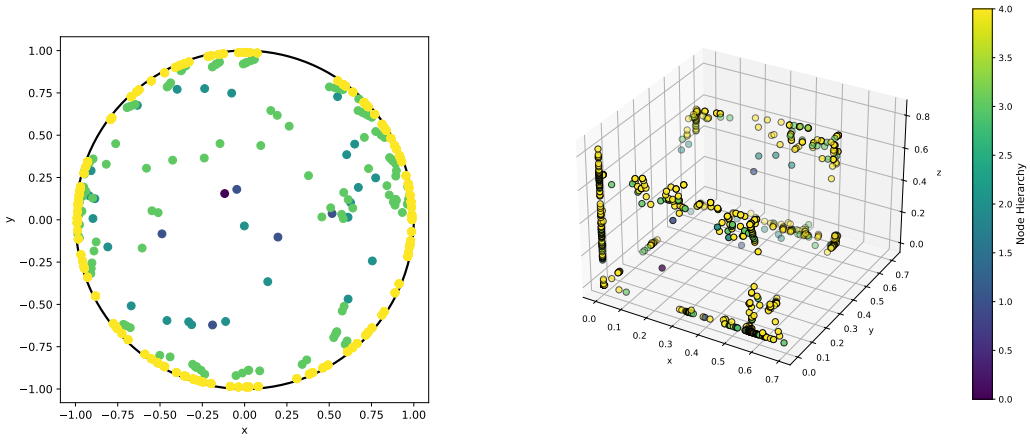
As is standard in the literature, all models are trained for a minimum of 100 epochs and a maximum of 5000 epochs, with early stopping applied if the validation criterion does not improve for 500 consecutive epochs. Hyperbolic models are optimized using the Riemannian Adam optimizer [43], while Euclidean models use the standard Adam optimizer [66].
This careful tuning protocol is crucial because, for GNNs, larger models are not necessarily better: increasing depth or hidden dimensionality can lead to oversmoothing, optimization instability, or unnecessary capacity. Consequently, fixing hidden dimension or depth across all methods may itself bias comparisons, especially in the hyperbolic graph learning literature, where Euclidean baselines are often evaluated in undertuned regimes while hyperbolic models are favored by low-dimensional settings. Our protocol avoids this issue by allowing each model family to select its best configuration under an identical tuning budget and by evaluating the final choice across multiple splits rather than relying on a single favorable run. All reported results correspond to the configuration achieving the best validation performance, and experiments were conducted on NVIDIA A100 and H100 GPUs. Full hyperparameter configurations are provided in the accompanying code repository. Generally, the experiments run in minutes and require only a few GB of VRAM, except for PubMed, which is batched to fit within the VRAM for LP. For PDP, experiments take hours and may also use the full

available VRAM.

To further isolate the effect of geometry, we keep the solver/decoder as simple and comparable as possible across methods. PDP uses a distance-based decoder by design, and LP employs the same functional form across all models, with distances computed in the corresponding latent manifold. Thus, the decoder depends on the learned metric but does not introduce additional model-specific expressivity. For NC and NR, we use GCN and GAT decoders for the corresponding Euclidean models, while the remaining architectures employ either linear decoders or fully hyperbolic decoders, as in the case of HyboNet. Curvature is handled according to the original implementations: HGCN uses learnable curvature, whereas HyboNet uses fixed curvature, which we do not tune except in a dedicated ablation. Finally, where applicable, we include Euclidean counterparts of hyperbolic architectures (e.g., HyboNet (Eucl.)), ensuring that message-passing structure and capacity remain fixed while only the representation geometry varies. Overall, this controlled protocol aims to ensure that performance differences are not driven by tuning artifacts, architectural mismatches, or biased capacity choices, but instead reflect the contribution of the geometric inductive bias.

F Additional Results

F.1 Pairwise Distance Prediction Task



(a) HyboNet, $d = 3$

(b) GCN, $d = 3$

Figure 5: Visualization of the embedding spaces learned by HyboNet and GCN models for $d = 3$ for the PDP task on the *tree*.

We report the results of the Pairwise Distance Prediction (PDP) task in Tables 4 and 5, evaluated across varying embedding dimensions on both the *tree* and *grid* graphs, and corresponding to Fig. 2. Figure 5 shows the learned embeddings of HyboNet and GCN for the *tree* graph with embedding dimension $d = 3$. For HyboNet, the points form 3D Lorentz space are mapped to the Poincaré disc. Here, node embeddings of close to the root nodes are located closer to the center and further away from the root and leaf nodes positioned toward the boundary, reflecting the underlying hierarchical structure of the graph. In contrast, GCN embeddings tend to occupy a roughly cubic region of the Euclidean space, suggesting an increased geometric distortion of pairwise distances.

Table 7 reports the final-layer curvature magnitudes c learned by HGCN on the PDP task, where the manifold curvature is $-c$. On the grid, the learned magnitude decreases from 0.3829 at $d = 3$ to 0.0262 at $d = 8$ and remains below 0.005 for $d \geq 16$. In contrast, the tree yields substantially nonzero curvature magnitudes across all tested dimensions. These results support the interpretation that HGCN adapts toward an approximately Euclidean final-layer geometry on the grid, while retaining negative curvature on the tree.

The PDP results can also be viewed from another, more qualitative perspective by explicitly confronting pairwise distances in data versus embedding space. To do this we plot the averaged pairwise distances (and the standard deviation) in embedding space conditioned on the true pairwise graph distances. Then we perform a linear fit on these points in the plot and report the R^2 score. The R^2 score or the coefficient of determination, measures how much of the variation in the dependent variable (average pairwise distance in embeddings space) is explained by the linear model (which is the true underlying relation here). An R^2 close to 1 means the points lie close to the fitted line, while an R^2 near 0 means the line explains little of the data's variability. So for us, it measures how faithfully the embedding preserves the graph metric on average. Results comparing HyboNet and Euclidean HyboNet are reported in Fig.6. At low embedding dimension, HyboNet achieves the highest R^2 on the *tree* graph (Table 6). As the embedding dimension increases, the performance gap progressively narrows and eventually vanishes.

Table 8 provides the full results of the distance-permutation control discussed in Section 3. After permutation, test stress is broadly similar across models at both embedding dimensions: approximately 0.15–0.16 on the tree and 0.53–0.58 on the grid. The hyperbolic advantage observed with geometry-aligned supervision is therefore absent in this control.

Overall, these results show that when supervision explicitly depends on pairwise distances, message passing can exploit the geometric inductive bias of the latent space to recover low-distortion structure, especially in low-dimensional regimes where representational capacity is limited.

Table 4: Stress loss (↓) (mean ± std) across embedding dimensionality for the pairwise-distance prediction task on the synthetic tree graph.

