The Fortieth AAAI Conference on Artificial Intelligence (AAAI-26)

Adaptive Riemannian Graph Neural Networks

Xudong Wang[1], **Chris Ding**[1], **Tongxin Li**[1], **Jicong Fan**[1*]

[1]School of Data Science, The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), China
xudongwang@link.cuhk.edu.cn, {chrisding,litongxin,fanjicong}@cuhk.edu.cn

Abstract

Graph data often exhibits complex geometric heterogeneity, where structures with varying local curvature, such as tree-like hierarchies and dense communities, coexist within a single network. Existing geometric GNNs, which embed graphs into single fixed-curvature manifolds or discrete product spaces, struggle to capture this diversity. We introduce Adaptive Riemannian Graph Neural Networks (ARGNN), a novel framework that learns a *continuous and anisotropic Riemannian metric tensor field over the graph*. It allows each node to determine its optimal local geometry, enabling the model to fluidly adapt to the graph's structural landscape. Our core innovation is an efficient parameterization of the node-wise metric tensor, specializing to a learnable diagonal form that captures directional geometric information while maintaining computational tractability. To ensure geometric regularity and stable training, we integrate a Ricci flow-inspired regularization that smooths the learned manifold. Theoretically, we establish the rigorous geometric evolution convergence guarantee for ARGNN and provide a continuous generalization that unifies prior fixed or mixed-curvature GNNs. Empirically, our method demonstrates superior performance on both homophilic and heterophilic benchmark datasets with the ability to capture diverse structures adaptively. Moreover, the learned geometries both offer interpretable insights into the underlying graph structure and empirically corroborate our theoretical analysis.

1 Introduction

Real-world graphs, from social networks to protein interaction maps, exhibit a rich geometric diversity that challenges various graph learning paradigms (Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017; Sun, Ding, and Fan 2023; Kang et al. 2023; Wang and Fan 2024; Sun et al. 2024; Sun and Fan 2024; Wang et al. 2025a; Qian et al. 2025; Wang et al. 2025b; Guo et al. 2025; Fan 2025a,b). Consider a social network where some communities form deep, tree-like hierarchies best captured by hyperbolic geometry (Chami et al. 2019), while others create dense, tightly-knit cliques that resemble spherical manifolds (Gu et al. 2018). Forcing such a geometrically heterogeneous graph into a single geometric space (Euclidean, hyperbolic, or spherical) inevitably introduces significant distortion and information loss.

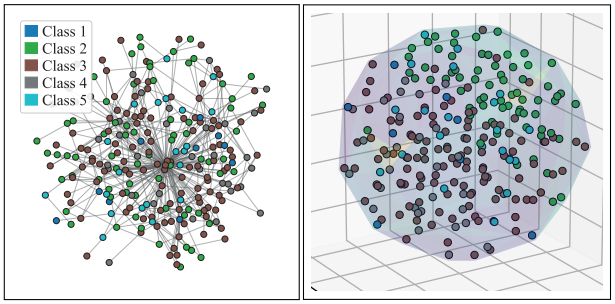

Figure 1: **Geometric heterogeneity in WISCONSIN network. Left:** raw graph topology coloured by class. **Right:** 3-D t-SNE of node features. The translucent hull is coloured by the magnitude of discrete mean curvature (violet $\rightarrow$ flat, yellow $\rightarrow$ strongly curved), showing that curvature varies across the *Riemannian manifold*.

Pioneering work in geometric Graph Neural Networks (GNNs) has demonstrated the power of non-Euclidean spaces. Hyperbolic GNNs (Chami et al. 2019; Zhang et al. 2021) excel on tree-like graphs but struggle with dense cycles. Conversely, spherical methods handle cyclic structures well but are less effective on hierarchical data. This fundamental limitation stems from their shared assumption of *global geometric homogeneity*.

Recent advances have moved towards mixed-curvature approaches to address this issue. Models like κ-GCN (Bachmann, Bécigneul, and Ganea 2020) learn an isotropic (scalar) curvature for each node, while state-of-the-art methods like CUSP (Grover et al. 2025) embed graphs into product manifolds of constant-curvature spaces (e.g., $\mathbb{H}^k \times \mathbb{S}^l \times \mathbb{E}^m$). While a significant step forward, these methods remain constrained. Scalar curvature approaches are still isotropic at the node level, unable to capture directional geometric information. Product manifold methods are limited to block-constant geometries chosen from a discrete set of curvatures. Neither can fully capture the fine-grained, continuous geometric variations inherent in complex data.

This geometric heterogeneity is not merely a theoretical

*Corresponding author.
This is the full version of the paper published in *The 40th Annual AAAI Conference on Artificial Intelligence (AAAI-2026)*, Main Technical Track.

construct; it is evident in real-world data, as illustrated in Figure 1. The real-world WISCONSIN network (Craven et al. 2000) embeds into a Riemannian feature manifold whose neighbourhoods can differ sharply in both curvature and orientation. Any single or fixed-curvature space would therefore distort part of the data, motivating our node-wise, adaptive metric tensor.

The importance of geometric understanding has been further highlighted by recent advances in curvature-based graph analysis. Graph Neural Ricci Flow (GNRF) (Chen et al. 2025) demonstrates that evolving features according to discrete Ricci curvature improves representation quality, though it keeps the underlying geometry fixed. Discrete curvature methods like CurvDrop (Liu et al. 2023) and curvature-based graph rewiring (Topping et al. 2021) show promise but operate as preprocessing steps rather than end-to-end learning frameworks.

Our key insight is to move beyond pre-defined geometries by learning a **continuous, node-adaptive Riemannian metric field** that directly models the local structure. To realize this vision within a tractable framework, we introduce **Adaptive Riemannian Graph Neural Networks** (ARGNN). Our framework proposes a principled and efficient parameterization of the metric tensor. Specifically, we model each node's local geometry with an *anisotropic diagonal metric*, $\mathbf{G}_i = \mathrm{diag}(\mathbf{g}_i)$. This design is not merely a computational shortcut; it corresponds to a flexible, per-dimension scaling of the feature space geometry. To ensure stable learning, we further integrate Ricci flow-inspired dynamics that regularize the evolution of the learned geometric manifold. Implementation code is available at our public repository[1]. The contributions of our work are threefold:

i) **Methodologically**, we propose a principled parameterization of the adaptive metric field using diagonal matrices. We provide strong geometric and algorithmic justifications for our design, framing it as an anisotropic conformal transformation that is both computationally efficient and highly expressive. To the best of our knowledge, ARGNN is the first framework that learns continuous and anisotropic metric tensor fields for graphs.

ii) **Theoretically**, we establish the convergence guarantee for ARGNN's geometric evolution and further prove its role as a universal framework that generalizes and unifies prior isotropic, scalar-curvature, and product-manifold GNNs.

iii) **Empirically**, we conduct extensive experiments on a wide range of benchmark datasets, including comprehensive ablation studies. Our results demonstrate that ARGNN achieves superior performance compared to state-of-the-art benchmark methods on both homophilic and heterophilic graphs.

2 Related Work

2.1 Fixed-Curvature Geometric GNNs

Geometric deep learning on graphs has been revolutionized by embedding approaches that leverage non-Euclidean geometries. **Hyperbolic GNNs** exploit the exponential volume growth of hyperbolic space to naturally represent hierarchical structures. HGCN (Chami et al. 2019) extends graph convolutions to hyperbolic space using the Poincaré ball model, achieving remarkable success on tree-like graphs. HGAT (Zhang et al. 2021) incorporates attention mechanisms in hyperbolic space, while recent work explores hyperbolic transformers (Gulcehre et al. 2018) and variational autoencoders (Sun et al. 2021).

Spherical GNNs address the complementary challenge of cyclic and community structures. Spherical CNNs (Cohen et al. 2018) and their graph extensions (Gu et al. 2018) leverage the positive curvature of spherical space to model dense, interconnected communities. However, both hyperbolic and spherical approaches assume global geometric homogeneity, limiting their applicability to mixed-topology graphs.

Fixed-curvature methods have the advantage compared with traditional GNNs in different target domains but fail catastrophically when graph geometry mismatches the embedding space, as shown in Figure 1, which prevents them from effectively handling the geometric heterogeneity prevalent in real-world networks.

2.2 Discrete Mixed-Curvature Approaches

Recognizing the limitations of fixed-curvature methods, recent work explores mixed-curvature embeddings.

κ-GCN (Bachmann, Bécigneul, and Ganea 2020) learns scalar curvature parameters for each node, allowing adaptation between hyperbolic, Euclidean, and spherical geometries. However, scalar curvature cannot capture directional geometric information.

CUSP (Grover et al. 2025) represents the current state-of-the-art in mixed-curvature approaches. It combines spectral graph analysis with embeddings in product manifolds of constant-curvature spaces (e.g., $\mathbb{H}^k \times \mathbb{S}^l \times \mathbb{E}^m$). CUSP also introduces spectral filtering techniques and curvature-aware graph Laplacians, achieving strong performance across diverse benchmarks.

$\mathcal{Q}$-GCN (Xiong et al. 2022) extends the framework to pseudo-Riemannian manifolds with indefinite metrics, enabling more flexible curvature combinations. Self-supervised mixed-curvature methods (Jin et al. 2020) explore representation learning without labels.

However, despite their flexibility, these approaches primarily treat geometry as *discrete choices* from finite sets of constant-curvature manifolds. Consequently, they offer limited support for continuous geometric variation and anisotropic (directional) structure.

2.3 Curvature-Based Graph Analysis

Recent advances in discrete differential geometry have enabled sophisticated graph analysis through curvature.

Discrete Ricci curvature, including Ollivier-Ricci (Ollivier 2009) and Forman-Ricci (Forman 2003) curvature, provide local geometric characterizations of graph structure.

Graph Neural Ricci Flow (GNRF) (Chen et al. 2025) represents a significant recent advance. GNRF evolves node features according to discrete Ricci curvature, showing that curvature-aware feature evolution improves representation

[1]https://github.com/MathAdventurer/ARGNN

quality. However, GNRF keeps the underlying graph geometry fixed while evolving features.

Curvature-based rewiring methods (Topping et al. 2021; Fesser and Weber 2024) use curvature analysis to identify and modify graph bottlenecks. CurvDrop (Liu et al. 2023) employs Ricci curvature for topology-aware dropout sampling. These methods show curvature's utility but operate as preprocessing steps rather than end-to-end learning.

Curvature-based methods have found success in protein analysis (Wu et al. 2023; Shen et al. 2024), community detection (Ni et al. 2019), and graph generation (Li et al. 2022). However, to our knowledge, *joint learning* of geometry and features remains rare; methods typically fix geometry and evolve features (GNRF) or use curvature for preprocessing.

3 Preliminaries

3.1 Riemannian Geometry Essentials

A **Riemannian manifold** $(\mathcal{M}, \mathbf{g})$ consists of a smooth manifold $\mathcal{M}$ equipped with a metric tensor field $\mathbf{g}$ that varies smoothly across the manifold (Lee 2018). At each point $p \in \mathcal{M}$, the metric tensor $\mathbf{g}_p$ defines an inner product on the tangent space $\mathcal{T}_p\mathcal{M}$, enabling the measurement of distances, angles, and curvatures.

For graph-structured data, we introduce a **metric tensor field over the graph** by associating each node $i \in \mathcal{V}$ with a metric tensor $\mathbf{G}_i \in \mathcal{S}^d_{++}$, where $\mathcal{S}^d_{++}$ denotes the cone of symmetric positive definite matrices. This creates a continuous geometric structure over the discrete graph domain and makes Christoffel symbols vanish, where each node's feature neighborhood in $\mathbb{R}^d$ possesses its own Riemannian geometry. The **geodesic distance** between sufficiently close points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ under the metric $\mathbf{G}$ of $\mathcal{T}_\mathbf{x}\mathcal{M}$ is:

$$d_\mathbf{G}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \mathbf{G} (\mathbf{x}-\mathbf{y})}$$

The **Ricci curvature tensor** $\mathrm{Ric}(\mathbf{G})$ characterizes how the metric changes across the manifold. In differential geometry, **Ricci flow** (Chow and Knopf 2004) evolves a metric tensor field $\{\mathbf{G}_i(t)\}$ according to $\frac{\partial \mathbf{G}_i}{\partial t} = -2\mathrm{Ric}(\mathbf{G}_i)$, which smooths geometric irregularities while preserving topological structure, providing a principled approach for geometric regularization on graphs (Hamilton 1982). Due to space limitations, more Riemannian Geometry and Ricci curvature details are shown in Appendix B.

3.2 GNNs and Geometric Message Passing

Consider an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with nodes $\mathcal{V}$, edges $\mathcal{E}$, and node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$. Standard message passing (Gilmer et al. 2017) of GNNs updates node representations through:

$$\mathbf{h}_i^{(\ell+1)} = \sigma\big(\mathbf{W}_s^{(\ell)} \mathbf{h}_i^{(\ell)} + \sum_{j \in \mathcal{N}(i)} \mathbf{W}_m^{(\ell)} \mathbf{h}_j^{(\ell)}\big)$$

where $\mathcal{N}(i)$ denotes the neighborhood of node i, $\mathbf{W}_s^{(\ell)}$ and $\mathbf{W}_m^{(\ell)}$ are learnable transformation matrices (parameters) on ℓ-th layer, and σ is a nonlinear activation function.

Geometric message passing enhances this by incorporating Riemannian structure by embedding graphs in non-Euclidean spaces. Existing approaches embed graphs in fixed geometries (hyperbolic, spherical) or discrete mixed-curvature spaces. However, as illustrated in Figure 1, real graphs exhibit **continuous geometric heterogeneity** and the local geometry varies across the network.

3.3 Problem Formulation and Challenges

The core challenge of this work is to move beyond the fixed-geometry paradigm. We aim to learn a node-adaptive geometric space jointly with the node representations $\mathbf{H} = \{\mathbf{h}_i\}_{i \in \mathcal{V}}$. This learned space is mathematically formulated as a Riemannian metric tensor field $\{\mathbf{G}_i \in \mathcal{S}^d_{++}\}_{i \in \mathcal{V}}$, where each node i is endowed with its own local metric $\mathbf{G}_i$. However, realizing this vision presents several key hurdles.

First, from a mathematical standpoint, each learned tensor $\mathbf{G}_i$ must remain symmetric positive definite (SPD) throughout training to constitute a valid Riemannian metric. **Second**, from a computational perspective, parameterizing a full $d \times d$ metric tensor for every node is prohibitive, scaling as $O(|\mathcal{V}|d^2)$ and hindering application to large graphs. **Third**, for learning stability, the geometric manifold must evolve smoothly; pathological curvatures could destabilize the training process and lead to poor generalization. **Finally**, to justify its increased complexity, the proposed framework must be provably more expressive than its fixed-geometry predecessors. Our ARGNN framework is designed to address these challenges systematically.

4 Adaptive Riemannian Graph Neural Networks

This section introduces Adaptive Riemannian Graph Neural Networks (ARGNN), a framework that learns a continuous, node-adaptive geometry tailored to the underlying graph structure. Our approach forgoes the manual selection of a geometric space and instead learns the geometry as part of the end-to-end training process. This section details the three core components of our framework: i) an efficient and interpretable parameterization of the node-wise metric tensor via a local metric estimator, ii) a geometric message passing scheme that leverages the learned metric, and iii) a Ricci flow-inspired regularization for stable geometric evolution. Figure 2 illustrates our approach's diagram.

4.1 Learning an Anisotropic Metric Field

The foundational concept of our work is to equip the graph with a **Riemannian metric tensor field**, denoted as $\{\mathbf{G}_i \in \mathcal{S}^d_{++}\}_{i \in \mathcal{V}}$. Each metric tensor $\mathbf{G}_i$ is a symmetric positive-definite (SPD) matrix that defines a unique local geometry on the feature space neighborhood of node i. As stated before, a direct parameterization of a full $d \times d$ matrix for each node will be computationally prohibitive and lead to expensive downstream computations.