MODEL	3	8	16	32	64	128
GCN	0.0678 ± 0.0035	0.0355 ± 0.0012	0.0194 ± 0.0015	0.0127 ± 0.0005	0.0081 ± 0.0002	0.0037 ± 0.0002
GAT	0.0464 ± 0.0018	0.0183 ± 0.0004	0.0106 ± 0.0004	0.0070 ± 0.0004	0.0047 ± 0.0003	0.0036 ± 0.0003
HYBONET (EUCL.)	0.0464 ± 0.0013	0.0227 ± 0.0041	0.0158 ± 0.0007	0.0094 ± 0.0006	0.0083 ± 0.0003	0.0091 ± 0.0016
HYBONET	**0.0383 ± 0.0051**	**0.0058 ± 0.0007**	**0.0035 ± 0.0001**	0.0034 ± 0.0002	0.0034 ± 0.0002	0.0035 ± 0.0001
HGCN	0.0542 ± 0.0054	0.0092 ± 0.0006	0.0055 ± 0.0003	**0.0021 ± 0.0004**	**0.0007 ± 0.0001**	**0.0006 ± 0.0001**

Table 5: Stress loss (↓) (mean ± std) across embedding dimensionality for the pairwise-distance prediction task on the synthetic grid graph.

MODEL	3	8	16	32	64	128
GCN	**0.0263** ± 0.0020	**0.0128** ± 0.0002	**0.0119** ± 0.0001	**0.0116** ± 0.0002	**0.0106** ± 0.0002	**0.0102** ± 0.0001
GAT	**0.0235** ± 0.0031	**0.0132** ± 0.0006	0.0120 ± 0.0010	**0.0116** ± 0.0005	0.0114 ± 0.0003	0.0112 ± 0.0002
HYBONET (EUCL.)	0.1258 ± 0.0664	0.0206 ± 0.0013	0.0152 ± 0.0004	0.0131 ± 0.0001	0.0122 ± 0.0002	0.0113 ± 0.0002
HYBONET	0.1792 ± 0.0532	0.0274 ± 0.0048	0.0184 ± 0.0008	0.0180 ± 0.0026	0.0186 ± 0.0017	0.0181 ± 0.0030
HGCN	0.1834 ± 0.0283	0.0236 ± 0.0120	0.0124 ± 0.0004	0.0115 ± 0.0001	0.0112 ± 0.0002	0.0112 ± 0.0002

Table 6: R^2 score (↑) of a linear fit to the embedding distances (averaged per true distance) at embedding dimensions 3 and 128, for tree and grid graphs, averaged over 10 random seeds.

	TREE		GRID	
MODEL	3	128	3	128
GCN	0.3260 ± 0.0491	0.9737 ± 0.0008	0.9324 ± 0.0026	0.9679 ± 0.0004
GAT	0.4625 ± 0.0033	0.9778 ± 0.0013	0.9404 ± 0.0060	0.9617 ± 0.0005
HYBONET (EUCL.)	0.5006 ± 0.0158	0.9341 ± 0.0022	0.5776 ± 0.2634	0.9664 ± 0.0004
HYBONET	0.5979 ± 0.0449	0.9882 ± 0.0007	0.5129 ± 0.1746	0.9365 ± 0.0074
HGCN	0.3595 ± 0.0527	0.9992 ± 0.0001	0.3896 ± 0.1331	0.9681 ± 0.0012

F.1.1 Sensitivity to Tree Depth and Branching Factor

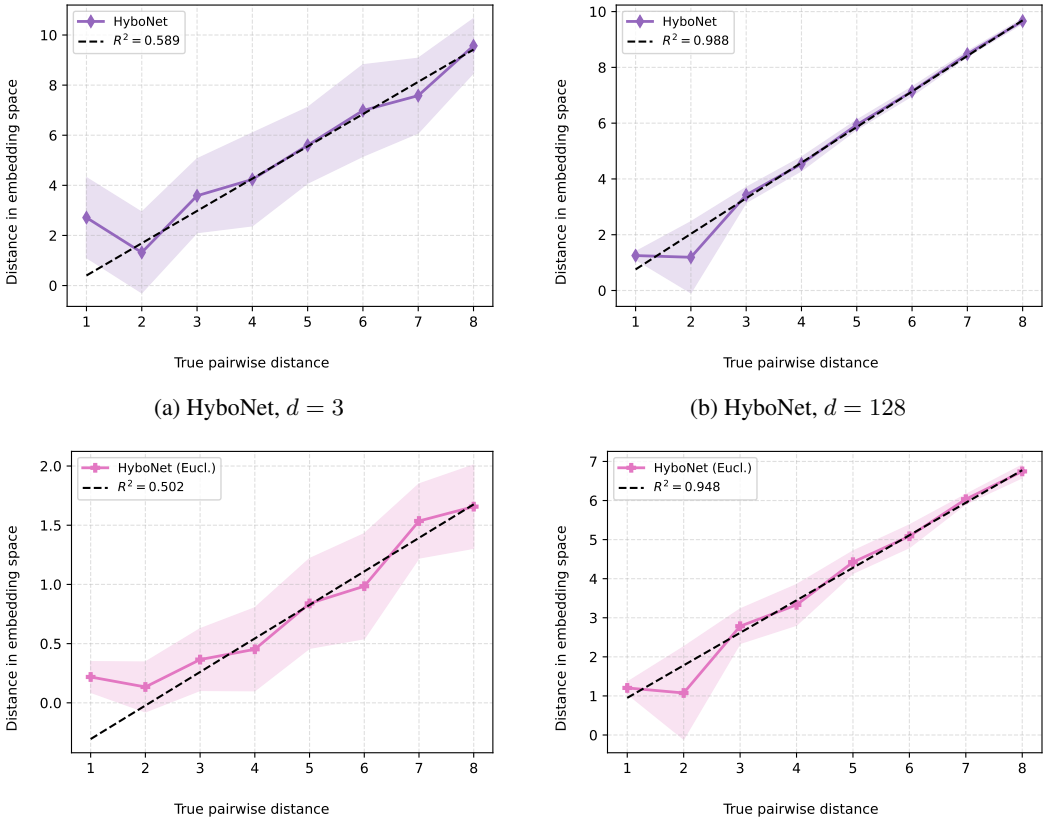
To assess whether the PDP findings extend beyond the original tree configuration, we vary the depth and branching factor of balanced rooted trees. For branching factor $b > 1$ and depth ℓ, measured as

Table 7: Final-layer curvature magnitudes c learned by HGCN for the PDP task; the manifold curvature is $-c$. Entries report mean $\pm$ standard deviation over 10 seeds.

Dataset	$d = 3$	$d = 8$	$d = 16$	$d = 32$	$d = 64$	$d = 128$
Tree	1.9282 ± 0.0972	1.0412 ± 0.0295	2.7957 ± 0.0978	2.0926 ± 0.1792	1.5245 ± 0.0877	1.2167 ± 0.0177
Grid	0.3829 ± 0.2024	0.0262 ± 0.0714	0.0025 ± 0.0023	0.0024 ± 0.0012	0.0009 ± 0.0007	0.0044 ± 0.0009

Table 8: PDP distance-permutation control: test stress loss (↓) after randomly permuting target distances across node pairs on the tree and grid graphs. Entries report mean $\pm$ standard deviation over 10 seeds.

	TREE		GRID	
MODEL	3	128	3	128
GCN	0.1515 ± 0.0048	0.1528 ± 0.0050	0.5349 ± 0.0111	0.5796 ± 0.0079
GAT	0.1515 ± 0.0048	0.1515 ± 0.0048	0.5784 ± 0.0102	0.5687 ± 0.0057
HYBONET (EUCL.)	0.1577 ± 0.0053	0.1561 ± 0.0086	0.5702 ± 0.0066	0.5824 ± 0.0077
HYBONET	0.1520 ± 0.0048	0.1564 ± 0.0054	0.5767 ± 0.0077	0.5713 ± 0.0076
HGCN	0.1522 ± 0.0048	0.1525 ± 0.0053	0.5700 ± 0.0054	0.5772 ± 0.0049



(a) HyboNet, $d = 3$

(b) HyboNet, $d = 128$

(c) HyboNet (Eucl.), $d = 3$

(d) HyboNet (Eucl.), $d = 128$

Figure 6: Line fit of the embedding distances plotted and averaged against the true graph distances for the pairwise–distance prediction task on the *tree*. Results are shown for HyboNet and HyboNet (Eucl.) graphs at low ($d = 3$) and high ($d = 128$) embedding dimensions. The R^2 score measures the goodness of fit of the line, i.e. how faithfully the embedding preserves graph distances.

the number of edges from the root to a leaf, the number of nodes is

$$|\mathcal{V}| = \sum_{k=0}^{\ell} b^k = \frac{b^{\ell+1} - 1}{b - 1}.$$

Table 9: Structural configurations for the additional PDP experiments on balanced rooted trees. Node counts correspond, in order, to the listed values of the varied parameter. Hidden dimension is fixed to $d = 8$.

Sweep	**Fixed parameter**	**Varied parameter**	**Number of nodes**
A: Depth	$b = 3$	$\ell \in \{2, 3, 4, 5, 6\}$	$13, 40, 121, 364, 1093$
B: Branching factor	$\ell = 4$	$b \in \{2, 3, 4, 5, 6\}$	$31, 121, 341, 781, 1555$

Table 10: PDP test stress loss (↓) on balanced trees with fixed branching factor $b = 3$ and varying depth ℓ, using hidden dimension $d = 8$. Entries report mean ± standard deviation over ten seeds. Bold indicates the lowest mean in each column.

	Depth ℓ				
Model	**2**	**3**	**4**	**5**	**6**
GAT	0.2081 ± 0.0922	0.0428 ± 0.0185	0.0249 ± 0.0026	0.0193 ± 0.0013	0.0216 ± 0.0011
GCN	0.2547 ± 0.1032	0.0986 ± 0.0300	0.0551 ± 0.0031	0.0435 ± 0.0041	0.0412 ± 0.0033
HGCN	0.2297 ± 0.1175	0.0636 ± 0.0374	0.0307 ± 0.0163	0.0155 ± 0.0032	$\mathbf{0.0150 \pm 0.0055}$
HyboNet	$\mathbf{0.1589 \pm 0.0750}$	$\mathbf{0.0407 \pm 0.0185}$	$\mathbf{0.0151 \pm 0.0068}$	$\mathbf{0.0068 \pm 0.0013}$	0.0181 ± 0.0071
HyboNet (Eucl.)	0.1885 ± 0.0952	0.0719 ± 0.0221	0.0347 ± 0.0038	0.0294 ± 0.0030	0.0360 ± 0.0050

Increasing depth at fixed branching factor increases hierarchy length and graph diameter, whereas increasing the branching factor at fixed depth increases the rate of expansion between consecutive levels. Every resulting tree remains 0-hyperbolic. These experiments therefore vary structural properties within the same hyperbolicity class, while also changing graph size. Table 9 summarizes the two controlled sweeps and their corresponding node counts. We fix the hidden dimension to $d = 8$, the smallest tested dimension at which a noticeable separation between Euclidean and hyperbolic models consistently appears (Fig. 2). All configurations follow the same feature-generation, training, and ten-seed evaluation protocol as the original PDP experiments.

Tables 10 and 11 show that hyperbolic models achieve the lowest mean test stress across all tested configurations. HyboNet performs best for depths $\ell = 2$–5 and branching factors $b = 2$–5, while HGCN achieves the lowest mean at $\ell = 6$ and $b = 6$, respectively. HyboNet also consistently outperforms its Euclidean counterpart, supporting the benefit of hyperbolic geometry within the matched architecture. These findings extend the original PDP results beyond a single tree configuration, showing that the observed advantage persists across varying hierarchy lengths, expansion rates, and associated graph sizes.

F.2 Link Prediction

Real-world datasets. As mentioned in the main text, in addition to the Receiver Operator Characteristic Area under the Curve (ROC AUC) results, we also report average precision (AP) to give a more comprehensive picture. The full LP results on the real word datasets are in Table 14 for ROC AUC and Table 15 for AP. As for the PDP task, we also include HyboNet (Eucl.) as an additional comparison to isolate the effect of model architecture here. The AP results therefore do not offer a completely new perspective on the model's performance, but merely underline the findings already made for ROC AUC. For full disclosure, we report the exact values of Stress Loss for the distortion analysis for LP. The absolute values are in Table 16 and the normalized values are in Table 17. Along with the ROC AUC results reported in Figure 4, we report AP and absolute stress loss in Figure 7 for the feature-noise experiments on Disease and Airport. To analyze the difference in MLP performance at noise level 1.0 in Figure 4, we compute an SNR-based feature separability score. Given a pairwise scoring function $s(\mathbf{x}_i, \mathbf{x}_j)$ (dot product or mean absolute difference), we define

$$\mathrm{SNR} = \frac{\mu_{\text{pos}} - \mu_{\text{neg}}}{\sigma_{\text{neg}}},$$

where μ_{pos} and μ_{neg} denote the mean scores over positive and negative node pairs (i.e., edges vs non-edges), and σ_{neg} is the standard deviation over negative pairs. As shown in Table 12, Airport retains substantially higher feature separability than Disease at the same noise level, indicating that noise degrades but does not eliminate informative structure. This explains why an MLP, which relies solely on features, still achieves high ROC AUC on Airport even under strong noise.

Table 11: PDP test stress loss (↓) on balanced trees with fixed depth $\ell = 4$ and varying branching factor b, using hidden dimension $d = 8$. Entries report mean ± standard deviation over ten seeds. Bold indicates the lowest mean in each column.