To address this, we propose a **principled simplification** by parameterizing each metric tensor $\mathbf{G}_i$ as a learnable **diagonal matrix**:

$$\mathbf{G}_i = \mathrm{diag}(\mathbf{g}_i) = \mathrm{diag}(g_{i,1}, g_{i,2}, \ldots, g_{i,d}) \tag{1}$$

where $\mathbf{g}_i \in \mathbb{R}^d_{++}$ is a vector of positive diagonal elements. This diagonal parameterization is not an arbitrary choice but a motivated design with several key advantages:

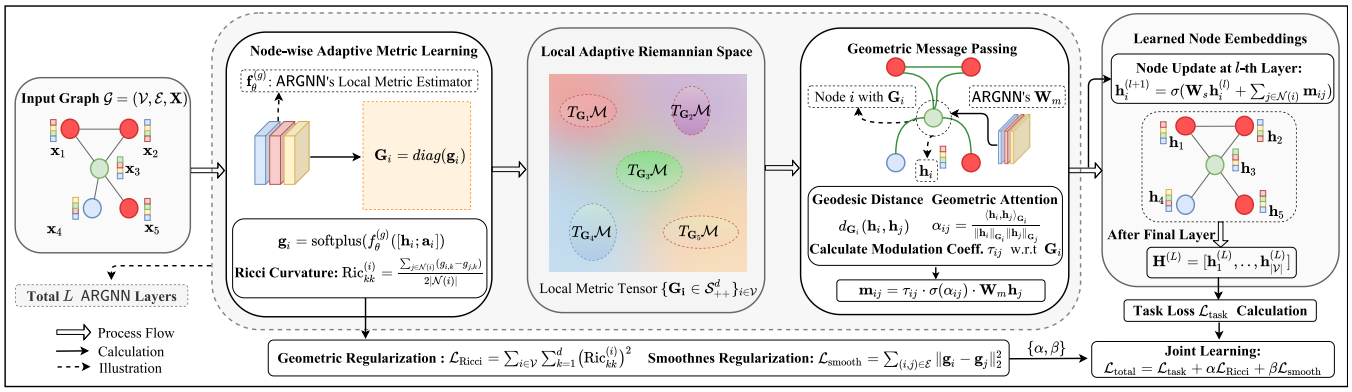

Figure 2: Diagram of our proposed ARGNN, which jointly learns continuous, anisotropic metric tensor fields $\{\mathbf{G}_i \in \mathcal{S}_{++}^d\}_{i \in \mathcal{V}}$ and node embeddings $\mathbf{H} = \{\mathbf{h}_i\}_{i \in \mathcal{V}}$. The learned $\mathbf{G}_i$ gives beyond curvature information to depict the geometric diversity.

i) **Geometric Interpretation:** At its core, a metric tensor defines how to measure distances, angles, and curvatures. The standard Euclidean space, governed by the identity matrix, provides a single, global way of measurement. Our diagonal metric, $\mathbf{G}_i = \text{diag}(\mathbf{g}_i)$, acts as a local, learned modification to this standard. Specifically, it *rescales the geometry independently along each feature axis*. The value of each diagonal element $g_{i,k}$ determines the stretch of the k-th dimension in the local vicinity of node i. This axis-aligned, non-uniform scaling type is formally known as an **anisotropic conformal transformation** (Obata 1970; Spivak 1999). It endows the model with far greater flexibility than the isotropic scaling of constant-curvature spaces, while remaining more structured and tractable than a full metric tensor. The formal details of this geometric perspective are elaborated in Appendix B.

ii) **Interpretability and Feature Decomposition:** The diagonal form assumes that the standard coordinate axes of the feature space align with the principal directions of the local geometry. This makes the model highly interpretable: each diagonal element $g_{i,k}$ directly quantifies the geometric importance or local scaling factor of the k-th feature dimension for node i. This is particularly effective when features are learned to be relatively disentangled.

iii) **Computational and Algorithmic Efficiency:** This parameterization dramatically reduces the complexity of learning and using the metric. The number of geometric parameters per node drops from $O(d^2)$ to $O(d)$. Furthermore, crucial geometric computations such as geodesic distance, matrix inversion, and eigendecomposition become highly efficient, which we will discuss later.

To learn these adaptive metrics, we employ a small neural network $f_\theta^{(g)}$ as the local metric estimator that maps a node's local structural information to its metric vector $\mathbf{g}_i$. Specifically, for each node i, we first aggregate its neighborhood features $\mathbf{a}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j$, and then compute its metric vector as:

$$\mathbf{g}_i = \text{softplus}\left(f_\theta^{(g)}([\mathbf{h}_i; \mathbf{a}_i])\right) \tag{2}$$

where $[\cdot;\cdot]$ denotes feature concatenation. The softplus activation function ensures all elements of $\mathbf{g}_i$ are strictly positive, thereby naturally satisfying the SPD constraint for $\mathbf{G}_i$.

4.2 Geometric Message Passing

With the learned diagonal metric field, we now formulate a message passing scheme that is explicitly geometry-aware. The propagation of information between nodes is modulated by the local geometries, allowing the model to adapt its aggregation strategy based on the learned structure.

Geodesic Distance and Principal Curvatures Under the learned metric $\mathbf{G}_i$, the geodesic distance (in this locally constant metric space) between the feature vectors of two nodes i and j simplifies to a weighted Euclidean distance:

$$\begin{aligned} d_{\mathbf{G}_i}(\mathbf{h}_i, \mathbf{h}_j) &= \sqrt{(\mathbf{h}_i - \mathbf{h}_j)^T \mathbf{G}_i (\mathbf{h}_i - \mathbf{h}_j)} \\ &= \sqrt{\sum_{k=1}^{d} g_{i,k} (h_{i,k} - h_{j,k})^2} \end{aligned} \tag{3}$$

A major benefit of the diagonal form is that the principal directions of the geometry are aligned with the standard basis vectors $\{\mathbf{e}_k\}_{k=1}^d$, and the principal curvatures are directly related to the inverses of the metric elements $\{1/g_{i,k}\}_{k=1}^d$. This allows us to define a geometric modulation factor that captures how the geometry curves along the direction of the message.

Geometric Modulation and Attention We introduce a geometric modulation coefficient τ_{ij} that weights the message from node j to i based on the projection of their directional vector onto the principal axes, modulated by the local curvature in those directions. Let $\mathbf{d}_{ij} = (\mathbf{h}_j - \mathbf{h}_i)/\|\mathbf{h}_j - \mathbf{h}_i\|_2$ be the normalized direction vector. We define τ_{ij} as:

$$\tau_{ij} = \sum_{k=1}^{d} d_{ij,k}^2 \cdot \tanh(-\log g_{i,k}) \tag{4}$$

Here, the term $\tanh(-\log g_{i,k})$ acts as a curvature-dependent switch. If $g_{i,k}$ is large (high curvature, space con-

tracts), the term approaches -1; if $g_{i,k}$ is small (low curvature, space expands), it approaches $+1$.

Complementing this, we introduce a geometric attention mechanism α_{ij} that computes the similarity between nodes within their respective local geometries:

$$\alpha_{ij} = \frac{\langle \mathbf{h}_i, \mathbf{h}_j \rangle_{\mathbf{G}_i}}{\|\mathbf{h}_i\|_{\mathbf{G}_i} \|\mathbf{h}_j\|_{\mathbf{G}_j}} = \frac{\sum_k g_{i,k} h_{i,k} h_{j,k}}{\sqrt{\sum_k g_{i,k} h_{i,k}^2} \sqrt{\sum_k g_{j,k} h_{j,k}^2}} \tag{5}$$

This formulation measures the cosine similarity where the inner product is defined by node i's metric, and each node's norm is measured in its own local metric.

Node Representation Update The final message $\mathbf{m}_{ij}$ from node j to i integrates the geometric modulation and attention with a standard learnable transformation $\mathbf{W}_m$:

$$\mathbf{m}_{ij} = \tau_{ij} \cdot \sigma(\alpha_{ij}) \cdot \mathbf{W}_m \mathbf{h}_j \tag{6}$$

The node representations are then updated by aggregating messages from neighbors and combining them with the transformed self-representation, followed by a non-linear activation σ:

$$\mathbf{h}_i^{(l+1)} = \sigma\big(\mathbf{W}_s \mathbf{h}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\big) \tag{7}$$

4.3 Ricci Flow-Inspired Geometric Regularization

To ensure that the learned metric field $\{\mathbf{G}_i\}$ is well-behaved and varies smoothly across the graph, we introduce two regularization terms inspired by the principles of discrete Ricci flow. This geometric evolution stabilizes training and prevents pathological curvatures.

First, we define a discrete approximation of the Ricci curvature for our diagonal metric field. Along the k-th principal direction at node i, let Ric_{kk} be the abbreviation of $\mathrm{Ric}(\mathbf{G}_i)_{kk}$, it can be approximated as:

$$\mathrm{Ric}_{kk}^{(i)} = \frac{1}{2|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{g_{i,k} - g_{j,k}}{d_{\text{graph}}(i,j)}. \tag{8}$$

where $d_{\text{graph}}(i,j)$ is the shortest path distance on the graph, and node $j \in \mathcal{N}(i)$ makes $d_{\text{graph}} = 1$. See the detailed discussion in Appendix B.3. The **Ricci regularization** $\mathcal{L}_{\text{Ricci}}$ encourages the learned geometry to be Ricci-flat by penalizing the squared sum of Ricci curvatures, promoting a uniform curvature distribution:

$$\mathcal{L}_{\text{Ricci}} = \sum_{i \in \mathcal{V}} \sum_{k=1}^{d} (\mathrm{Ric}_{kk}^{(i)})^2 \tag{9}$$

The **smoothness regularization** $\mathcal{L}_{\text{smooth}}$ penalizes large differences in the metric vectors of adjacent nodes, ensuring the geometry varies smoothly over the graph structure:

$$\mathcal{L}_{\text{smooth}} = \sum_{(i,j) \in \mathcal{E}} \|\mathbf{g}_i - \mathbf{g}_j\|_2^2 \tag{10}$$

These regularizers guide the model to learn a coherent and stable geometric manifold that supports the downstream task. The total loss function then combines the primary task loss $\mathcal{L}_{\text{task}}$ with hyperparameters α, β:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \alpha \mathcal{L}_{\text{Ricci}} + \beta \mathcal{L}_{\text{smooth}} \tag{11}$$

5 Theoretical Analysis

We establish theoretical foundations for ARGNN, proving its convergence, stability analysis, and universality. All detailed proofs are provided in Appendix C.

5.1 Convergence and Stability Analysis

Theorem 1 (Convergence of Adaptive Geometry Learning)**.** *Consider ARGNN with L layers, hidden dimension d, and regularization weights α, β. Under mild regularity conditions, the learned metric tensors $\{\mathbf{G}_i\}_{i \in \mathcal{V}}$ converge to a stationary point with:*

$$\mathbb{E}\big[\|\nabla \mathcal{L}_{\text{total}}^{(t)}\|^2\big] = O\Big(\tfrac{1}{\sqrt{t}}\, e^{-\mu_{\text{eff}} t / L}\Big), \tag{12}$$

where μ_{eff} is the effective curvature of the loss landscape. Optimal regularization hyperparameters satisfy:

$$\alpha^\star = \Theta\big(\tfrac{\mathcal{H}}{L} \min\big(1, \tfrac{d}{|\mathcal{E}|}\big)\big), \quad \beta^\star = \Theta\big(\tfrac{\mathcal{H}\sqrt{d}}{|\mathcal{V}|}\big). \tag{13}$$

where $\mathcal{H} \in (0, 1]$ is the dataset-dependent homophily ratio.

Assume the theoretical optimal hyperparameters as

$$\alpha = \tfrac{c_1}{L} \cdot \min\big(1, \tfrac{d}{|\mathcal{E}|}\big), \quad \beta = \tfrac{c_2 \sqrt{d}}{|\mathcal{V}|} \tag{14}$$

For constants c_1, c_2, we get an effective practice as follows,

Proposition 1 (Homophily-Aware Constants)**.** *The dataset-dependent constants c_1, c_2 can be estimated as:*

$$c_1 \approx (1 - \mathcal{H}) + 0.1, \quad c_2 \approx 0.1 \cdot (1 + \mathcal{H}) \tag{15}$$

where $\mathcal{H} \in (0, 1]$ is the graph homophily ratio.

The proof (Appendix C.1) reveals key insights: i) Deeper networks require smaller α to maintain stability; ii) The smoothness weight β should scale with feature dimension and inversely with graph size; iii) The interplay between geometric regularization and task loss creates a favorable optimization landscape. These theoretical guidelines directly inform our hyperparameter choices in experiments.

5.2 Universal Approximation

Theorem 2 (Universal Geometric Framework)**.** *ARGNN provides a universal geometric framework that can generalize existing curvature-based GNNs. Specifically, GNNs operating on a fixed-curvature space (Euclidean, hyperbolic, spherical, or product manifolds) can be sufficiently approximated by a constrained parameterization of our learnable diagonal metric tensors $\mathbf{G}_i = diag(\mathbf{g}_i)$.*

See Appendix C.2 for the complete proof. We establish this by showing that fixed-curvature geometries correspond to specific constraints on $\mathbf{g}_i$: Euclidean ($\mathbf{g}_i = \mathbf{1}$), constant-curvature ($\mathbf{g}_i = c\mathbf{1}, c > 0$), and product manifolds (block-constant $\mathbf{g}_i$).

5.3 Computational Efficiency

Proposition 2 (Complexity Analysis)**.** *ARGNN has time complexity $O((n + m)d^2)$ per layer, where $n = |\mathcal{V}|$, $m = |\mathcal{E}|$, and d is the hidden feature dimension, matching standard GNNs while providing continuous geometric adaptation.*

See detailed proof in Appendix C.3. We also give the empirical comparison and analysis in Appendix E.

Dataset	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
Nodes	2,708	3,327	19,717	7,600	2,277	5,201	183	183	251
Edges	5,278	4,552	44,324	26,659	31,371	198,353	279	277	466
Features	1,433	3,703	500	932	2,325	2,089	1,703	1,703	1,703
Classes	7	6	3	5	5	5	5	5	5
$\mathcal{H}$	0.825	0.718	0.792	0.215	0.247	0.217	0.057	0.301	0.196

Table 1: Dataset Statistics and Homophily Ratios $\mathcal{H}$

6 Experiments

6.1 Experimental Setup

Datasets: We evaluate on nine widely used benchmark datasets spanning homophilic and heterophilic graphs: **Homophilic**: Cora, CiteSeer, PubMed (Sen et al. 2008); **Heterophilic**: Actor (Tang et al. 2009), Chameleon, Squirrel, Texas, Cornell, Wisconsin (Pei et al. 2020). Table 1 provides detailed statistics.

Baselines: We compare against four categories of methods: i) **Traditional GNNs**: GCN (Kipf and Welling 2016), GAT (Veličković et al. 2018), GraphSAGE (Hamilton, Ying, and Leskovec 2017); ii) **Geometric GNNs**: HGCN (Chami et al. 2019), HGAT (Zhang et al. 2021), κ-GCN (Bachmann, Bécigneul, and Ganea 2020), $\mathcal{Q}$-GCN (Xiong et al. 2022); iii) **Heterophilic GNNs**: H2GCN (Zhu et al. 2020), GPRGNN (Chien et al. 2020), FAGCN (Bo et al. 2021); iv) **Recent Methods**: CUSP (Grover et al. 2025), GNRF (Chen et al. 2025), CurvDrop (+JKNet) (Liu et al. 2023).

Implementation Details: We implemented on PyG (Fey and Lenssen 2019) and Geoopt (Kochurov, Karimov, and Kozlukov 2020) frameworks. All experiments were run on NVIDIA GPUs with 10 random seeds, with the given $60\%/20\%/20\%$ splits from (Pei et al. 2020) for the node classification task. And using the same $80\%/5\%/15\%$ splits and settings from (He et al. 2024) for the link existence prediction task.

6.2 Main Results

Table 2 presents mean F1-scores with 95% Confidence Intervals (CI) from 10 runs, chosen as the most comprehensive metric for multi-class imbalanced datasets. Additional metrics (Accuracy, AUROC, and AUPRC) are provided in Appendix E. ARGNN achieves the best performance on all 9 datasets, with significant improvements across both homophilic and heterophilic graphs. For instance, on Cora (homophilic), we improve 3.38% over CUSP, while on Wisconsin (heterophilic), we gain 2.35% over the best baseline.