	Branching factor b				
Model	**2**	**3**	**4**	**5**	**6**
GAT	0.0465 ± 0.0145	0.0249 ± 0.0026	0.0211 ± 0.0019	0.0185 ± 0.0006	0.0175 ± 0.0004
GCN	0.0883 ± 0.0214	0.0551 ± 0.0031	0.0453 ± 0.0040	0.0355 ± 0.0023	0.0311 ± 0.0022
HGCN	0.0445 ± 0.0156	0.0307 ± 0.0163	0.0092 ± 0.0006	0.0133 ± 0.0037	$\mathbf{0.0100 \pm 0.0026}$
HyboNet	$\mathbf{0.0393 \pm 0.0216}$	$\mathbf{0.0151 \pm 0.0068}$	$\mathbf{0.0058 \pm 0.0007}$	$\mathbf{0.0071 \pm 0.0014}$	0.0106 ± 0.0017
HyboNet (Eucl.)	0.0564 ± 0.0158	0.0347 ± 0.0038	0.0290 ± 0.0026	0.0235 ± 0.0013	0.0191 ± 0.0013

Table 12: SNR-based feature separability under different noise levels. Higher values indicate better separability.

Dataset	**Method**	**0.0**	**0.5**	**1.0**	**Noise only**
Airport	DOT	0.6128	0.2404	0.1041	0.0170
	DIFF	1.1763	0.3650	0.0979	0.0246
Disease	DOT	0.0826	0.0494	0.0385	0.0250
	DIFF	0.1946	0.0533	0.0106	0.0024

To examine how the observed hyperbolic advantage depends on feature informativeness and decoder choice, Table 13 brings together two complementary ablations on Disease link prediction. Panel A summarizes the feature-noise analysis shown in Figure 4: with a distance-based decoder, increasing Gaussian noise raises the gain of the best hyperbolic model over the best Euclidean GNN from -0.12 to $+18.92$ ROC AUC points. Together with the decreasing feature separability reported in Table 12, this suggests that hyperbolic representations become more useful as informative feature cues weaken. With clean features (Panel B), the hyperbolic advantage observed with distance-based decoding disappears with both concatenation-based decoders. These results support the interpretation that the benefit depends on how predictions use the learned representations: distance-based decoding explicitly relies on the latent metric, whereas concatenation-based decoding can predict links without requiring metric faithfulness.

Synthetic Datasets. We evaluate all our models on the *Tree1111*$_\gamma$ dataset from Katsman and Gilbert [27], which varies node feature informativeness and enables probing model behavior across different feature–structure regimes. Similar to the noisy Disease and Airport experiments in real-world settings, these experiments further stress the role of geometry in a controlled synthetic setting. The results in Figure 8 reproduces the qualitative behavior of the real-world datasets across varying feature informativeness. All GNNs outperform the MLP for small values of γ as they can make use of structural information. All GNNs achieve smaller stress losses than the features and the MLP for small values of γ. GAT and HyboNet continue to do so for larger γ, while the MLP largely follows the trend of the features, which naturally achieve lower stress losses for higher γ values (features are more informative about the graph structure). The hyperbolic methods HGCN and HyboNet generally achieve better performance than GCN, but only HyboNet does so by achieving low stress losses. The notable exception here is GAT, which performs unusually strongly in this setting. We attribute this to the nearly orthogonal 1000-dimensional features used in *Tree1111*$_\gamma$, which act as near-unique node identifiers (similar to one-hot encodings); attention can leverage these signals while avoiding excessive smoothing, unlike degree-normalized aggregation as in GCN. Last, other than on the noisy real-world experiences, HGCN performs better than HyboNet, which we also attribute to the different attention mechanism in HGCN.

Table 13: Complementary ablations on Disease link prediction: (A) feature noise with a fixed distance-based decoder, and (B) decoder choice with clean features. Entries report ROC AUC (↑), mean ± standard deviation over ten splits. Δ denotes the best hyperbolic score minus the best Euclidean GNN score, excluding the graph-free MLP.

A: Feature quality with a distance-based decoder

Model	**Geometry**	$\boldsymbol{\sigma = 0}$	$\boldsymbol{\sigma = 0.5}$	$\boldsymbol{\sigma = 1.0}$
MLP (graph-free)	Euclidean	89.03 ± 1.07	69.94 ± 3.45	57.29 ± 4.89
GCN	Euclidean	89.62 ± 0.99	74.18 ± 0.82	64.72 ± 2.40
GAT	Euclidean	84.90 ± 1.56	72.77 ± 1.36	57.78 ± 8.08
HGCN	Hyperbolic (Poincaré)	82.36 ± 2.90	84.09 ± 1.78	83.64 ± 3.44
HyboNet	Hyperbolic (Lorentz)	89.50 ± 1.70	85.20 ± 1.70	82.95 ± 1.93
$\boldsymbol{\Delta}$		$\mathbf{-0.12}$	$\mathbf{+11.02}$	$\mathbf{+18.92}$

B: Decoder choice with clean features

Model	**Geometry**	**Distance**	**Concat + Linear**	**Concat + MLP**
MLP (graph-free)	Euclidean	98.43 ± 0.74	78.56 ± 10.68	97.91 ± 0.82
GCN	Euclidean	91.67 ± 1.79	93.14 ± 2.24	97.98 ± 0.59
GAT	Euclidean	93.25 ± 1.59	76.09 ± 2.63	96.17 ± 0.88
HyboNet (Eucl.)	Euclidean	79.76 ± 4.84	—	—
HGCN	Hyperbolic (Poincaré)	95.16 ± 2.50	87.64 ± 1.46	93.49 ± 4.05
HyboNet	Hyperbolic (Lorentz)	97.99 ± 0.72	92.01 ± 1.23	95.06 ± 3.10
$\boldsymbol{\Delta}$		$\mathbf{+4.74}$	$\mathbf{-1.13}$	$\mathbf{-2.92}$

Table 14: ROC AUC (↑) results for real-world link prediction (LP). We report Gromov δ-hyperbolicity (lower is more hyperbolic).

Model	**Disease** $\delta = 0$	**Airport** $\delta = 2$	**Cora** $\delta = 4$	**Pubmed** $\delta = 4.5$	**Citeseer** $\delta = 6.5$
MLP	98.43±0.74	97.66±0.12	90.33±0.94	94.52±0.26	91.45±0.68
GCN	91.67±1.79	96.58±0.35	92.98±1.01	95.03±0.19	96.02±0.50
GAT	93.25±1.59	95.67±0.33	93.50±0.68	96.77±0.18	93.32±0.92
HyboNet (Eucl.)	79.76±4.84	95.43±0.29	92.92±0.75	95.72±0.17	92.92±0.76
HyboNet	97.99±0.72	98.15±0.17	93.70±0.58	97.08±0.09	91.53±1.40
HGCN	95.16±2.50	97.66±0.15	93.88±1.09	96.86±0.11	94.46±0.61

F.3 Node Regression

Here we provide additional results on the synthetic tree and grid datasets. Since for NR, we also use a stress loss as an evaluation metric in training, but on the regression labels from the train set (instead of pairwise shortest path graph distances), we report the results on the test set in Table 18. The findings remain the same as for MAE, HyboNet dominates on the tree dataset and the GCN/GAT on the grid. HGCN performs reasonably well on both datasets due to its learnable curvature. The curvature–performance trade-off for HGCN (with fixed curvature) shown in Figure 9 is also very similar as for the MAE. We additionally added the result for $d = 3$. Tables 19 and 20 provide a distortion analysis, measuring how faithfully the learned node representations preserve the input graph metric via the stress loss (Section 3). They show that no method does preserve pairwise graph distances as measured by the Normalized Stress Loss on the tree and control dataset. This is expected since the task does not require the method to preserve the global geometry. In particular with the root node as anchor, models can completely ignore angular effects and collapse all node embeddings in the same level of the tree. On the grid dataset, GCN, GAT and HGCN do reduce distortion to some extent. By a similar analogy, using a central point as the center of the grid implies the grid geometry has to be preserved to some extent, particularly for smaller distances.

Table 15: AP (↑) results for real-world link prediction (LP). We report Gromov δ-hyperbolicity (lower is more hyperbolic).

Model	Disease $\delta = 0$	Airport $\delta = 2$	Cora $\delta = 4$	Pubmed $\delta = 4.5$	Citeseer $\delta = 6.5$
MLP	97.71±1.18	96.77±0.22	89.12±1.35	93.54±0.31	90.44±0.71
GCN	85.47±3.36	94.62±0.63	92.57±1.29	94.57±0.25	96.04±0.61
GAT	90.38±2.05	95.34±0.41	93.81±0.76	96.95±0.18	93.76±0.98
HyboNet (Eucl.)	75.03±6.59	94.52±0.47	92.78±0.88	95.79±0.13	93.80±0.74
HyboNet	96.54±1.62	97.55±0.38	93.78±0.66	97.01±0.09	92.47±1.44
HGCN	92.60±3.90	96.97±0.22	93.80±1.25	96.71±0.14	94.71±0.77

Table 16: Stress loss (↓) when using embeddings from LP to predict pairwise shortest path graph distances with linear regression.

Model	Disease $\delta = 0$	Airport $\delta = 2$	Cora $\delta = 4$	Pubmed $\delta = 4.5$	Citeseer $\delta = 6.5$
Features	0.0842	0.0907	0.1507	0.0754	0.2584
MLP	0.0573±0.0003	0.0476±0.0005	0.1498±0.0000	0.0754±0.0000	0.2611±0.0004
GCN	0.0457±0.0011	0.0567±0.0009	0.1501±0.0001	0.0754±0.0000	0.2618±0.0004
GAT	0.0598±0.0007	0.0518±0.0012	0.1495±0.0001	0.0754±0.0000	0.2625±0.0007
HyboNet (Eucl.)	0.0506±0.0042	0.0661±0.0010	0.1563±0.0001	0.0754±0.0000	0.2618±0.0002
HyboNet	0.0316±0.0022	0.0396±0.0009	0.1561±0.0004	0.0754±0.0000	0.2652±0.0004
HGCN	0.0497±0.0032	0.0479±0.0027	0.1502±0.0021	0.0754±0.0000	0.2569±0.0006

Table 17: Stress loss (↓) when using embeddings from LP to predict pairwise shortest path graph distances with linear regression. Values are normalized such that 1 corresponds to the same loss as with the distances computed directly in feature-space.

Model	Disease $\delta = 0$	Airport $\delta = 2$	Cora $\delta = 4$	Pubmed $\delta = 4.5$	Citeseer $\delta = 6.5$
MLP	0.68±0.00	0.53±0.01	0.99±0.00	1.00±0.00	1.01±0.00
GCN	0.54±0.01	0.63±0.01	1.00±0.00	1.00±0.00	1.01±0.00
GAT	0.71±0.01	0.58±0.01	0.99±0.00	1.00±0.00	1.02±0.00
HyboNet (Eucl.)	0.60±0.05	0.74±0.01	1.04±0.00	1.00±0.00	1.01±0.00
HyboNet	0.38±0.03	0.44±0.01	1.03±0.00	1.00±0.00	1.03±0.00
HGCN	0.59±0.04	0.54±0.03	1.00±0.01	1.00±0.00	0.99±0.00

Table 18: Node Regression test stress loss (↓) results on the synthetic tree and grid dataset for embedding dimension 3 and ≤ 128.

	Tree		Grid		Control Tree
Model	3	≤128	3	≤128	≤128
MLP	0.2144±0.0076	0.2236±0.0086	0.5772±0.1847	0.7364±0.3329	0.2319±0.0215
GCN	0.0375±0.0536	0.0341±0.0542	0.0029±0.0010	0.0027±0.0009	0.2229±0.0196
GAT	0.0708±0.0713	0.0104±0.0061	0.0053±0.0044	0.0022±0.0012	0.2184±0.0163
HyboNet	0.0044±0.0046	0.0008±0.0007	0.0583±0.0263	0.0069±0.0040	0.2198±0.0125
HGCN	0.0243±0.0072	0.0057±0.0020	0.0039±0.0022	0.0019±0.0014	0.2325±0.0217

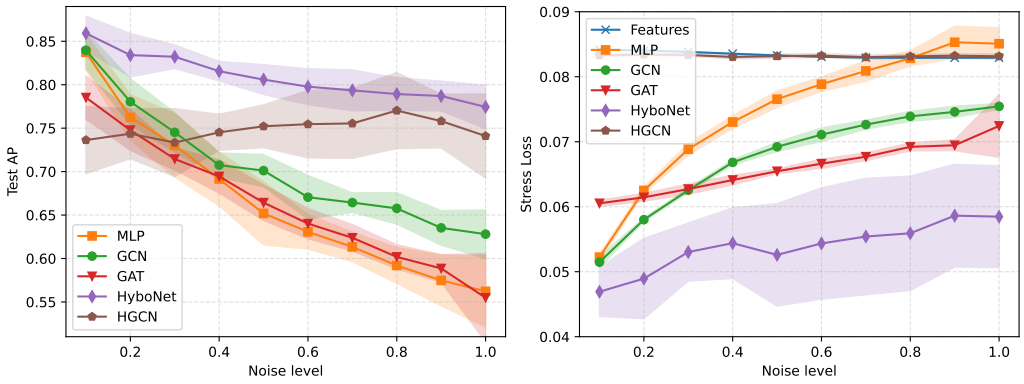
Beyond evaluating learned representations, we also explore whether task-side geometric structure can be detected before training. As a preliminary diagnostic for node regression, we compare pairwise distances in the graph and label spaces, normalizing each by its maximum:

$$\widetilde{d}_{\mathcal{G}}(i,j) = \frac{d_{\mathcal{G}}(i,j)}{\max_{u \neq v} d_{\mathcal{G}}(u,v)}, \qquad \widetilde{d}_{\mathcal{Y}}(i,j) = \frac{|y_i - y_j|}{\max_{u \neq v} |y_u - y_v|}.$$