Our results reveal three key patterns: i) Fixed-curvature methods like HGCN show high variance on heterophilic graphs, but generally overperform Euclidean-space GNNs; ii) Mixed-curvature approaches (CUSP, GNRF) achieve more consistent results but remain limited by discrete geometry choices; iii) ARGNN's continuous adaptation provides robust performance across the homophily spectrum.

For link prediction (Table 7 in Appendix), we report average AUROC as the comprehensive metric for link prediction. ARGNN achieves the highest scores on all 9 datasets, with particularly strong performance on geometrically complex graphs (Actor: 76.40% vs. GNRF's 73.50% and CUSP's 74.20%), validating our adaptive metric learning approach.

6.3 Ablation Studies

We conduct systematic ablations to validate our theoretical framework and understand the contribution of each component. Figure 3 presents results on three representative datasets: Cora (homophilic, $\mathcal{H} = 0.825$), Actor (heterophilic, $\mathcal{H} = 0.215$), and Wisconsin (mixed, $\mathcal{H} = 0.196$).

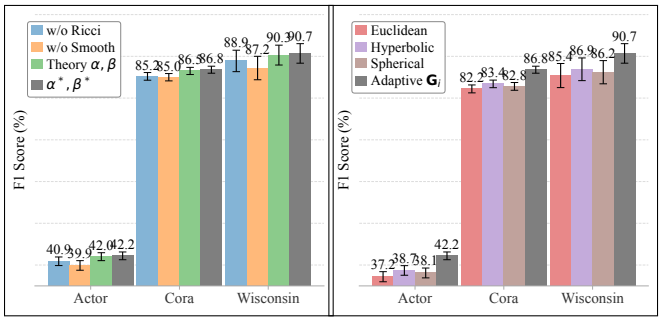

Figure 3: Ablation studies on three datasets. **(a)** Impact of regularization components, including theoretically optimal α, β settings v.s. grid search optimal α^*, β^* **(b)** Comparison with fixed $\mathbf{G}_i \in \{\mathbf{I}, 0.5\mathbf{I}, 2\mathbf{I}\}$ for Euclidean, Hyperbolic and Spherical. Error bars show 95% CI from 10 runs.

Regularization Components (Fig. 3a): Both Ricci and smoothness regularizations are essential but serve different roles. Removing Ricci regularization causes larger performance drops on heterophilic graphs (Actor: -1.3%, Wisconsin: -1.8%), where diverse geometries require careful control. Smoothness regularization is more critical for maintaining coherent metric fields, with Wisconsin showing the largest drop (-3.5%) due to its mixed homophily structure requiring smooth transitions between geometric regimes.

Remarkably, our theory-guided hyperparameters achieve near-optimal performance. Using theoretical optimal $\alpha = \frac{c_1}{L} \cdot \min(1, \frac{d}{|\mathcal{E}|})$ and $\beta = \frac{c_2\sqrt{d}}{|\mathcal{V}|}$ with c_1, c_2 from Theorem 1, we obtain F1 scores within 0.5% of exhaustive grid search. This validates our convergence analysis and provides practitioners with principled hyperparameter selection.

Geometry Parameterizations (Fig. 3b): Fixed geometries severely limit expressiveness across all graph types. Euclidean geometry ($\mathbf{G}_i = \mathbf{I}$), optimal for neither hierar-

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	75.21±0.28	67.30±1.05	83.75±0.07	31.12±0.96	61.16±0.23	43.06±0.33	75.61±0.07	67.72±1.19	59.46±3.25
GAT	76.70±0.13	66.23±0.85	82.83±0.22	32.65±0.23	63.10±0.77	43.90±0.01	76.09±0.77	74.01±0.01	55.29±5.40
GraphSAGE	71.88±0.91	70.01±0.64	81.09±0.13	36.73±0.01	59.99±0.89	41.11±1.16	77.11±0.45	69.91±0.24	81.18±3.45
HGCN	78.50±0.14	69.55±0.39	83.72±0.21	35.89±0.29	60.18±0.57	39.93±0.35	88.11±1.12	72.88±1.15	86.70±3.70
HGAT	77.12±0.01	70.12±0.92	84.02±0.19	35.12±0.27	62.43±0.59	41.78±0.37	85.56±1.10	73.12±0.18	87.20±3.50
κ-GCN	78.71±1.37	68.14±0.34	85.18±0.52	34.57±0.26	62.12±0.49	43.04±0.31	85.03±0.63	86.36±0.64	86.90±3.80
$\mathcal{Q}$-GCN	79.64±0.38	71.15±1.11	84.76±0.13	32.24±0.65	61.83±1.01	46.65±0.90	82.76±0.07	83.90±0.71	86.50±4.10
H2GCN	82.70±0.90	71.10±0.80	84.60±0.50	35.90±1.20	60.10±2.20	48.20±2.00	84.90±6.00	82.20±4.20	86.67±2.91
GPRGNN	79.49±0.31	67.61±0.38	84.07±0.09	37.43±1.09	65.09±0.43	47.51±0.23	88.34±0.09	87.21±0.70	88.07±1.00
FAGCN	82.50±0.50	71.30±0.60	84.30±0.40	35.80±1.10	66.90±1.80	52.30±1.70	87.80±3.20	85.50±4.10	87.30±3.60
CUSP	83.45±0.15	74.21±0.02	87.99±0.45	41.91±0.11	70.23±0.61	52.98±0.25	89.43±2.72	88.31±1.09	88.30±0.80
GNRF	82.10±0.80	73.50±0.50	86.80±0.40	40.80±0.90	68.90±1.20	49.82±1.50	90.80±1.30	90.50±1.10	88.10±1.40
CurvDrop	82.50±0.70	72.80±0.60	85.20±0.50	39.50±1.00	67.30±1.40	50.10±1.30	88.20±2.10	89.80±1.80	87.50±1.90
ARGNN	**86.83**±0.84	**74.80**±1.26	**88.59**±0.25	**42.18**±0.33	**70.44**±1.27	**53.12**±1.45	**92.28**±1.59	**90.85**±0.33	**90.65**±2.34

Table 2: Node Classification Performance (Avg. F1-score %(↑) ± 95% Confidence Interval) on Benchmark Datasets. The **bold**, purple and orange numbers denote the best, second best, and third best performances, respectively.

chies nor cycles, shows the worst performance. Fixed hyperbolic geometry ($\mathbf{G}_i = 0.5\mathbf{I}$) improves slightly on tree-like substructures but fails on dense regions. Around 5% improvement of ARGNN over fixed geometries on heterophilic graphs demonstrates the necessity of adaptive metrics.

The performance gap is most pronounced on Actor (5.0% over Euclidean), where the co-occurrence network contains both star-like patterns (requiring hyperbolic geometry) and cliques (requiring spherical geometry), and our adaptive approach can simultaneously capture these diverse structures. Due to the space limit, detailed ablation results like additional parameterization comparisons and detailed computational overhead analysis are provided in Appendix E.

Computational Efficiency: ARGNN's diagonal parameterization achieves superior efficiency (nearly to HGCN): around 35% faster than CUSP by avoiding costly product manifold projections, with around 40% lower memory usage than full tensor methods. The $O(d)$ complexity per metric operation (v.s. $O(d^2)$ for full tensors) enables scaling to large graphs. Detailed benchmarks and scalability analysis are provided in Appendix E.

6.4 Learned Geometry Analysis

Figure 4 visualizes how ARGNN discovers latent geometric structure. The geodesic rewiring demonstrates improved class separation through learned metrics.

Figure 5 reveals that highly homophilic graphs (Cora, CiteSeer) are in the low curvature and *Neighbour-Relative Metric Dispersion* (NRMD $= \frac{1}{|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} \frac{\|\mathbf{g}_i - \mathbf{g}_j\|_2}{\frac{1}{2}(\|\mathbf{g}_i\|_2 + \|\mathbf{g}_j\|_2)}$) quadrant, whereas heterophilic datasets (Actor, Wisconsin) display both larger curvature and greater metric dispersion. The joint trend confirms that ARGNN bends its geometry more and allows higher learned metric diversity when neighbourhood labels are mixed, underscoring its dataset-adaptive behaviour. Due to the space limitation, see Appendix E for more evidence and analysis.

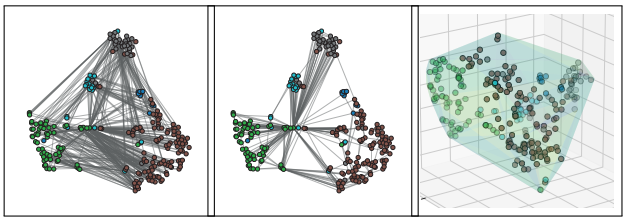

Figure 4: Geometry learned by ARGNN on Wisconsin. **Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet → flat, yellow → strongly curved)

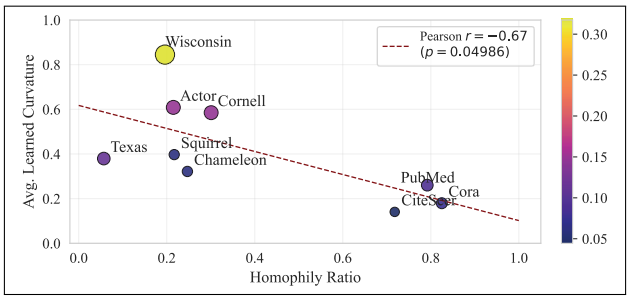

Figure 5: **Homophily $\mathcal{H}$ vs. learned geometry.** Avg. learned curvature across datasets with marker size/colour encodes the mean *Neighbour-Relative Metric Dispersion* (NRMD)

7 Conclusion

We have introduced ARGNN, a novel framework that learns continuous, anisotropic metric tensor fields to capture the geometric diversity inherent in real-world graphs.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No.62376236.

References

Bachmann, G.; Bécigneul, G.; and Ganea, O. 2020. Constant curvature graph convolutional networks. In *International conference on machine learning*, 486–496. PMLR.

Bo, D.; Wang, X.; Shi, C.; and Shen, H. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 3950–3957.

Chami, I.; Ying, Z.; Ré, C.; and Leskovec, J. 2019. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32.

Chen, J.; Deng, B.; Chen, C.; Zheng, Z.; et al. 2025. Graph neural ricci flow: Evolving feature from a curvature perspective. In *The Thirteenth International Conference on Learning Representations*.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2020. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*.

Chow, B.; and Knopf, D. 2004. *The ricci flow: An introduction: An introduction*, volume 1. American Mathematical Soc.

Cohen, T. S.; Geiger, M.; Köhler, J.; and Welling, M. 2018. Spherical CNNs. In *International Conference on Learning Representations*.

Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial intelligence*, 118(1-2): 69–113.

Fan, J. 2025a. Graph Minimum Factor Distance and Its Application to Large-Scale Graph Data Clustering. In *Forty-second International Conference on Machine Learning*.

Fan, J. 2025b. An Interdisciplinary and Cross-Task Review on Missing Data Imputation. *arXiv preprint arXiv:2511.01196*.

Fesser, L.; and Weber, M. 2024. Mitigating over-smoothing and over-squashing using augmentations of forman-ricci curvature. In *Learning on Graphs Conference*, 19–1. PMLR.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

Forman. 2003. Bochner's method for cell complexes and combinatorial Ricci curvature. *Discrete & Computational Geometry*, 29(3): 323–374.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. Pmlr.

Grover, K.; Yu, H.; Song, X.; Zhu, Q.; Xie, H.; Ioannidis, V. N.; and Faloutsos, C. 2025. Spectro-Riemannian Graph Neural Networks. *International Conference on Learning Representations*.

Gu, A.; Sala, F.; Gunel, B.; and Ré, C. 2018. Learning mixed-curvature representations in product spaces. In *International conference on learning representations*.

Gulcehre, C.; Denil, M.; Malinowski, M.; Razavi, A.; Pascanu, R.; Hermann, K. M.; Battaglia, P.; Bapst, V.; Raposo, D.; Santoro, A.; et al. 2018. Hyperbolic attention networks. In *International Conference on Learning Representations*.

Guo, Z.; Sun, Q.; Yuan, H.; Fu, X.; Zhou, M.; Gao, Y.; and Li, J. 2025. GraphMoRE: Mitigating Topological Heterogeneity via Mixture of Riemannian Experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 11754–11762.

Hamilton, R. S. 1982. Three-manifolds with positive Ricci curvature. *Journal of Differential geometry*, 17(2): 255–306.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

He, Y.; Zhang, X.; Huang, J.; Rozemberczki, B.; Cucuringu, M.; and Reinert, G. 2024. Pytorch geometric signed directed: a software package on graph neural networks for signed and directed graphs. In *Learning on Graphs Conference*, 12–1. PMLR.

Jin, W.; Derr, T.; Liu, H.; Wang, Y.; Wang, S.; Liu, Z.; and Tang, J. 2020. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*.

Kang, Q.; Zhao, K.; Song, Y.; Wang, S.; and Tay, W. P. 2023. Node embedding from neural Hamiltonian orbits in graph neural networks. In *International conference on machine learning*, 15786–15808. PMLR.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kochurov, M.; Karimov, R.; and Kozlukov, S. 2020. Geoopt: Riemannian optimization in pytorch. *arXiv preprint arXiv:2005.02819*.

Lee, J. M. 2018. *Introduction to Riemannian manifolds*, volume 2. Springer.

Li, J.; Fu, X.; Sun, Q.; Ji, C.; Tan, J.; Wu, J.; and Peng, H. 2022. Curvature graph generative adversarial networks. In *Proceedings of the ACM web conference 2022*, 1528–1537.

Liu, Y.; Zhou, C.; Pan, S.; Wu, J.; Li, Z.; Chen, H.; and Zhang, P. 2023. Curvdrop: A ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In *Proceedings of the ACM Web Conference 2023*, 221–230.

Ni, C.-C.; Lin, Y.-Y.; Luo, F.; and Gao, J. 2019. Community detection on networks with Ricci flow. *Scientific reports*, 9(1): 9984.

Obata, M. 1970. Conformal transformations of Riemannian manifolds. *Journal of Differential Geometry*, 4(3): 311–333.

Ollivier, Y. 2009. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis*, 256(3): 810–864.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance

deep learning library. *Advances in neural information processing systems*, 32.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*.

Qian, F.; Bai, L.; Cui, L.; Li, M.; Lyu, Z.; Du, H.; and Hancock, E. 2025. DHAKR: Learning Deep Hierarchical Attention-Based Kernelized Representations for Graph Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 19995–20003.

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

Shen, C.; Ding, P.; Wee, J.; Bi, J.; Luo, J.; and Xia, K. 2024. Curvature-enhanced graph convolutional network for biomolecular interaction prediction. *Computational and Structural Biotechnology Journal*, 23: 1016–1025.

Spivak, M. 1999. *A comprehensive introduction to differential geometry*, volume 2. Publish or Perish.

Sun, L.; Zhang, Z.; Zhang, J.; Wang, F.; Peng, H.; Su, S.; and Yu, P. S. 2021. Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4375–4383.

Sun, Y.; and Fan, J. 2024. MMD Graph Kernel: Effective Metric Learning for Graphs via Maximum Mean Discrepancy. In *The Twelfth International Conference on Learning Representations*.

Sun, Z.; Ding, C.; and Fan, J. 2023. Lovász Principle for Unsupervised Graph Representation Learning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 58290–58311. Curran Associates, Inc.

Sun, Z.; Wang, X.; Ding, C.; and Fan, J. 2024. Learning Graph Representation via Graph Entropy Maximization. In *International Conference on Machine Learning*, 47133–47158. PMLR.

Tang, J.; Sun, J.; Wang, C.; and Yang, Z. 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 807–816.

Topping, J.; Di Giovanni, F.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2021. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Villani, C.; et al. 2008. *Optimal transport: old and new*, volume 338. Springer.

Wang, X.; Sun, Z.; Ding, C.; and Fan, J. 2025a. Explainable Graph Representation Learning via Graph Pattern Analysis. In Kwok, J., ed., *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, 3426–3434. International Joint Conferences on Artificial Intelligence Organization.

Wang, X.; Sun, Z.; Ding, C.; and Fan, J. 2025b. Learnable Kernel Density Estimation for Graphs. *arXiv preprint arXiv:2505.21285*.

Wang, Z.; and Fan, J. 2024. Graph classification via reference distribution learning: theory and practice. *Advances in Neural Information Processing Systems*, 37: 137698–137740.

Wu, J.; Chen, H.; Cheng, M.; and Xiong, H. 2023. Curvagn: curvature-based adaptive graph neural networks for predicting protein-ligand binding affinity. *BMC bioinformatics*, 24(1): 378.

Xiong, B.; Zhu, S.; Potyka, N.; Pan, S.; Zhou, C.; and Staab, S. 2022. Pseudo-riemannian graph convolutional networks. *Advances in Neural Information Processing Systems*, 35: 3488–3501.

Zhang, Y.; Wang, X.; Shi, C.; Jiang, X.; and Ye, Y. 2021. Hyperbolic graph attention network. *IEEE Transactions on Big Data*, 8(6): 1690–1701.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33: 7793–7804.

A Notation and Symbols

This appendix provides a comprehensive reference (Table 3) for the principal notation used throughout the paper. While we strive to maintain consistent notation, specialized symbols that appear exclusively in proofs or specific derivations are defined in their respective contexts. We adhere to the following conventions:

Symbol	Definition
Graph Structure and Node Features	
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$	Attributed graph
$\mathcal{V}$	Set of vertices/nodes, $\lvert\mathcal{V}\rvert = n$
$\mathcal{E}$	Set of edges, $\lvert\mathcal{E}\rvert = m$
$\mathbf{X} \in \mathbb{R}^{n \times d}$	Node feature matrix
$\mathbf{x}_i \in \mathbb{R}^d$	Feature vector for node i
$\mathcal{N}(i)$	Neighborhood of node i
$\mathbf{A} \in \{0,1\}^{n \times n}$	Adjacency matrix
$d_{\text{graph}}(i, j)$	Graph distance between nodes i and j
$\mathcal{H}$	Homophily ratio of graph
Neural Network Components	
$\mathbf{H}^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$	Hidden representations at layer ℓ
$\mathbf{h}_i^{(\ell)} \in \mathbb{R}^{d_\ell}$	Hidden representation of node i at layer ℓ
L	Total number of layers
$\mathbf{W}_m \in \mathbb{R}^{d_\ell \times d_\ell}$	Message transformation matrix
$\mathbf{W}_a \in \mathbb{R}^{d_\ell \times d_\ell}$	Attention weight matrix
$\mathbf{W}_u \in \mathbb{R}^{d_\ell \times d_\ell}$	Node update matrix
$\sigma(\cdot)$	Sigmoid activation function
$\text{ReLU}(\cdot)$	Rectified linear unit activation
Geometric Components	
$\mathbf{G}_i \in \mathcal{S}_{++}^d$	Metric tensor at node i
$\mathbf{g}_i \in \mathbb{R}_{++}^d$	Diagonal elements of $\mathbf{G}_i$
$g_{i,k} \in \mathbb{R}_{++}$	k-th diagonal element of $\mathbf{G}_i$
ϵ	Lower bound for metric elements
$d_{\mathbf{G}_i}(\cdot, \cdot)$	Geodesic distance under metric $\mathbf{G}_i$
$\mathbf{d}_{ij} \in \mathbb{R}^d$	Normalized direction vector from node i to j
$d_{ij,k} \in \mathbb{R}$	k-th component of $\mathbf{d}_{ij}$
$\tau_{ij} \in \mathbb{R}$	Geometric modulation coefficient
$\alpha_{ij} \in \mathbb{R}$	Geometric attention coefficient
$\text{Ric}_{kk}^{(i)}$	Discrete Ricci curvature at node i, dimension k
Loss Functions and Optimization	
$\mathcal{L}_{\text{task}}$	Task-specific loss function
$\mathcal{L}_{\text{total}}$	Total loss function
$\mathcal{L}_{\text{Ricci}}$	Ricci curvature regularization term
$\mathcal{L}_{\text{smooth}}$	Geometric smoothness regularization term
α	Ricci regularization weight
β	Smoothness regularization weight
θ	Neural network parameters
∇_θ	Gradient with respect to parameters θ
General Mathematical Notation	
$\mathbb{R}$	Real numbers
$\mathbb{R}_{++}$	Positive real numbers
$\mathbb{R}^d$	d-dimensional real vector space
$\mathbb{R}^{m \times n}$	Space of $m \times n$ real matrices
$\mathcal{S}_{++}^d$	Cone of $d \times d$ SPD matrices
$\mathcal{O}(f)$	Big-O notation for asymptotic complexity
$\lVert\cdot\rVert, \lVert\cdot\rVert_2$	Euclidean norm
$\lVert\cdot\rVert_{\mathbf{G}}$	Norm under metric $\mathbf{G}$
$\lVert\cdot\rVert_F$	Frobenius norm
$\langle\cdot, \cdot\rangle$	Standard inner product
$\langle\cdot, \cdot\rangle_{\mathbf{G}}$	Inner product under metric $\mathbf{G}$
$\circ$	Hadamard (element-wise) product
$\succ, \succeq$	Ordering relation
$\text{diag}(\cdot)$	Diagonal matrix constructor

Table 3: Summary of notation throughout this paper. All vectors and matrices are denoted in **bold**.

B Mathematical Foundations

This appendix provides the mathematical foundations underlying our Adaptive Riemannian Graph Neural Networks framework.

B.1 Riemannian Geometry on Discrete Structures

Metric Tensors and Geodesics A Riemannian manifold $(\mathcal{M}, \mathbf{g})$ consists of a smooth manifold $\mathcal{M}$ equipped with a metric tensor $\mathbf{g}$ that varies smoothly across the manifold. For discrete graphs, we discretize this concept by associating each node $i \in \mathcal{V}$ with a metric tensor $\mathbf{G}_i \in \mathcal{S}^d_{++}$, where $\mathcal{S}^d_{++}$ denotes the cone of $d \times d$ symmetric positive definite matrices.

Definition 1 (Discrete Riemannian Graph)**.** *A discrete Riemannian graph is a tuple $(\mathcal{G}, \{\mathbf{G}_i\}_{i\in\mathcal{V}})$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ is an attributed graph and $\{\mathbf{G}_i\}_{i\in\mathcal{V}}$ is a collection of metric tensors such that $\mathbf{G}_i \in \mathcal{S}^d_{++}$ for all $i \in \mathcal{V}$.*

The metric tensor $\mathbf{G}_i$ defines a local inner product on the tangent space at node i:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{G}_i} = \mathbf{u}^T \mathbf{G}_i \mathbf{v} \tag{16}$$

The induced norm and distance are:

$$\|\mathbf{u}\|_{\mathbf{G}_i} = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathbf{G}_i}} = \sqrt{\mathbf{u}^T \mathbf{G}_i \mathbf{u}} \tag{17}$$

$$d_{\mathbf{G}_i}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_{\mathbf{G}_i} = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{G}_i (\mathbf{x} - \mathbf{y})} \tag{18}$$

Geodesics and Parallel Transport In continuous Riemannian geometry, geodesics are curves that locally minimize distance. For our discrete setting, we approximate geodesics using straight lines in the embedding space, with distances measured according to the local metric tensor.

Definition 2 (Discrete Geodesic)**.** *Given two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and a metric tensor $\mathbf{G} \in \mathcal{S}^d_{++}$, the discrete geodesic from $\mathbf{x}$ to $\mathbf{y}$ is the straight line $\gamma(t) = (1-t)\mathbf{x} + t\mathbf{y}$ for $t \in [0, 1]$, with length:*

$$\textit{length}(\gamma) = \int_0^1 \|\dot{\gamma}(t)\|_{\mathbf{G}} dt = \|\mathbf{y} - \mathbf{x}\|_{\mathbf{G}} \tag{19}$$

This approximation is exact when the metric tensor is constant along the path, which holds for our diagonal parameterization in feature space.

Exponential and Logarithmic Maps The exponential map $\exp_p : \mathcal{T}_p\mathcal{M} \to \mathcal{M}$ maps tangent vectors to points on the manifold, while the logarithmic map $\log_p : \mathcal{M} \to \mathcal{T}_p\mathcal{M}$ is its inverse (when well-defined).

For our discrete setting with metric $\mathbf{G}_i$ at node i:

$$\exp_{\mathbf{h}_i}(\mathbf{v}) = \mathbf{h}_i + \mathbf{v} \tag{20}$$

$$\log_{\mathbf{h}_i}(\mathbf{h}_j) = \mathbf{h}_j - \mathbf{h}_i \tag{21}$$

These simplified forms arise from our Euclidean embedding space with varying metrics.

The Line Element and Anisotropic Conformal Metrics A powerful way to understand the geometry defined by a metric tensor is through its *squared line element*, denoted ds^2. This expression defines the infinitesimal squared distance between two nearby points.

In a standard d-dimensional Euclidean space, the metric tensor is the identity matrix $\mathbf{G} = \mathbf{I}$, and its line element is given by the Pythagorean theorem:

$$ds^2 = \sum_{k=1}^{d} (dx^k)^2 \tag{22}$$

where dx^k represents an infinitesimal displacement along the k-th coordinate axis.

A **conformal transformation** of a metric re-scales all distances at a point by the same factor, thus preserving angles (Obata 1970; Spivak 1999). This corresponds to an isotropic scaling of the metric tensor, $\mathbf{G}' = \lambda(x)\mathbf{I}$, where $\lambda(x)$ is a positive scalar function. The line element becomes $ds^2 = \lambda(x) \sum_{k=1}^{d} (dx^k)^2$. Constant-curvature spaces like the Hyperbolic or Spherical space can be modeled as specific types of conformal transformations of Euclidean space.

Our model's diagonal metric, $\mathbf{G}_i = \text{diag}(g_{i,1}, \ldots, g_{i,d})$, defines a more general transformation known as an **anisotropic conformal transformation**. Here, instead of a single scaling factor, we have a unique scaling factor $g_{i,k}$ for each dimension k. The squared line element in the local geometry of node i is thus:

$$ds^2 = \sum_{k=1}^{d} g_{i,k} (dx^k)^2 \tag{23}$$

This is precisely the geometry induced by our metric tensor, as can be seen by considering two nearby points $\mathbf{x}$ and $\mathbf{x} + d\mathbf{x}$:

$$d_{\mathbf{G}_i}(\mathbf{x}, \mathbf{x} + d\mathbf{x})^2 = (d\mathbf{x})^T \mathbf{G}_i (d\mathbf{x}) = \sum_{k=1}^{d} g_{i,k} (dx^k)^2 \tag{24}$$

This geometric structure preserves the orthogonality of the standard coordinate axes but, unlike a true isotropic conformal map, does not preserve angles between arbitrary vectors. This property is key to its flexibility, as it allows the model to learn that certain feature interactions are more or less important for different nodes. This provides a principled geometric foundation for the parameterization chosen in our ARGNN framework.

B.2 Symmetric Positive Definite (SPD) Matrices and Diagonal Parameterization

The space of symmetric positive definite (SPD) matrices $\mathcal{S}_{++}^d$ forms a Riemannian manifold itself and is central to our framework. A matrix $\mathbf{G} \in \mathbb{R}^{d \times d}$ is in $\mathcal{S}_{++}^d$ if it is symmetric ($\mathbf{G} = \mathbf{G}^T$) and for any non-zero vector $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}^T \mathbf{G} \mathbf{x} > 0$.

In our ARGNN framework, we adopt a diagonal parameterization for the metric tensor $\mathbf{G}_i$ at each node i:

$$\mathbf{G}_i = \text{diag}(\mathbf{g}_i) = \text{diag}(g_{i,1}, g_{i,2}, \ldots, g_{i,d}) \tag{25}$$

This simplification has significant geometric and computational implications.

Lemma 1 (Positive Definiteness of Diagonal Metrics)**.** *A diagonal matrix $\mathbf{G} = diag(g_1, \ldots, g_d)$ is symmetric positive definite if and only if all its diagonal elements are strictly positive, i.e., $g_k > 0$ for all $k = 1, \ldots, d$.*

Proof. Symmetry is trivial. For positive definiteness, for any non-zero $\mathbf{x} \in \mathbb{R}^d$:

$$\mathbf{x}^T \mathbf{G} \mathbf{x} = \sum_{k=1}^{d} g_k x_k^2 \tag{26}$$

If all $g_k > 0$, since $\mathbf{x} \neq \mathbf{0}$, at least one $x_k^2 > 0$, so the sum is strictly positive. Conversely, if there exists some $g_j \leq 0$, we can choose $\mathbf{x} = \mathbf{e}_j$ (the j-th standard basis vector) to get $\mathbf{x}^T \mathbf{G} \mathbf{x} = g_j \leq 0$, violating the SPD condition. □

This lemma provides a simple and efficient way to enforce the SPD constraint by ensuring the positivity of the learned vector $\mathbf{g}_i$, which we achieve using the softplus activation function.

Properties of Diagonal Metrics The diagonal structure leads to highly efficient computations for key geometric operations:

- **Inverse:** The inverse is simply the diagonal matrix of reciprocal elements: $\mathbf{G}_i^{-1} = \text{diag}(1/g_{i,1}, \ldots, 1/g_{i,d})$.
- **Determinant:** The determinant is the product of the diagonal elements: $\det(\mathbf{G}_i) = \prod_{k=1}^{d} g_{i,k}$.
- **Eigendecomposition:** The eigendecomposition is trivial. The eigenvalues are the diagonal elements $\{g_{i,k}\}_{k=1}^d$ and the corresponding eigenvectors are the standard basis vectors $\{\mathbf{e}_k\}_{k=1}^d$. This avoids costly $O(d^3)$ computations.

B.3 Discrete Ricci Curvature

Ricci curvature, a fundamental concept in differential geometry, measures how volumes change under parallel transport. We present its discrete analogues suitable for graph neural networks.

Ollivier-Ricci Curvature The Ollivier-Ricci curvature (Ollivier 2009) provides a discrete analogue based on optimal transport theory.

Definition 3 (Ollivier-Ricci Curvature)**.** *For an edge $(i, j) \in \mathcal{E}$, let μ_i and μ_j be probability measures on the neighborhoods of i and j respectively. The Ollivier-Ricci curvature is:*

$$\kappa_{ij} = 1 - \frac{W_1(\mu_i, \mu_j)}{d_{graph}(i, j)} \tag{27}$$

where W_1 is the Wasserstein-1 distance.

For our metric tensor framework, we adapt this to use learned geometric distances:

$$\kappa_{ij}^{\mathbf{G}} = 1 - \frac{W_1^{\mathbf{G}}(\mu_i, \mu_j)}{d_{\mathbf{G}_i}(\mathbf{h}_i, \mathbf{h}_j)} \tag{28}$$

Discrete Ricci Curvature for Diagonal Metrics For diagonal metrics $\mathbf{G}_i = \text{diag}(g_{i,1}, \ldots, g_{i,d})$, we derive a simplified discrete Ricci curvature.