For connected graphs and nonconstant scalar labels, we define the pairwise slopes and their maximum as

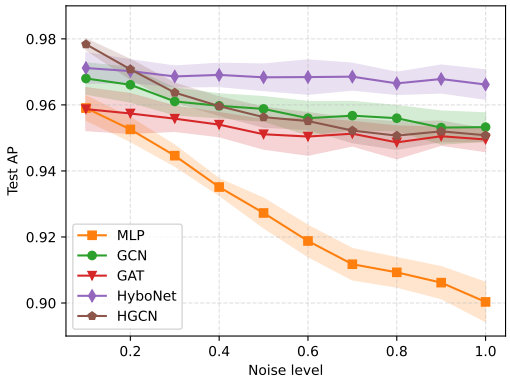
$$s_{ij} = \frac{\widetilde{d}_{\mathcal{Y}}(i,j)}{\widetilde{d}_{\mathcal{G}}(i,j)}, \qquad L_{\max} = \max_{i \neq j} s_{ij}.$$

When computed over all pairs, $L_{\max}$ is the smallest upper Lipschitz constant of the normalized label map on the observed nodes. Smaller values indicate smoother target variation relative to the graph

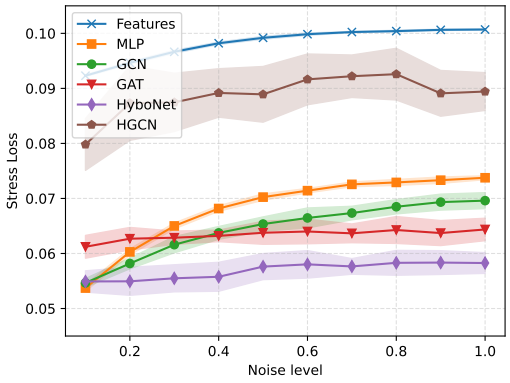


(a) AP (↑) for LP on noisy Disease.

(b) Stress loss (↓) for LP on noisy Disease.



(c) AP (↑) for LP on noisy Airport.



(d) Stress loss (↓) on noisy Airport.

Figure 7: LP results (AP and Stress Loss) on real-world datasets Disease ($\delta = 0$) and Airport ($\delta = 2$) for different levels of noise added to the features. In all cases the GNNs clearly outperform the MLP and the hyperbolic methods on Disease and HyboNet on Airport outperform the Euclidean models.

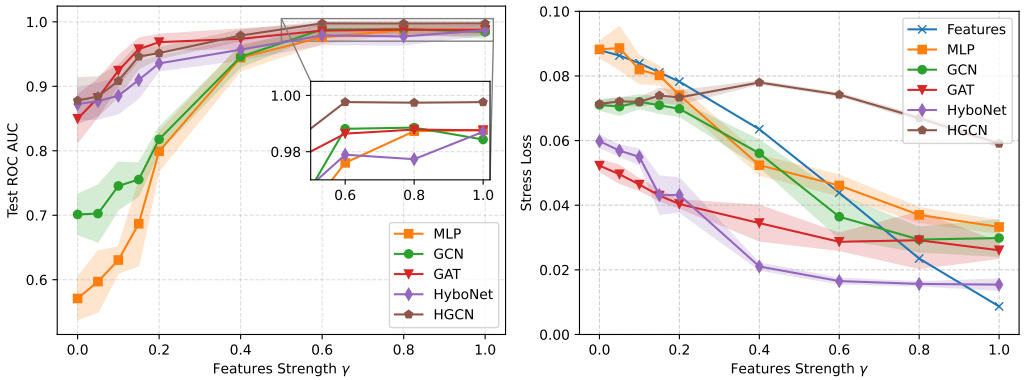
Table 19: Normalized Stress Loss (↓) on the synthetic tree and grid dataset using NR embeddings for dimension 3 and ≤ 128.

	Tree			Grid	
Model	**3**	**≤128**	**Control**	**3**	**≤128**
MLP	1.03±0.07	1.30±0.08	1.03±0.02	1.18±0.06	1.06±0.01
GCN	1.21±0.03	1.23±0.09	1.31±0.08	0.64±0.01	0.97±0.01
GAT	1.29±0.09	1.38±0.10	1.16±0.09	0.63±0.02	0.63±0.02
HyboNet	1.40±0.01	1.49±0.03	1.08±0.13	1.04±0.09	0.93±0.06
HGCN	1.25±0.15	1.23±0.23	0.90±0.03	0.83±0.11	0.86±0.15

metric. We compare the observed labels with controls obtained by permuting labels across nodes or replacing them with independent uniform random labels.

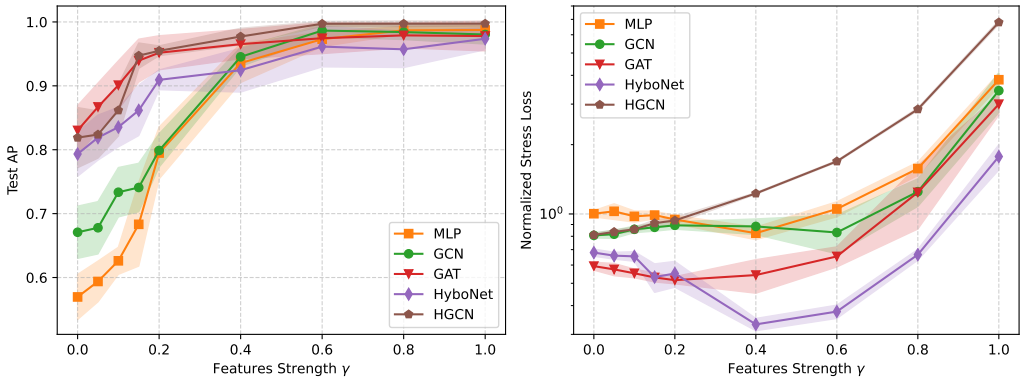
Table 21 shows that both controls increase the score markedly on Tree and Grid, indicating that the diagnostic detects the loss of task-side smoothness. Airport exhibits a smaller separation between observed and randomized labels, suggesting weaker geometric structure detectable by this diagnostic. Comparisons should be made within each dataset, since normalization makes the absolute score depend on graph diameter and label range.

This diagnostic captures only upper-Lipschitz smoothness, rather than the two-sided distance preservation considered in Section 4. Smooth labels may collapse distances between distinct nodes; consequently, a small score neither establishes geometry–task alignment nor guarantees a hyperbolic advantage. The maximum is also sensitive to extreme pairs. Although the diagnostic requires no



(a) ROC AUC (↑) for LP on $Tree1111_\gamma$ over different γ values.