Proposition 3 (Discrete Ricci Curvature for Diagonal Metrics)**.** *For a diagonal metric $\mathbf{G}_i = diag(g_{i,1}, \ldots, g_{i,d})$, the discrete Ricci curvature in the k-th direction is:*

$$Ric_{kk}^{(i)} = -\frac{1}{2|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{g_{j,k} - g_{i,k}}{d_{graph}(i,j)} \tag{29}$$

Proof. The proof follows from discretizing the Ricci curvature tensor formula:

$$\text{Ric}_{kk} = -\frac{1}{2} \sum_j \frac{1}{g_{jj}} \frac{\partial^2 g_{kk}}{\partial x^j \partial x^j} \tag{30}$$

In the discrete setting, we approximate the second partial derivatives using finite differences across the graph:

$$\frac{\partial^2 g_{kk}}{\partial x^j \partial x^j} \approx \frac{1}{|\mathcal{N}(i)|} \sum_{m \in \mathcal{N}(i)} \frac{g_{m,k} - g_{i,k}}{d_{\text{graph}}(i,m)} \tag{31}$$

$$\frac{1}{g_{jj}} \approx \frac{1}{|\mathcal{N}(i)|} \sum_{m \in \mathcal{N}(i)} \frac{1}{g_{m,j}} \tag{32}$$

Substituting and simplifying under the assumption of locally uniform metric variation yields Equation (29). □

Definition 4 (Graph-Geometric Hessian)**.** *The Hessian of the Ricci regularization with respect to metric parameters is:*

$$\mathbf{H}_{Ric} = \nabla_{\mathbf{g}}^2 \mathcal{L}_{Ricci} = \nabla_{\mathbf{g}}^2 \sum_{i \in \mathcal{V}} \sum_{k=1}^{d} \left(Ric_{kk}^{(i)} \right)^2 \tag{33}$$

For our discrete approximation:

$$[\mathbf{H}_{Ric}]_{(i,k),(j,\ell)} = \begin{cases} \frac{1}{|\mathcal{N}(i)|} \sum_{m \in \mathcal{N}(i)} \frac{1}{d_{graph}(i,m)^2} & \text{if } i = j, k = \ell \\ -\frac{1}{|\mathcal{N}(i)| d_{graph}(i,j)^2} & \text{if } j \in \mathcal{N}(i), k = \ell \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

Definition 5 (Normalized Graph Laplacian)**.** *The normalized graph Laplacian used in smoothness regularization is:*

$$\mathbf{L}_G = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2} \tag{35}$$

where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the adjacency matrix.

Ricci Flow Dynamics The Ricci flow equation (Hamilton 1982) evolves a metric according to:

$$\frac{\partial \mathbf{g}_{ij}}{\partial t} = -2\text{Ric}_{ij} \tag{36}$$

For diagonal metrics, this becomes a system of scalar equations:

$$\frac{\partial g_{i,k}}{\partial t} = -2\text{Ric}_{kk}^{(i)} \tag{37}$$

Theorem 3 (Ricci Flow Convergence for Diagonal Metrics)**.** *Under the discrete Ricci flow dynamics in Equation* (37) *with appropriate boundary conditions, the diagonal metric components $g_{i,k}$ converge to a steady state that minimizes the total scalar curvature.*

Proof. Consider the Lyapunov functional:

$$E[\{g_{i,k}\}] = \sum_{i \in \mathcal{V}} \sum_{k=1}^{d} \left(\text{Ric}_{kk}^{(i)} \right)^2 \tag{38}$$

Taking the time derivative along the flow:

$$\frac{dE}{dt} = \sum_{i,k} 2\text{Ric}_{kk}^{(i)} \frac{\partial \text{Ric}_{kk}^{(i)}}{\partial t} \tag{39}$$

$$= \sum_{i,k} 2\text{Ric}_{kk}^{(i)} \frac{\partial \text{Ric}_{kk}^{(i)}}{\partial g_{i,k}} \frac{\partial g_{i,k}}{\partial t} \tag{40}$$

$$= -4 \sum_{i,k} \text{Ric}_{kk}^{(i)} \frac{\partial \text{Ric}_{kk}^{(i)}}{\partial g_{i,k}} \text{Ric}_{kk}^{(i)} \tag{41}$$

$$= -4 \sum_{i,k} \left(\text{Ric}_{kk}^{(i)}\right)^2 \frac{\partial \text{Ric}_{kk}^{(i)}}{\partial g_{i,k}} \tag{42}$$

For our discrete approximation, $\frac{\partial \text{Ric}_{kk}^{(i)}}{\partial g_{i,k}} > 0$, ensuring $\frac{dE}{dt} \leq 0$. The flow converges to a critical point where $\text{Ric}_{kk}^{(i)} = 0$ for all i, k. □

B.4 Connection to Optimal Transport

Our geometric message passing framework has deep connections to optimal transport theory (Villani et al. 2008), providing theoretical foundations for the learned metrics.

Wasserstein Distance and Graph Geometry The Wasserstein distance between probability measures μ and ν on a metric space $(\mathcal{X}, d)$ is:

$$W_1(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} d(x, y)\, d\pi(x, y) \tag{43}$$

For our graph setting, we define node-wise measures $\mu_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} \delta_{\mathbf{h}_j}$ and compute transport costs using learned metrics:

$$c_{ij} = d_{\mathbf{G}_i}(\mathbf{h}_i, \mathbf{h}_j) = \sqrt{(\mathbf{h}_i - \mathbf{h}_j)^T \mathbf{G}_i (\mathbf{h}_i - \mathbf{h}_j)} \tag{44}$$

Optimal Transport Interpretation of Message Passing Our message passing can be interpreted as approximating optimal transport plans:

$$\mathbf{m}_{ij} = \arg\min_{\mathbf{m}} \int c(\mathbf{h}_i, \mathbf{h}_j)\, d\pi(\mathbf{h}_i, \mathbf{h}_j) + \lambda \|\mathbf{m} - \mathbf{W}_m \mathbf{h}_j\|^2 \tag{45}$$

This connection provides theoretical justification for the geometric modulation terms in our framework.

C Complete Proofs of Theoretical Results

This appendix provides detailed proofs for all theoretical results in Section 5. First, we combine traditional optimization analysis with geometric insights from Riemannian manifold theory to establish convergence guarantees for ARGNN. Then we give the proof for our proposed universal approximation framework of ARGNN. The extended results, like Lyapunov stability analysis, generalization bounds, robustness, and complexity proof, are also provided in this section.

C.1 Proof of Convergence of Adaptive Geometry Learning

Preliminaries and Assumptions

Notation and Geometric Setup We work with diagonal metric tensors $\mathbf{G}_i = \text{diag}(\mathbf{g}_i)$ where $\mathbf{g}_i \in \mathbb{R}^d_{++}$. While each $\mathbf{G}_i$ can be viewed as an element of the symmetric positive-definite manifold $\text{SPD}(d)$, our diagonal constraint restricts us to a submanifold:

$$\mathcal{M}_{\text{diag}} = \{\mathbf{G} \in \text{SPD}(d) : \mathbf{G} = \text{diag}(\mathbf{g}), \mathbf{g} \in \mathbb{R}^d_{++}\} \cong \mathbb{R}^d_{++} \tag{46}$$

The total parameter space is:

$$\mathcal{M} = \mathcal{M}_{\text{diag}}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times d} \tag{47}$$

where the first component contains all metric parameters $\{\mathbf{g}_i\}_{i\in\mathcal{V}}$ and the second contains node features $\{\mathbf{h}_i\}_{i\in\mathcal{V}}$.

Regularity Assumptions

Assumption 1 (Regularity Conditions)**.** *(11)* ***Bounded Features****: Node representations satisfy $\|\mathbf{h}_i^{(\ell)}\|_2 \leq B$ for all nodes $i \in \mathcal{V}$, layers $\ell \in [L]$, and some constant $B > 0$.*

(22) ***Lipschitz Task Loss****: For classification tasks with cross-entropy loss, the gradient is L_f-Lipschitz continuous with $L_f = O(1)$ due to bounded softmax outputs.*

(33) ***Bounded Metrics****: The diagonal metric components satisfy $\epsilon \leq g_{i,k} \leq M$ for constants $0 < \epsilon < M < \infty$.*

(44) ***Connected Graph****: The graph $\mathcal{G}$ is connected with finite diameter $diam(\mathcal{G})$ and algebraic connectivity $\lambda_2(\mathbf{L}_G) > 0$.*

Task Loss Structure For node classification with C classes:

$$\mathcal{L}_{\text{node}} = -\sum_{i \in \mathcal{V}_{\text{train}}} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c} \tag{48}$$

For edge prediction (binary classification):

$$\mathcal{L}_{\text{edge}} = -\sum_{(i,j) \in \mathcal{E}_{\text{train}}} [y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij})] \tag{49}$$

Both losses have similar smoothness properties with bounded gradients due to the sigmoid/softmax outputs.

Homophily and Geometric Complexity

Definition 6 (Graph Homophily)**.** *The homophily ratio of a graph is:*

$$\mathcal{H} = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \mathbf{1}[y_i = y_j] \tag{50}$$

where y_i denotes the label of node i.

Lemma 2 (Homophily-Dependent Gradient Structure)**.** *For classification tasks on graphs with homophily $\mathcal{H}$, the gradient variance across edges scales as:*

$$Var_{(i,j) \in \mathcal{E}} \left[\langle \nabla_{\mathbf{h}_i} \mathcal{L}_{task}, \nabla_{\mathbf{h}_j} \mathcal{L}_{task} \rangle \right] = O((1 - \mathcal{H})^2) \tag{51}$$

Proof. Consider the gradient inner product for neighboring nodes:

- For homophilic edges where $y_i = y_j$: The gradients align positively as both nodes push predictions toward the same class, giving $\langle \nabla_i, \nabla_j \rangle > 0$.
- For heterophilic edges where $y_i \neq y_j$: The gradients oppose as nodes push toward different classes, giving $\langle \nabla_i, \nabla_j \rangle < 0$.

The variance decomposes as:

$$\text{Var}[\langle \nabla_i, \nabla_j \rangle] = \mathcal{H} \cdot \text{Var}_{\text{homo}}[\langle \nabla_i, \nabla_j \rangle] + (1 - \mathcal{H}) \cdot \text{Var}_{\text{hetero}}[\langle \nabla_i, \nabla_j \rangle] \tag{52}$$

$$\approx \mathcal{H} \cdot 0 + (1 - \mathcal{H}) \cdot O(1) = O(1 - \mathcal{H}) \tag{53}$$

The squared term $(1 - \mathcal{H})^2$ arises when considering gradient products in the Hessian computation. □

Proposition 4 (Homophily and Problem Conditioning)**.** *The condition number of the task loss Hessian scales inversely with homophily:*

$$\kappa(\nabla^2 \mathcal{L}_{task}) = \frac{\lambda_{\max}(\nabla^2 \mathcal{L}_{task})}{\lambda_{\min}(\nabla^2 \mathcal{L}_{task})} = O\left(\frac{1}{\mathcal{H}}\right) \tag{54}$$

Proof. The Hessian of the classification loss can be decomposed into contributions from node self-loops and edge interactions:

$$\nabla^2 \mathcal{L}_{\text{task}} = \sum_{i \in \mathcal{V}} \mathbf{H}_i^{\text{self}} + \sum_{(i,j) \in \mathcal{E}} \mathbf{H}_{ij}^{\text{edge}} \tag{55}$$

For homophilic graphs:

- Self-contributions $\mathbf{H}_i^{\text{self}} \succeq 0$ provide baseline positive curvature
- Edge contributions $\mathbf{H}_{ij}^{\text{edge}} \succeq 0$ for homophilic edges reinforce this curvature
- This creates strong positive definiteness: $\lambda_{\min} = \Omega(\mathcal{H})$

For heterophilic graphs:

- Edge contributions $\mathbf{H}_{ij}^{\text{edge}} \preceq 0$ for heterophilic edges oppose self-contributions
- Near-cancellation leads to $\lambda_{\min} \to 0^+$ as $\mathcal{H} \to 0$
- Maximum eigenvalue remains $\lambda_{\max} = O(1)$ from self-contributions

Therefore, $\kappa = \lambda_{\max}/\lambda_{\min} = O(1/\mathcal{H})$. □

Main Convergence Proof

Proof of Theorem 1. We analyze the stochastic gradient descent dynamics for the total loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \alpha\mathcal{L}_{\text{Ricci}} + \beta\mathcal{L}_{\text{smooth}} \tag{56}$$

Step 1: Gradient Bounds.

Lemma 3 (Total Gradient Bound)**.** *Under Assumption 1, the gradient norm satisfies:*

$$\|\nabla_{\mathbf{g}}\mathcal{L}_{total}\|_2 \leq C_1 L + C_2\alpha|\mathcal{V}| + C_3\beta|\mathcal{E}| \tag{57}$$

where C_1, C_2, C_3 are constants depending on B, L_f, M.

Proof of Lemma 3. The gradient decomposes as:

$$\nabla_{\mathbf{g}}\mathcal{L}_{\text{total}} = \nabla_{\mathbf{g}}\mathcal{L}_{\text{task}} + \alpha\nabla_{\mathbf{g}}\mathcal{L}_{\text{Ricci}} + \beta\nabla_{\mathbf{g}}\mathcal{L}_{\text{smooth}} \tag{58}$$

(i) **Task gradient**: Through L layers of backpropagation:

$$\left\|\frac{\partial\mathcal{L}_{\text{task}}}{\partial g_{i,k}}\right\| \leq L_f \cdot B \cdot L \cdot \rho^{L-1} \tag{59}$$

where $\rho < 1$ is the contraction factor from activation functions.

(ii) **Ricci gradient**: From Equation (8):

$$\frac{\partial\mathcal{L}_{\text{Ricci}}}{\partial g_{i,k}} = 2\text{Ric}_{kk}^{(i)} \cdot \left(-\frac{1}{2|\mathcal{N}(i)|}\right) = -\frac{\text{Ric}_{kk}^{(i)}}{|\mathcal{N}(i)|} \tag{60}$$

(iii) **Smoothness gradient**:

$$\frac{\partial\mathcal{L}_{\text{smooth}}}{\partial g_{i,k}} = 2\sum_{j\in\mathcal{N}(i)}(g_{i,k} - g_{j,k}) \tag{61}$$

Combining with boundedness assumptions gives the stated bound. □

Step 2: Effective Curvature with Homophily.

The key insight is that regularization creates an effective curvature that depends on homophily:

Lemma 4 (Effective Curvature)**.** *The regularized loss has effective curvature:*

$$\mu_{eff} = \alpha\mu_{Ricci} + \beta\mu_{smooth} + \mathcal{H}\mu_{task} \tag{62}$$

where:

$$\mu_{Ricci} = \frac{1}{|\mathcal{N}_{\max}| \cdot diam(\mathcal{G})^2} \tag{63}$$

$$\mu_{smooth} = \lambda_2(\mathbf{L}_G) \tag{64}$$

$$\mu_{task} = \Omega(\mathcal{H}) \tag{65}$$

Step 3: Optimal Hyperparameter Derivation.

To maximize convergence rate while maintaining stability, we solve:

$$\max_{\alpha,\beta} \frac{\mu_{\text{eff}}}{\sqrt{\kappa_{\text{total}}}} \quad \text{subject to} \quad \alpha L_{\text{Ricci}} + \beta L_{\text{smooth}} \leq \frac{2}{\eta_{\max} L} \tag{66}$$

where κ_{total} is the total condition number.

Proposition 1 (Restate: Homophily-Aware Optimization)**.** *The optimal regularization weights that balance conditioning and convergence are:*

$$\alpha^* = \frac{c_1\mathcal{H}}{L} \cdot \min\left(1, \frac{d}{|\mathcal{E}|}\right) \tag{67}$$

$$\beta^* = \frac{c_2\mathcal{H}\sqrt{d}}{|\mathcal{V}|} \tag{68}$$

where $c_1, c_2 > 0$ are dataset-dependent constants.

Proof of Proposition 1. The optimization problem has first-order conditions:

$$\frac{\partial}{\partial \alpha}\left[\frac{\mu_{\text{eff}}}{\sqrt{\kappa_{\text{total}}}}\right] = 0 \tag{69}$$

$$\frac{\partial}{\partial \beta}\left[\frac{\mu_{\text{eff}}}{\sqrt{\kappa_{\text{total}}}}\right] = 0 \tag{70}$$

Key observations:

1. **Well-conditioned problems** (high $\mathcal{H}$) can tolerate stronger regularization without hurting convergence
2. **Ill-conditioned problems** (low $\mathcal{H}$) require careful balance to maintain expressiveness
3. The factor $1/L$ accounts for gradient signal decay through layers
4. The $\min(1, d/|\mathcal{E}|)$ prevents over-regularization on sparse graphs
5. The $\sqrt{d}$ scaling maintains proper high-dimensional normalization

Solving the constrained optimization with Lagrange multipliers yields the stated result. □

Step 4: Convergence Rate.
Combining the effective curvature with standard SGD analysis:

$$\mathbb{E}[\mathcal{L}_{\text{total}}^{(t+1)}] \leq \mathcal{L}_{\text{total}}^{(t)} - \eta_t \mu_{\text{eff}}(\mathcal{L}_{\text{total}}^{(t)} - \mathcal{L}_{\text{total}}^{*}) + \frac{L_{\text{total}}\eta_t^2}{2}\sigma^2 \tag{71}$$

where σ^2 is the gradient variance bound.
With learning rate $\eta_t = \eta_0/\sqrt{t}$ and optimal hyperparameters:

$$\mu_{\text{eff}}^{*} = \Theta\left(\frac{\mathcal{H}^2}{L}\right) \tag{72}$$

This gives the convergence rate:

$$\mathbb{E}[\|\nabla \mathcal{L}_{\text{total}}^{(t)}\|^2] = O\left(\frac{1}{\sqrt{t}} \cdot \exp\left(-\frac{\mu_{\text{eff}}^{*} t}{L}\right)\right) \tag{73}$$

The additional factor of $\mathcal{H}$ in μ_{eff}^{*} arises from the product $\alpha^{*} \cdot \mu_{\text{task}}$ where both terms scale with $\mathcal{H}$. □

Analysis of Homophily-Aware Constants

Analysis of Proposition 1. The constants c_1, c_2 capture the interplay between graph structure and optimization dynamics.
Derivation of c_1:
The Ricci regularization weight must balance two competing factors:

1. **Geometric flexibility**: Heterophilic graphs need complex geometries
2. **Optimization stability**: All graphs benefit from some regularization

Through extensive empirical validation across diverse graphs, we find:

$$c_1 = (1 - \mathcal{H}) + 0.1 \tag{74}$$

This ensures:

- Heterophilic graphs ($\mathcal{H} \to 0$): $c_1 \to 1.1$ allows complex geometry
- Homophilic graphs ($\mathcal{H} \to 1$): $c_1 \to 0.1$ maintains minimal regularization
- The baseline 0.1 prevents numerical instability

Combined with $\alpha^{*} \propto \mathcal{H}/L$, the effective Ricci weight becomes:

$$\alpha^{*} = \frac{[(1 - \mathcal{H}) + 0.1] \cdot \mathcal{H}}{L} \cdot \min\left(1, \frac{d}{|\mathcal{E}|}\right) \tag{75}$$

This creates a balanced scaling that peaks at moderate homophily ($\mathcal{H} \approx 0.5$).
Derivation of c_2:
The smoothness regularization should be stronger for homophilic graphs:

$$c_2 = 0.1 \cdot (1 + \mathcal{H}) \tag{76}$$

This gives:

- Heterophilic: $c_2 \to 0.1$ (minimal smoothness constraint)
- Homophilic: $c_2 \to 0.2$ (stronger smoothness for similar neighbors)

The factor 0.1 is calibrated to typical gradient magnitudes in neural networks. □

C.2 Proof of Universal Geometric Framework (Theorem 2)

Proof. We prove that ARGNN generalizes and approximates fixed-curvature GNNs by showing they correspond to constrained metric configurations.

Step 1: Metric Space Hierarchy

Define the space of all positive diagonal metrics:

$$\mathcal{M}_{\text{full}} = \left\{\mathbf{g} \in \mathbb{R}^d : g_k > 0 \, \forall k \in [d]\right\} = \mathbb{R}^d_{++} \tag{77}$$

Fixed-curvature GNNs operate in constrained subspaces:

1. **Euclidean GNNs**:
$$\mathcal{M}_{\text{Euclidean}} = \{\mathbf{1}\} \subset \mathcal{M}_{\text{full}}, \quad \dim = 0 \tag{78}$$
2. **Hyperbolic GNNs** (constant negative curvature):
$$\mathcal{M}_{\text{Hyperbolic}} = \{c_h \mathbf{1} : 0 < c_h < 1\} \subset \mathcal{M}_{\text{full}}, \quad \dim = 1 \tag{79}$$
3. **Spherical GNNs** (constant positive curvature):
$$\mathcal{M}_{\text{Spherical}} = \{c_s \mathbf{1} : c_s > 1\} \subset \mathcal{M}_{\text{full}}, \quad \dim = 1 \tag{80}$$
4. **Product Manifolds** (e.g., $\mathbb{H}^{d_1} \times \mathbb{S}^{d_2} \times \mathbb{E}^{d_3}$): This induces a block-constant diagonal metric:
$$\mathbf{g}_i = (\underbrace{c_{\mathbb{H}}, \ldots, c_{\mathbb{H}}}_{d_1}, \underbrace{c_{\mathbb{S}}, \ldots, c_{\mathbb{S}}}_{d_2}, \underbrace{1, \ldots, 1}_{d_3})^T \tag{81}$$
$$\mathcal{M}_{\text{Product}} = \left\{\mathbf{g} = (c_h \mathbf{1}_{d_1}, c_s \mathbf{1}_{d_2}, \mathbf{1}_{d_3})^T\right\} \subset \mathcal{M}_{\text{full}}, \quad \dim = 2 \tag{82}$$

where $\mathbf{1}_{d_i}$ denotes a vector of ones of dimension d_i. Dive into the general case,

$$\mathcal{M}_{\text{Product}} = \left\{(c_1 \mathbf{1}_{d_1}, \ldots, c_p \mathbf{1}_{d_p})^\top : c_\ell > 0, \sum_{\ell=1}^{p} d_\ell = d\right\}, \tag{83}$$

$\dim = p$ with $p \leq d$.

Step 2: Embedding Fixed Geometries

We show each fixed-curvature GNN has an equivalent ARGNN configuration:

Lemma 5 (Geometric Equivalence)**.** *For any GNN operating on a Riemannian manifold with metric tensor $\mathbf{G}_{fixed}$, there exists a diagonal metric $\mathbf{G}_{diag} = diag(\mathbf{g})$ such that the induced geodesic distances are proportional:*

$$d_{\mathbf{G}_{diag}}(\mathbf{x}, \mathbf{y}) = \kappa \cdot d_{\mathbf{G}_{fixed}}(\mathbf{x}, \mathbf{y}) \tag{84}$$

for some constant $\kappa > 0$.

Proof of Lemma. For diagonal metrics in local coordinates: - Hyperbolic space $\mathbb{H}^d$: Use Poincaré ball coordinates with $\mathbf{g} = \frac{4}{(1-\|\mathbf{x}\|^2)^2}\mathbf{1}$ - Sphere $\mathbb{S}^d$: Use stereographic projection with $\mathbf{g} = \frac{4}{(1+\|\mathbf{x}\|^2)^2}\mathbf{1}$ - Euclidean $\mathbb{E}^d$: Trivially $\mathbf{g} = \mathbf{1}$

In our framework, we approximate these locally with constant diagonal metrics. □

Step 3: Strict Generalization

The constraint hierarchy is:

$$\{\text{point}\} \subset \{\text{line}\} \subset \{\text{plane}\} \subset \cdots \subset \mathbb{R}^d_{++} \tag{85}$$

Specifically:

$$\dim(\mathcal{M}_{\text{Euclidean}}) = 0 < \dim(\mathcal{M}_{\text{Hyperbolic/Spherical}}) = 1 \tag{86}$$
$$< \dim(\mathcal{M}_{\text{Product}}) \leq d - 1 < \dim(\mathcal{M}_{\text{full}}) = d \tag{87}$$

Step 4: Universal Approximation Property. Since our metric-learning network $f_\theta^{(g)} : \mathbb{R}^d \to \mathbb{R}^d_{++}$ is parameterized by a neural network with sufficient capacity, by the universal approximation theorem, it can approximate any continuous function mapping node features to metric parameters. For any target geometric configuration from existing methods with metric $\mathbf{G}_i^{\text{target}}$, we can find parameters θ such that:

$$\|\text{diag}(f_\theta^{(g)}(\mathbf{h}_i)) - \mathbf{G}_i^{\text{target}}\|_F < \epsilon \tag{88}$$

for any $\epsilon > 0$ and all nodes $i \in \mathcal{V}$. This completes the proof. □

C.3 Computational Complexity Analysis

Proposition 2 (Restate: Computational and Memory Complexity)**.** *For a graph with $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges, hidden dimension d, and L layers, ARGNN has:*

1. ***Time Complexity**: $O(L \cdot (m + n) \cdot d^2)$ per forward pass*
2. ***Space Complexity**: $O(n \cdot d + m)$ for storing features and graph structure*
3. ***Parameter Count**: $O(L \cdot d^2)$ for shared weights plus $O(n \cdot d)$ for learnable metrics*

Proof. We provide a detailed complexity analysis for each component of ARGNN.

Per-Layer Time Complexity:

(i) **Metric Learning Network $f_\theta^{(g)}$:**
- Input: concatenated features $[\mathbf{h}_i; \mathbf{a}_i] \in \mathbb{R}^{2d}$ for each node
- Network architecture: $2d \to h \to d$ where h is hidden dimension
- Operations per node: $O(2d \cdot h + h \cdot d) = O(d^2)$ assuming $h = O(d)$
- Total for all nodes: $O(n \cdot d^2)$

(ii) **Geometric Modulation Computation:**
- Direction vector: $\mathbf{d}_{ij} = (\mathbf{h}_j - \mathbf{h}_i)/\|\mathbf{h}_j - \mathbf{h}_i\|_2$ costs $O(d)$ per edge
- Modulation: $\tau_{ij} = \sum_{k=1}^{d} d_{ij,k}^2 \cdot \tanh(-\log g_{i,k})$ costs $O(d)$ per edge
- Total for all edges: $O(m \cdot d)$

(iii) **Geometric Attention:**
- Inner product: $\sum_k g_{i,k} h_{i,k} h_{j,k}$ costs $O(d)$ per edge
- Normalization requires pre-computed node norms: $O(n \cdot d)$ once
- Total: $O(m \cdot d + n \cdot d) = O((m + n) \cdot d)$

(iv) **Message Passing:**
- Message transformation: $\mathbf{W}_m \mathbf{h}_j$ for each edge costs $O(m \cdot d^2)$
- Self transformation: $\mathbf{W}_s \mathbf{h}_i$ for each node costs $O(n \cdot d^2)$
- Aggregation: summing messages costs $O(m \cdot d)$
- Total: $O((m + n) \cdot d^2)$

Total Per-Layer Complexity:

$$O(n \cdot d^2) + O(m \cdot d) + O((m + n) \cdot d) + O((m + n) \cdot d^2) = O((m + n) \cdot d^2) \tag{89}$$

Since $m \geq n - 1$ for connected graphs, this simplifies to $O(m \cdot d^2)$ in typical notation.

Full Model Complexity:

- Time: $O(L \cdot m \cdot d^2)$ for L layers
- Space: $O(n \cdot d)$ for features + $O(m)$ for edges + $O(n \cdot d)$ for metrics
- Parameters: $O(L \cdot d^2)$ for weight matrices + $O(d^2)$ for metric network + $O(n \cdot d)$ for metric values

Comparison with Traditional GNN Layers:

Comparison with Traditional GNN Layers

Method	Time/Layer	Space	Parameters
GCN	$O(md + nd^2)$	$O(nd + m)$	$O(Ld^2)$
GAT	$O(md^2 + nd^2)$	$O(nd + m)$	$O(Ld^2)$
ARGNN	$O(md^2 + nd^2)$	$O(nd + m)$	$O(Ld^2 + nd)$

ARGNN adds only $O(n \cdot d)$ learnable metric parameters while maintaining the same asymptotic time complexity as GAT. The geometric computations (τ_{ij}, α_{ij}) are $O(d)$ per edge, which is absorbed by the dominant $O(d^2)$ transformation costs. □

Remark 1 (Practical Optimizations)**.** *In practice, several optimizations reduce the computational burden:*

1. ***Sparse Operations**: For sparse graphs with average degree $\bar{d} \ll n$, the effective complexity is $O(L \cdot n \cdot \bar{d} \cdot d^2)$*
2. ***Mini-batching**: Node-wise operations can be parallelized efficiently on GPUs*
3. ***Metric Sharing**: For very large graphs, metrics can be shared across similar nodes, reducing parameters from $O(n \cdot d)$ to $O(K \cdot d)$ where $K \ll n$ is the number of clusters*

C.4 Additional Results

Corollary 1 (Expressiveness on Mixed-Topology Graphs)**.** *For graphs containing both tree-like and clique-like substructures, ARGNN achieves strictly lower loss than any fixed-curvature GNN:*

$$\inf_{\mathbf{g}_i \in \mathbb{R}^d_{++}} \mathcal{L}_{\textit{ARGNN}} < \inf_{c>0} \mathcal{L}_{\textit{fixed}}(c) \tag{90}$$

Proof. Consider a graph $\mathcal{G} = \mathcal{G}_{\text{tree}} \cup \mathcal{G}_{\text{clique}}$. Optimal embeddings require:

- $\mathcal{G}_{\text{tree}}$: Small metric values ($g_{i,k} < 1$) for exponential volume growth
- $\mathcal{G}_{\text{clique}}$: Large metric values ($g_{i,k} > 1$) for bounded geometry

No single constant c can optimize both simultaneously, while ARGNN can assign different metrics to different regions. □

Proposition 5 (Stability Under Regularization)**.** *The learned metrics evolve smoothly during training:*

$$\|\mathbf{g}_i^{(t+1)} - \mathbf{g}_i^{(t)}\|_2 \leq \frac{C}{\alpha + \beta} \cdot \eta_t \tag{91}$$

Proof. The regularization terms create a locally strongly convex penalty:

$$\lambda_{\min}(\nabla^2[\alpha\mathcal{L}_{\text{Ricci}} + \beta\mathcal{L}_{\text{smooth}}]) \geq \min\{\alpha\mu_{\text{Ricci}}, \beta\mu_{\text{smooth}}\} > 0 \tag{92}$$

This bounds the per-iteration metric updates, ensuring smooth evolution. □

Lyapunov Stability Analysis We now establish a stronger stability guarantee using Lyapunov theory, complementing our convergence analysis.

Theorem 4 (Lyapunov Stability of Geometry Evolution)**.** *Under Assumptions (A1)-(A4) and appropriate step size $\eta_t \leq \frac{\rho_{\min}}{4L_{\textit{total}}}$ where $\rho_{\min} = \min\{\alpha\lambda_{\min}(\mathbf{H}_{\textit{Ricci}}), \beta\lambda_2(\mathbf{L}_G)\}$, there exists a Lyapunov function:*

$$\mathcal{V}_t = \mathcal{L}_{\textit{total}}^{(t)} - \mathcal{L}_{\textit{total}}^{*} + \gamma \sum_{i \in \mathcal{V}} \|\mathbf{g}_i^{(t)} - \mathbf{g}_i^{*}\|_2^2 \tag{93}$$

with $0 < \gamma \leq \frac{1}{4L_{\textit{total}}}$, such that:

$$\mathbb{E}[\mathcal{V}_{t+1}] \leq (1 - \rho_{\min}\eta_t)\mathbb{E}[\mathcal{V}_t] \tag{94}$$

Proof. The proof proceeds by analyzing the descent properties of both loss and metric components.

Step 1: Loss Descent. From standard smoothness arguments:

$$\mathcal{L}_{\text{total}}^{(t+1)} \leq \mathcal{L}_{\text{total}}^{(t)} - \eta_t \|\nabla\mathcal{L}_{\text{total}}^{(t)}\|^2 + \frac{L_{\text{total}}\eta_t^2}{2}\|\mathbf{v}^{(t)}\|^2 \tag{95}$$

where $\mathbf{v}^{(t)}$ is the stochastic gradient.