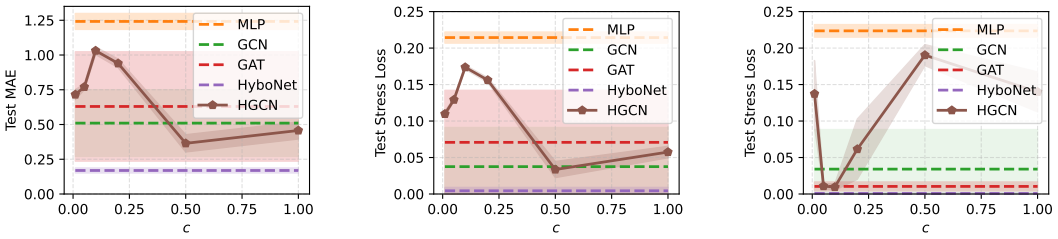
(b) Stress loss (↓) for LP on $Tree1111_\gamma$ over different γ values.



(c) AP (↑) for LP on $Tree1111_\gamma$ over different γ values.

(d) Normalized Stress loss (↓) on $Tree1111_\gamma$ over different γ values.

Figure 8: LP ROC AUC, AP, Stress Loss and Normalized Stress Loss on the synthetic $Tree1111_\gamma$ datasets. γ encodes the features informativeness. The GNNs outperform MLP for small γ and GAT, HGCN and HyboNet perform best, while HyboNet achieves the smallest distortion as measured by the stress loss.



(a) MAE, Dimension 3.

(b) Stress, Dimension 3.

(c) Stress, Dimension 128.

Figure 9: NR performance of HGCN vs. its fixed curvature magnitude c on the synthetic tree dataset. We report NR MAE (↓) for hidden dimension 3 (a) and test stress loss (↓) for hidden dimension 3 (b) and 128 (c).

model training, computing all pairwise graph distances can be costly; sampling pairs reduces this cost but may underestimate the maximum. Robust slope quantiles, lower-distance preservation, and rank correlations between graph and label distances could provide complementary information.

Table 20: Absolute Stress Loss (↓) on the synthetic tree and grid dataset using the embeddings of dimensions 3 and ≤ 128 from Node Regression.

	TREE		GRID		CONTROL TREE
MODEL	3	≤128	3	≤128	≤128
FEATURES	0.1192	0.1192	0.5791	0.5791	0.1192
MLP	0.1226±0.0085	0.1550±0.0100	0.6860±0.0368	0.6121±0.0074	0.1231±0.0030
GCN	0.1439±0.0030	0.1467±0.0105	0.3731±0.0075	0.5625±0.0077	0.1560±0.0096
GAT	0.1539±0.0104	0.1641±0.0121	0.3637±0.0087	0.3667±0.0093	0.1380±0.0105
HYBONET	0.1665±0.0006	0.1775±0.0039	0.6049±0.0534	0.5409±0.0355	0.1291±0.0155
HGCN	0.1486±0.0181	0.1463±0.0270	0.4800±0.0638	0.5007±0.0873	0.1077±0.0038

Table 21: Maximum normalized pairwise slope $L_{\max}$ for observed, permuted, and uniformly randomized labels. Comparisons within each dataset assess how randomization affects target smoothness relative to the graph metric.

Labels	Airport	Tree	Grid
Observed	10.36	2.00	27.00
Permuted	12.00	8.00	46.29
Random uniform	11.88	7.76	52.25

F.4 Node Classification

Real-world datasets. As mentioned in the main text, in addition to the Macro F1 results, we also report accuracy (Acc) to give a more comprehensive picture. The full NC results on the real word datasets are in Table 23 for Macro F1 and Table 24 for Acc. As for the PDP task, we also include HyboNet (Eucl.) as an additional comparison to isolate the effect of model architecture here. The Acc results do not offer a completely new perspective on the model's performance, but merely underline the findings already made for Macro F1. The reason for this is simply because most of these datasets are not very imbalanced, with the exception of Disease. If a model ignores the minority class, the accuracy results can still be good.

On the Disease* dataset with the $70/15/15\%$ split that keeps the ratio of the two classes in all splits, we can see a larger difference between the Macro F1 and Accuracy columns in the two tables (Table 23 and Table 24). All models (with the exception of the MLP) achieve almost equally good performance w.r.t. to accuracy but using Macro F1, the differences become more pronounced. This is not the case for the old Disease results using the $70/5/25\%$ split, with balanced $50/50\%$ ratio in validation and test set, where naturally the F1 and the Accuracy results match more closely. We would also like to note, that the performance advantage of HyboNet is much more pronounced in this setting. This likely arises due to the very few training samples of the minority class in train set (due to the old splitting technique). In this case, the performance comes more from the inductive bias in a heavily restricted sample regime (low sample complexity, which is something we proposed as future work), rather than from geometry of the task. For precisely this reason, we unified the splitting (sufficiently large data to learn in all cases), resulting in a much more fair scenario to isolate the effect of geometry–task alignment.

Again, as for LP, for full disclosure, we report the exact values of Stress Loss for the distortion analysis for NC. The absolute values are in Table 25 and the normalized values are in Table 26. No model is able to significantly reduce distortion for any dataset.

We provide an ablation with alternative decoders (Linear and MLP), with results shown in Table 22. The NC decoder ablation suggests that decoder choice does not change the relative behavior of hyperbolic and Euclidean models.

Synthetic Datasets. We evaluate all our models on the synthetic tree graph we used in our NR experiments, the results are summarized in Table 27. Different from NR, we could not verify the performance advantage of HGNNs in this setting. In the embedding dimension ≤ 128 setting, GCN, HyboNet and HGCN are all able to perform perfect classification. Again they do this without preserving the graph structure as indicated by the (Normalized) Stress Loss. If we constrain the embedding dimension to $d = 3$, the results seem to indicate some advantage of hyperbolic methods,

Table 22: NC decoder ablation (Test Macro F1 ↑).

	MLP DECODER				
MODEL	DISEASE	AIRPORT	CORA	PUBMED	CITESEER
GCN	83.01±6.32	91.38±1.53	74.59±0.96	75.27±1.18	61.86±1.39
GAT	64.31±5.77	89.82±1.57	76.98±1.41	75.65±1.50	62.75±1.71
HYBONET	91.11±3.20	90.21±1.63	71.63±4.16	78.30±1.68	53.24±2.75
HGCN	88.90±5.22	91.19±2.01	77.53±0.91	77.94±1.05	62.92±1.98
	LINEAR DECODER				
GCN	84.43±4.86	91.25±1.53	75.89±0.85	75.78±1.78	63.20±1.74
GAT	63.29±8.34	89.78±1.07	78.79±1.29	75.77±1.56	63.94±1.49
HYBONET	87.13±7.01	90.66±0.90	77.55±1.40	78.18±1.54	61.18±1.54

Table 23: Macro F1 scores (↑) for real-world node classification (NC). We report Gromov δ-hyperbolicity (lower is more hyperbolic).