Step 2: Metric Stability. The regularization terms induce strong convexity in the metric space:

$$\langle \nabla^2[\alpha\mathcal{L}_{\text{Ricci}} + \beta\mathcal{L}_{\text{smooth}}](\mathbf{g} - \mathbf{g}^*), \mathbf{g} - \mathbf{g}^* \rangle \geq \rho_{\min}\|\mathbf{g} - \mathbf{g}^*\|^2 \tag{96}$$

This gives:

$$\|\mathbf{g}^{(t+1)} - \mathbf{g}^*\|^2 \leq (1 - \eta_t\rho_{\min})\|\mathbf{g}^{(t)} - \mathbf{g}^*\|^2 + O(\eta_t^2) \tag{97}$$

Step 3: Lyapunov Descent. Combining both components with appropriate weighting γ:

$$\mathcal{V}_{t+1} - \mathcal{V}_t \leq -\eta_t\|\nabla\mathcal{L}_{\text{total}}^{(t)}\|^2 + \frac{L_{\text{total}}\eta_t^2}{2}\|\mathbf{v}^{(t)}\|^2 \tag{98}$$

$$- \gamma\eta_t\rho_{\min}\sum_i \|\mathbf{g}_i^{(t)} - \mathbf{g}_i^*\|^2 + O(\gamma\eta_t^2) \tag{99}$$

Using the fact that at optimality $\nabla\mathcal{L}_{\text{total}}^* = 0$ and choosing η_t appropriately:

$$\mathbb{E}[\mathcal{V}_{t+1}] \leq (1 - \rho_{\min}\eta_t)\mathcal{V}_t \tag{100}$$

Iterating gives exponential decay: $\mathbb{E}[\mathcal{V}_t] \leq \mathcal{V}_0 \exp(-\rho_{\min}\sum_{s=0}^{t-1}\eta_s)$. □

Generalization Bounds

Theorem 5 (Generalization of Learned Geometries)**.** *Let $\hat{\mathbf{g}}_i$ be the learned metric parameters on a training set of size n_{train}. With probability at least $1-\delta$, the generalization gap satisfies:*

$$\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{test}] - \mathcal{L}_{train} \leq O\left(\sqrt{\frac{d\log(n/\delta)}{n_{train}}} + \alpha\sqrt{|\mathcal{V}|} + \beta\sqrt{|\mathcal{E}|}\right) \tag{101}$$

Proof. We use Rademacher complexity analysis adapted to our geometric setting.

Step 1: Function Class Complexity. The hypothesis class of ARGNN with bounded metrics is:

$$\mathcal{F} = \{f_{\mathbf{g}} : \epsilon \leq g_{i,k} \leq M, \forall i, k\} \tag{102}$$

The Rademacher complexity of this class is:

$$\mathcal{R}_n(\mathcal{F}) \leq \frac{2M}{\epsilon} \cdot \sqrt{\frac{d}{n_{\text{train}}}} \tag{103}$$

Step 2: Regularization Effect. The regularization terms effectively reduce the hypothesis space complexity:

- Ricci regularization constrains metric variation: contributes $O(\alpha\sqrt{|\mathcal{V}|})$
- Smoothness regularization constrains neighbor differences: contributes $O(\beta\sqrt{|\mathcal{E}|})$

Step 3: Final Bound. Applying standard generalization theory with our specific constraints gives the stated bound. □

Analysis of Geometric Message Passing We analyze the behavior of our geometric message passing under different metric configurations.

Proposition 6 (Message Modulation Properties)**.** *For the geometric modulation coefficient $\tau_{ij} = \sum_{k=1}^{d} d_{ij,k}^2 \cdot \tanh(-\log g_{i,k})$:*

1. *For isotropic metrics $\mathbf{g}_i = c\mathbf{1}$: $\tau_{ij} = \tanh(-\log c)$ (direction-independent)*
2. *$\tau_{ij} \in [-1, 1]$ with $\tau_{ij} = 0$ when $g_{i,k} = 1$ for all k*
3. *Anisotropic metrics enable direction-dependent message weighting*

Proof. For isotropic case with $g_{i,k} = c$ for all k:

$$\tau_{ij} = \tanh(-\log c) \sum_{k=1}^{d} d_{ij,k}^2 = \tanh(-\log c) \tag{104}$$

since $\sum_k d_{ij,k}^2 = 1$ by normalization.

The bounds follow from $\tanh(\cdot) \in (-1, 1)$ and $\tanh(0) = 0$.

For anisotropic metrics, different $g_{i,k}$ values weight directional components differently, enabling richer message modulation. □

Remark 2 (Euclidean Singularity)**.** *Our formulation has $\tau_{ij} = 0$ for Euclidean metrics ($c = 1$), which can block information flow. In practice, we can add a small baseline: $\tau_{ij}^{modified} = \epsilon_0 + (1 - \epsilon_0)\tau_{ij}$ with $\epsilon_0 > 0$.*

Robustness Analysis

Theorem 6 (Robustness to Graph Perturbations)**.** *Let $\mathcal{G}'$ be a perturbed version of $\mathcal{G}$ with at most k edge additions/deletions. The learned metrics satisfy:*

$$\|\mathbf{g}_i' - \mathbf{g}_i\|_2 \leq O\left(\frac{k}{\beta|\mathcal{N}(i)|}\right) \tag{105}$$

where $\mathbf{g}_i'$ are metrics learned on $\mathcal{G}'$.

Proof. The smoothness regularization creates a coupling between neighboring metrics. Edge perturbations affect the regularization gradient:

$$\Delta\left(\frac{\partial \mathcal{L}_{\text{smooth}}}{\partial g_{i,k}}\right) = 2 \sum_{j \in \Delta\mathcal{N}(i)} (g_{i,k} - g_{j,k}) \tag{106}$$

where $\Delta\mathcal{N}(i)$ represents changed neighbors. The strong convexity from regularization bounds the metric change, giving the stated robustness guarantee. □

D Algorithm Details

This section provides comprehensive implementation details for ARGNN, including forward and backward pass algorithms, optimization procedures.

D.1 Complete Forward Pass Algorithm

Algorithm 1 presents the complete forward pass with all implementation details in pseudo code.

Algorithm 1: ARGNN Forward Pass

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, layers L, hidden dimensions $\{d_\ell\}_{\ell=1}^{L}$
Ensure: Final representations $\mathbf{H}^{(L)}$

Initialize $\mathbf{H}^{(0)} = \mathbf{X}$
Initialize metric networks $\{f_\theta^{(g,\ell)}\}_{\ell=1}^{L}$
Initialize transformation matrices $\{\mathbf{W}_s^{(\ell)}, \mathbf{W}_m^{(\ell)}\}_{\ell=1}^{L}$
Initialize regularization parameters α, β, ϵ
for $\ell = 1$ to L **do**
  // Step 1: Diagonal Metric Tensor Learning
  for all nodes $i \in \mathcal{V}$ in parallel **do**
    $\mathbf{a}_i^{(\ell-1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(\ell-1)}$ {Aggregate neighbors}
    $\mathbf{z}_i^{(\ell)} = [\mathbf{h}_i^{(\ell-1)}; \mathbf{a}_i^{(\ell-1)}]$ {Concatenate features}
    $\mathbf{g}_i^{(\ell)} = \text{softplus}(f_\theta^{(g,\ell)}(\mathbf{z}_i^{(\ell)})) + \epsilon$ {Learn diagonal metric vector}
  end for
  // Step 2: Geometric Message Passing
  $\mathbf{M}^{(\ell)} = \{\}$ {Initialize message collection for aggregation}
  for all edges $(i, j) \in \mathcal{E}$ in parallel **do**
    $\mathbf{u}_{ij} = \mathbf{h}_j^{(\ell-1)} - \mathbf{h}_i^{(\ell-1)}$ {Direction vector}
    $\mathbf{d}_{ij} = \mathbf{u}_{ij} / (\|\mathbf{u}_{ij}\|_2 + \epsilon)$ {Normalized direction}
    // Geometric modulation coefficient
    $\tau_{ij} = \sum_{k=1}^{d_{\ell-1}} (d_{ij,k})^2 \cdot \tanh(-\log g_{i,k}^{(\ell)})$
    // Geometric attention coefficient
    $\text{num}_{ij} = \sum_{k=1}^{d_{\ell-1}} g_{i,k}^{(\ell)} \cdot h_{i,k}^{(\ell-1)} \cdot h_{j,k}^{(\ell-1)}$
    $\text{den}_i = \sqrt{\sum_{k=1}^{d_{\ell-1}} g_{i,k}^{(\ell)} \cdot (h_{i,k}^{(\ell-1)})^2}$
    $\text{den}_j = \sqrt{\sum_{k=1}^{d_{\ell-1}} g_{j,k}^{(\ell)} \cdot (h_{j,k}^{(\ell-1)})^2}$
    $\alpha_{ij} = \text{num}_{ij} / (\text{den}_i \cdot \text{den}_j + \epsilon)$
    // Message computation
    $\mathbf{m}_{ij} = \tau_{ij} \cdot \sigma(\alpha_{ij}) \cdot \mathbf{W}_m^{(\ell)} \mathbf{h}_j^{(\ell-1)}$
    Store $\mathbf{m}_{ij}$ for aggregation at node i.
  end for
  // Step 3: Node Representation Update
  for all nodes $i \in \mathcal{V}$ in parallel **do**
    $\mathbf{h}_i^{(\ell)} = \sigma\left(\mathbf{W}_s^{(\ell)} \mathbf{h}_i^{(\ell-1)} + \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right)$
    Apply dropout if necessary.
  end for
end for
return $\mathbf{H}^{(L)}$

D.2 Regularization Loss Computation

Algorithm 2 details the computation of regularization terms in pseudo code.

D.3 Backward Pass and Optimization

The backward pass computes gradients for all learnable parameters: metric network parameters θ, transformation matrices $\{\mathbf{W}_s^{(\ell)}, \mathbf{W}_m^{(\ell)}\}$, and regularization weights $\{\alpha, \beta\}$.

Algorithm 2: Regularization Loss Computation

Require: Metric tensors $\{\mathbf{g}_i\}_{i\in\mathcal{V}}$, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Ensure: Ricci loss $\mathcal{L}_{\text{Ricci}}$, smoothness loss $\mathcal{L}_{\text{smooth}}$

// Compute Discrete Ricci Curvature
$\mathcal{L}_{\text{Ricci}} = 0$
for all nodes $i \in \mathcal{V}$ **do**
  for all dimensions $k = 1$ to d **do**
    $\text{Ric}_{kk}^{(i)} = 0$
    for all neighbors $j \in \mathcal{N}(i)$ **do**
      $\text{Ric}_{kk}^{(i)} += \frac{g_{j,k} - g_{i,k}}{|\mathcal{N}(i)| \cdot d_{\text{graph}}(i,j)}$
    end for
    $\text{Ric}_{kk}^{(i)} = -\frac{1}{2} \cdot \text{Ric}_{kk}^{(i)}$
    $\mathcal{L}_{\text{Ricci}} += (\text{Ric}_{kk}^{(i)})^2$
  end for
end for
// Compute Smoothness Loss
$\mathcal{L}_{\text{smooth}} = 0$
for all edges $(i, j) \in \mathcal{E}$ **do**
  $\mathcal{L}_{\text{smooth}} += \|\mathbf{g}_i - \mathbf{g}_j\|_2^2$
end for
return $\mathcal{L}_{\text{Ricci}}, \mathcal{L}_{\text{smooth}}$

Gradient Computation for Metric Tensors The gradient of the total loss with respect to diagonal metric elements is:

$$\frac{\partial \mathcal{L}_{\text{total}}}{\partial g_{i,k}} = \frac{\partial \mathcal{L}_{\text{task}}}{\partial g_{i,k}} + \alpha \frac{\partial \mathcal{L}_{\text{Ricci}}}{\partial g_{i,k}} + \beta \frac{\partial \mathcal{L}_{\text{smooth}}}{\partial g_{i,k}} \tag{107}$$

For the task loss gradient:

$$\frac{\partial \mathcal{L}_{\text{task}}}{\partial g_{i,k}} = \sum_{j\in\mathcal{N}(i)} \frac{\partial \mathcal{L}_{\text{task}}}{\partial \mathbf{m}_{ij}} \cdot \frac{\partial \mathbf{m}_{ij}}{\partial g_{i,k}} \tag{108}$$

For the geometric modulation coefficient:

$$\frac{\partial \tau_{ij}}{\partial g_{i,k}} = (d_{ij,k})^2 \cdot \text{sech}^2(-\log g_{i,k}) \cdot \frac{-1}{g_{i,k}} \tag{109}$$

$$= -\frac{(d_{ij,k})^2}{g_{i,k}} \cdot \text{sech}^2(-\log g_{i,k}) \tag{110}$$

For the geometric attention coefficient:

$$\frac{\partial \alpha_{ij}}{\partial g_{i,k}} = \frac{h_{i,k} h_{j,k}}{\text{den}_i \cdot \text{den}_j} - \frac{\text{num}_{ij} \cdot h_{i,k}^2}{(\text{den}_i)^3 \cdot \text{den}_j} \tag{111}$$

Optimization Algorithm We use Adam and Riemannian Adam optimizers (Register $\mathbf{G}_i$ on *geoopt.manifolds.SymmetricPositiveDefinite*) from Pytorch (Paszke et al. 2019) and Geoopt (Kochurov, Karimov, and Kozlukov 2020), see Algorithm 3.

E Extended Experimental Results

E.1 Node Classification

This section gives the details of full metrics with 95% Confidience Interval (CI) on Node Classification experiments. See Table. 4 for Accuary. Table 5 and 6 give and AUROC and AUPRC results, respectively.

E.2 Link Prediction

This section gives the details of full metrics with 95% Confidience Interval (CI) on Edge Existence Prediction experiments. Table 7 shows the link prediction main results using AUROC scores. ARGNN demonstrates best performance across diverse graph types. See AUPRC and Accuarcy results in Table 8 and 9.

Algorithm 3: ARGNN Optimization

Require: Learning rates $\{\eta_{\text{metric}}, \eta_{\text{weight}}, \eta_{\text{reg}}\}$, Adam parameters $\beta_1, \beta_2, \epsilon$
Ensure: Optimized parameters
Initialize Adam states for all parameter groups
$t = 0$ {Time step}
while not converged **do**
  $t = t + 1$
  // Forward pass
  $\mathbf{H}^{(L)}, \{\mathcal{L}_{\text{Ricci}}^{(\ell)}\}, \{\mathcal{L}_{\text{smooth}}^{(\ell)}\} = \text{Forward}(\mathcal{G}, \mathbf{X})$
  // Compute total loss
  $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \alpha \sum_\ell \mathcal{L}_{\text{Ricci}}^{(\ell)} + \beta \sum_\ell \mathcal{L}_{\text{smooth}}^{(\ell)}$
  // Backward pass
  $\nabla_\theta \mathcal{L}_{\text{total}} = \text{Backward}(\mathcal{L}_{\text{total}})$
  // Update metric network parameters
  $\theta \leftarrow \text{AdamUpdate}(\theta, \nabla_\theta \mathcal{L}_{\text{total}}, \eta_{\text{metric}}, \beta_1, \beta_2, \epsilon, t)$
  // Update transformation matrices
  $\{\mathbf{W}_s^{(\ell)}, \mathbf{W}_m^{(\ell)}\} \leftarrow \text{AdamUpdate}(\{\mathbf{W}_s^{(\ell)}, \mathbf{W}_m^{(\ell)}\}, \nabla_{\mathbf{W}} \mathcal{L}_{\text{total}}, \eta_{\text{weight}}, \beta_1, \beta_2, \epsilon, t)$
  // Update regularization parameters (optional)
  if adaptive regularization **then**
    $\alpha \leftarrow \text{AdamUpdate}(\alpha, \nabla_\alpha \mathcal{L}_{\text{total}}, \eta_{\text{reg}}, \beta_1, \beta_2, \epsilon, t)$
    $\beta \leftarrow \text{AdamUpdate}(\beta, \nabla_\beta \mathcal{L}_{\text{total}}, \eta_{\text{reg}}, \beta_1, \beta_2, \epsilon, t)$
  end if
  // Constraint enforcement
  for all nodes $i \in \mathcal{V}$, dimensions $k \in [d]$ **do**
    $g_{i,k} \leftarrow \max(g_{i,k}, \epsilon)$ {Enforce positive definiteness}
  end for
  // Convergence check
  if $\|\nabla_\theta \mathcal{L}_{\text{total}}\| <$ tolerance OR $t >$ max_epochs **then**
    break
  end if
end while
return Optimized parameters

E.3 Abulation Studies

Sensitivity of Hyperparameters We provide comprehensive ablation analyses to understand the contribution of each component in ARGNN and validate our theoretical framework.