MODEL	DISEASE $\delta = 0$	DISEASE* $\delta = 0$	AIRPORT $\delta = 2$	CORA $\delta = 4$	PUBMED $\delta = 4.5$	CITESEER $\delta = 6.5$
MLP	77.12±7.53	48.62±5.06	88.79±1.97	55.63±1.50	70.45±1.12	54.29±1.14
GCN	89.14±2.33	89.50±4.82	92.46±0.86	80.09±1.01	78.49±1.47	64.95±1.50
GAT	86.78±9.14	85.71±5.50	88.33±5.73	80.71±1.22	78.44±1.88	65.16±1.48
HYBONET (EUCL.)	89.18±1.56	90.51±3.74	90.54±1.40	78.65±1.04	78.55±1.28	62.51±1.67
HYBONET	94.42±1.70	90.48±5.12	90.87±1.10	78.72±1.01	78.33±1.97	62.98±1.48
HGCN	89.52±1.60	90.51±3.45	90.92±2.04	78.34±0.95	78.21±1.47	63.29±2.07

but with a few caveats. The results suggest that HGCN clearly performs best, with HyboNet following, but checking the standard deviations, GCN could also be among the best. The reason for this that across the 10 splits, all models (except for HGCN and MLP), perform quite well, while on others they basically fail to learn the task. Now the hyperparameters were chosen only across 2 runs, on which all models (except MLP) performed quite well. All in all, the results suggest that in future work a more in-depth analysis is necessary to make a clear statement, maybe also considering sample complexity.

Table 24: Accuracy (↑) results for node classification (NC) tasks. We report Gromov δ-hyperbolicity (lower is more hyperbolic). Note that accuracy on Disease is less meaningful in comparison to the other models than the one of Disease* due to class imbalance effects.

Model	Disease $\delta=0$	Disease* $\delta=0$	Airport $\delta=2$	Cora $\delta=4$	Pubmed $\delta=4.5$	Citeseer $\delta=6.5$
MLP	71.04±4.30	60.45±3.14	89.48±2.70	57.40±1.73	71.03±1.14	56.86±1.02
GCN	89.88±3.04	95.41±2.36	93.18±0.98	81.08±0.76	78.33±1.24	68.45±1.40
GAT	87.69±8.41	94.01±2.28	89.77±4.63	81.86±1.13	77.95±1.79	68.56±1.22
HyboNet (Eucl.)	89.46±1.99	96.11±1.46	91.51±1.35	79.84±1.06	78.04±1.13	66.35±1.30
HyboNet	94.50±1.85	95.99±2.23	92.05±1.54	79.89±1.14	77.91±1.81	66.76±1.95
HGCN	89.92±1.28	96.18±1.40	91.69±1.73	79.48±0.66	78.51±1.61	66.94±1.70

Table 25: Absolute Stress Loss (↓) results of the real-world datasets when using embeddings from NC.

Model	Disease $\delta=0$	Disease* $\delta=0$	Airport $\delta=2$	Cora $\delta=4$	Pubmed $\delta=4.5$	Citeseer $\delta=6.5$
Features	0.2301	0.2301	0.0907	0.1508	0.0754	0.2584
MLP	0.4700±0.1301	0.4249±0.1156	0.0833±0.0013	0.2820±0.0077	0.0754±0.0000	0.2572±0.0005
GCN	0.3098±0.0298	0.3465±0.0353	0.0930±0.0016	0.1495±0.0001	0.0754±0.0000	0.2652±0.0007
GAT	0.3057±0.0829	0.3447±0.0677	0.0832±0.0101	0.1503±0.0004	0.0754±0.0000	0.2698±0.0009
HyboNet (Eucl.)	0.2858±0.0270	0.3216±0.0184	0.0937±0.0011	0.1563±0.0001	0.0754±0.0000	0.2701±0.0007
HyboNet	0.2778±0.0270	0.2457±0.0219	0.0644±0.0042	0.1511±0.0001	0.0754±0.0000	0.2653±0.0011
HGCN	0.2861±0.0245	0.2627±0.0324	0.0834±0.0019	0.1512±0.0027	0.0754±0.0000	0.2578±0.0045

Table 26: Normalized Stress Loss (↓) results of the real-world datasets when using embeddings from NC. Values are normalized such that 1 corresponds to the same loss as with the distances using directly the features.

Model	Disease $\delta=0$	Disease* $\delta=0$	Airport $\delta=2$	Cora $\delta=4$	Pubmed $\delta=4.5$	Citeseer $\delta=6.5$
MLP	2.04±0.57	1.85±0.50	0.93±0.01	1.87±0.05	1.00±0.00	0.99±0.00
GCN	1.35±0.13	1.51±0.15	1.04±0.02	0.99±0.00	1.00±0.00	1.03±0.00
GAT	1.33±0.36	1.50±0.29	0.93±0.11	1.00±0.00	1.00±0.00	1.04±0.00
HyboNet (Eucl.)	1.24±0.12	1.40±0.08	1.05±0.01	1.04±0.00	1.00±0.00	1.05±0.00
HyboNet	1.21±0.12	1.07±0.10	0.72±0.05	1.00±0.00	1.00±0.00	1.03±0.00
HGCN	1.24±0.11	1.14±0.14	0.93±0.02	1.00±0.02	1.00±0.00	1.00±0.02

Table 27: Macro F1 scores and Accuracy (↑) and (Normalized) Stress Loss (↓) for node classification (NC) on the synthetic tree dataset also used for node regression. The features loss for the normalization is 0.1192.

	Macro F1		Accuracy		Stress Loss		Norm. Stress Loss	
Model	**3**	**≤128**	**3**	**≤128**	**3**	**≤128**	**3**	**≤128**
MLP	22.69±2.27	21.80±1.79	35.15±9.53	35.24±8.70	0.1341±0.0088	0.3490±0.3242	1.13±0.07	2.93±2.72
GCN	70.62±29.87	100.00±0.00	75.75±26.36	100.00±0.00	0.1447±0.0108	0.1489±0.0044	1.21±0.09	1.25±0.04
GAT	69.09±18.16	69.96±12.10	69.09±18.16	63.36±14.49	0.1433±0.0438	0.1352±0.0098	1.20±0.37	1.13±0.08
HyboNet	79.78±31.81	97.99±3.21	82.78±27.13	98.85±2.10	0.1265±0.0131	0.1134±0.0228	1.06±0.11	0.95±0.19
HGCN	97.02±1.73	100.00±0.00	99.42±1.73	100.00±0.00	0.1457±0.0125	0.1304±0.0059	1.22±0.11	1.09±0.05

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer:[Yes]

   Justification: See Abstract and Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We do provide in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 3.1, 4.1 5.1, 5.2 and Appendix E.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Sections 3.1, 4.1 5.1, 5.2 and Appendix E.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report variability estimates for all experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix E.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research presented in this paper fully complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, codebases, and prior models used in this work are properly credited through citations to the original sources.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce new experimental code and data processing pipelines, which are made available in an anonymized repository to ensure reproducibility.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, crowdsourcing, or any form of user study.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.