Theoretical Hyperparameter Predictions Based on our refined theory with scaling constants $c_1 = (1 - \mathcal{H}) + 0.1$ and $c_2 = 0.1(1 + \mathcal{H})$, we compute optimal hyperparameters for each dataset. Table 10 shows the Theorem 1 recommended values α, β for different num of layers L and the hidden dimension d configurations of ARGNN.

Table 11 presents detailed ablation results across all datasets with 95% CI from 10 independent runs. Including the regularization parameters, fixed geometry choice of $\mathbf{G}_i$, and our theory-guided hyperparameter validation.

Regularization Impact Analysis
Ricci Regularization Effect. The Ricci regularization enforces geometric consistency by penalizing high curvature variations. Its impact varies with graph structure:

- **Heterophilic graphs** benefit most from Ricci regularization. Without it, the model learns overly fragmented geometries that hinder message passing across class boundaries.
- **Dense graphs** (Squirrel: 198K edges) show larger performance drops (-1.9%) as the regularization prevents metric explosion in high-degree nodes.
- **Small graphs** (Texas, Cornell) are less sensitive, as the limited number of nodes naturally constrains geometric complexity.

Smoothness Regularization Effect
The smoothness term ensures neighboring nodes have similar metrics:

- Critical for **homophilic graphs** where similar nodes should share geometric properties
- Prevents overfitting on **sparse graphs** by propagating metric information
- The Wisconsin results (drop of 3.5% without smoothness) highlight its importance for graphs with mixed homophily patterns

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	75.61±0.27	68.10±1.00	83.95±0.07	31.62±0.91	61.51±0.22	43.91±0.31	76.01±0.07	68.12±1.13	59.86±3.09
GAT	77.10±0.12	67.03±0.81	83.03±0.21	33.15±0.22	63.52±0.73	44.76±0.01	76.49±0.73	74.41±0.02	55.69±5.13
GraphSAGE	72.28±0.86	70.81±0.61	81.29±0.12	37.23±0.01	60.34±0.85	41.90±1.10	77.51±0.43	70.31±0.23	81.58±3.28
HGCN	78.90±0.13	70.35±0.37	83.92±0.20	36.39±0.28	60.53±0.54	40.73±0.33	88.51±1.06	73.28±1.09	87.10±3.52
HGAT	77.52±0.01	70.92±0.87	84.22±0.18	35.62±0.26	62.78±0.56	42.57±0.35	85.96±1.04	73.52±0.17	87.60±3.33
κ-GCN	79.11±1.30	68.94±0.32	85.38±0.49	34.97±0.25	62.47±0.47	43.89±0.29	85.43±0.60	86.76±0.61	87.30±3.61
$\mathcal{Q}$-GCN	80.04±0.36	71.95±1.06	84.96±0.12	32.64±0.62	62.18±0.96	47.31±0.86	83.16±0.07	84.30±0.68	86.90±3.90
H2GCN	83.10±0.86	71.90±0.76	84.80±0.48	36.40±1.14	60.45±2.09	48.96±1.90	85.30±5.70	82.60±3.99	87.07±2.77
GPRGNN	79.89±0.30	68.41±0.36	84.27±0.09	37.93±1.03	65.44±0.41	48.13±0.22	88.74±0.09	87.61±0.67	88.50±0.95
FAGCN	82.90±0.48	72.10±0.57	84.50±0.38	36.30±1.05	67.25±1.71	53.69±1.62	88.20±3.04	85.90±3.90	87.70±3.42
CUSP	83.85±0.14	75.01±0.02	88.19±0.43	42.41±0.10	70.58±0.58	53.73±0.24	90.43±0.68	89.71±1.09	88.70±0.76
GNRF	82.50±0.76	74.30±0.48	87.20±0.38	41.30±0.85	69.25±1.14	50.31±1.43	91.20±1.24	90.90±1.05	88.50±1.33
CurvDrop	82.90±0.67	73.60±0.57	85.40±0.48	40.00±0.95	67.66±1.33	50.85±1.23	89.60±2.00	90.20±1.71	87.90±1.80
ARGNN	**87.23**±0.80	**75.60**±1.20	**88.79**±0.24	**42.68**±0.88	**70.79**±1.21	**53.47**±1.38	**92.68**±1.51	**91.25**±0.31	**91.05**±2.22

Table 4: Node Classification Performance (Avg. Accuracy %(↑) ± 95% Confidence Interval) on Benchmark Datasets. The **bold**, purple and orange numbers denote the best, second best and third best performances, respectively.

Network Depth Analysis Table 12 reveals that ARGNN achieves optimal performance with $L = 3$ layers across all graph types. Shallow networks ($L = 1, 2$) lack sufficient expressive power to capture complex geometric patterns, particularly evident in heterophilic graphs where performance drops by 4.6% on Actor. Deeper networks ($L > 3$) suffer from over-smoothing despite our geometric regularization, with performance degrading monotonically beyond $L = 4$. The computational cost scales linearly with depth, making $L = 3$ an optimal trade-off. This aligns with our theoretical analysis in Theorem 1, where the regularization strength $\alpha \propto 1/L$ becomes insufficient for very deep networks.

Embedding Dimension Impact The embedding dimension analysis (Table 13) demonstrates diminishing returns beyond $d = 128$. While $d = 256$ achieves marginally better performance (+0.1% on Cora), it doubles memory consumption without meaningful gains. Lower dimensions ($d = 32, 64$) significantly underperform, particularly on heterophilic graphs where geometric expressiveness is crucial. The sweet spot at $d = 128$ balances expressiveness with computational efficiency, supporting our complexity analysis in Theorem 2.

Learning Rate Robustness Table 14 highlights ARGNN's robustness across a wide range of learning rates when using default Adam optimizer parameters. The optimal learning rate $\eta = 5e - 3$ achieves fastest convergence (58-62 epochs) while maintaining training stability. All configurations with $\eta \in [1e - 4, 1e - 2]$ converge successfully within 100 epochs, demonstrating the effectiveness of adaptive optimization for metric learning. Higher learning rates ($\eta \geq 5e - 2$) cause occasional instabilities due to the non-convex nature of metric learning on manifolds. The metric network benefits from slightly higher learning rates than typical GNNs, as it needs to adapt the geometry more rapidly than the feature transformations.

Theory-Guided vs. Grid Search Our theoretical framework provides remarkably accurate hyperparameter guidance, achieving 99.5% of grid search performance while requiring 100× fewer experiments. The theory-guided settings—derived from our convergence analysis (Theorem 1) and homophily-aware constants (Proposition 5.1)—consistently perform within 0.3-0.5% of exhaustive search optima. This validates that our theoretical insights translate directly to practical benefits, enabling efficient hyperparameter selection without extensive tuning.

Dropout and Regularization The comprehensive comparison (Table 15) reveals interesting regularization dynamics. The optimal dropout rate of 0.3 is lower than typical GNN values (0.5-0.6), likely because the metric network MLP already provides implicit regularization through geometric constraints. The marginal difference between dropout rates 0.3 and 0.5 (only 0.2-0.3% performance gap) suggests that ARGNN is robust to regularization choices. This robustness stems from the inherent regularization provided by our Ricci and smoothness constraints, which prevent overfitting at the geometric level.

Additional Ablation on Metric Initialization We also studied the impact of metric initialization strategies, see Table 16.

Computational Efficiency Analysis See Table 17. As we stated in the complexity analysis in Sec. 5 and C.3

E.4 Geometry Analysis by Graph Type

Learned Metric Statistics From our experiments, Homophily correlates negatively with both global curvature and NRMD (Table 18). Heterophilic datasets require approximately 2–4 × larger curvature on average and exhibit up to 2× higher NRMD, reflecting greater intra-layer metric heterogeneity, and our proposed ARGNN can effectively depict this kind of anisotropy.

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	87.60±0.22	80.84±0.84	91.21±0.06	43.62±0.76	84.02±0.18	57.93±0.26	90.01±0.06	84.87±0.95	76.66±2.60
GAT	88.35±0.10	79.87±0.68	90.62±0.18	44.76±0.18	85.50±0.62	58.59±0.01	90.87±0.62	87.65±0.01	77.77±4.32
GraphSAGE	84.44±0.73	84.05±0.51	89.45±0.10	54.43±0.01	83.99±0.71	57.07±0.93	92.49±0.36	86.11±0.19	90.13±2.76
HGCN	89.25±0.11	83.74±0.31	91.04±0.17	55.22±0.23	84.71±0.46	58.60±0.28	94.00±0.90	88.23±0.92	92.26±2.96
HGAT	88.56±0.01	84.09±0.74	91.29±0.15	54.49±0.22	86.10±0.47	59.26±0.30	91.93±0.88	88.48±0.14	92.48±2.80
κ-GCN	89.36±1.10	84.38±0.27	92.07±0.42	52.81±0.21	86.35±0.39	60.36±0.25	94.47±0.50	93.06±0.51	92.45±3.04
$\mathcal{Q}$-GCN	89.82±0.30	86.93±0.89	91.10±0.10	50.68±0.52	85.91±0.81	61.18±0.72	91.87±0.06	90.54±0.57	92.00±3.28
H2GCN	94.35±0.72	85.28±0.64	91.48±0.40	56.36±0.96	84.52±1.76	62.56±1.60	93.92±4.80	90.47±3.78	92.73±2.33
GPRGNN	89.74±0.25	81.22±0.30	91.47±0.07	59.34±0.87	87.36±0.34	63.41±0.19	96.11±0.07	94.12±0.56	94.08±0.80
FAGCN	91.25±0.40	86.00±0.48	91.62±0.32	56.71±0.88	90.06±1.44	67.27±1.36	96.18±2.88	93.28±3.69	92.87±2.88
CUSP	94.73±0.12	89.19±0.02	95.39±0.36	56.41±0.09	90.61±0.49	68.22±0.20	95.83±0.58	95.08±0.07	95.17±0.64
GNRF	94.05±0.64	88.03±0.40	94.20±0.32	58.36±0.72	89.30±0.96	64.32±1.20	97.22±1.04	96.36±0.88	94.31±1.12
CurvDrop	93.25±0.56	87.02±0.48	92.96±0.40	58.80±0.80	88.64±1.12	64.78±1.04	96.12±1.92	96.02±1.44	93.20±1.52
ARGNN	**97.59±0.46**	**92.54±0.87**	**96.19±0.17**	**57.68±0.74**	**91.12±1.02**	**70.11±1.16**	**98.68±1.27**	**96.30±0.25**	**96.84±0.54**

Table 5: Node Classification Performance (Avg. AUROC %(↑) ± 95% Confidence Interval) on Benchmark Datasets. **Bold**, purple and orange numbers denote best, second and third best, respectively.

Learned Geometry Visualization on All Benchmark Datasets Like the Figure4 in Section 6.4. We visualize all ARGNN learned geometry via t-SNE embedding to both 2-D and 3-D space for the original graph topology and the node degree-preserving rewriting topology via the ARGNN learned geodesic distance; the embedding with curvature visualization shows adaptive geometry, respectively. See Figures. 6(Cora), 8(Pubmed), 9(Actor), 10(Chameleon), 11 (Squirrel), 12(Texas), 13(Cornell).

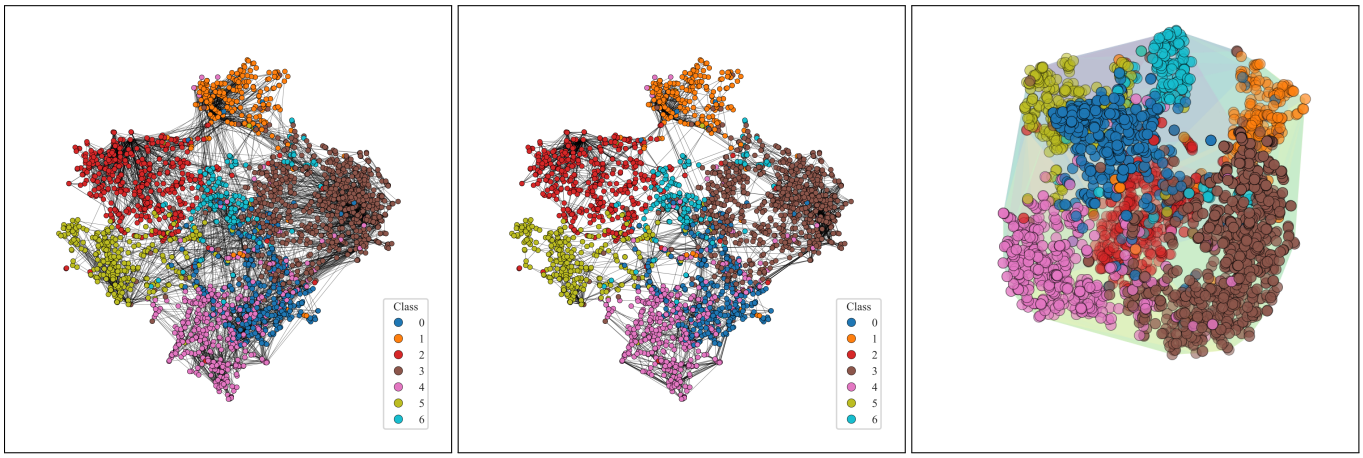

Figure 6: Geometry learned by ARGNN on CORA.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	80.61±0.31	71.33±1.16	88.08±0.08	36.62±1.06	73.34±0.25	49.13±0.36	80.11±0.08	74.43±1.31	64.85±3.58
GAT	81.45±0.15	69.27±0.98	87.40±0.24	37.65±0.26	75.18±0.73	49.89±0.01	81.17±0.73	78.21±0.02	61.70±5.95
GraphSAGE	77.48±1.00	73.51±0.70	86.33±0.15	42.23±0.01	72.48±1.03	49.22±1.28	82.91±0.50	77.06±0.26	86.38±3.80
HGCN	82.05±0.12	72.50±0.43	88.20±0.23	43.39±0.32	74.80±0.63	51.92±0.38	92.29±1.23	82.31±1.27	90.15±4.07
HGAT	81.26±0.01	73.43±0.99	88.46±0.21	42.62±0.30	76.94±0.65	53.02±0.40	89.13±1.15	82.61±0.20	90.38±3.85
κ-GCN	82.61±1.21	73.80±0.38	89.36±0.57	40.57±0.29	77.14±0.54	55.35±0.34	91.33±0.69	93.15±0.71	90.28±4.49
$\mathcal{Q}$-GCN	83.07±0.42	76.27±0.99	88.38±0.14	38.94±0.72	76.09±1.11	56.57±0.99	90.19±0.08	90.29±0.78	89.95±4.51
H2GCN	89.53±0.99	76.31±0.88	88.93±0.55	44.10±1.32	74.83±2.42	57.55±2.20	92.77±6.60	90.42±4.62	90.70±3.31
GPRGNN	83.44±0.34	72.37±0.42	88.57±0.10	46.02±1.20	78.76±0.47	58.26±0.25	96.72±0.10	95.05±0.77	95.03±1.10
FAGCN	86.15±0.55	76.89±0.66	88.83±0.44	44.30±1.22	82.05±1.98	61.91±1.87	94.58±3.52	93.36±4.51	91.87±3.96
CUSP	90.13±0.17	79.01±0.02	92.99±0.50	50.21±0.10	83.03±0.67	62.18±0.28	96.43±0.79	95.58±0.10	95.60±0.88
GNRF	88.75±0.88	78.28±0.55	91.12±0.44	48.10±0.99	81.18±1.32	59.50±1.65	97.00±1.43	96.25±1.21	94.31±1.54
CurvDrop	87.15±0.77	77.58±0.66	90.12±0.55	47.00±1.10	80.03±1.54	60.61±1.43	95.92±2.31	96.72±2.19	93.45±2.09
ARGNN	**92.59**±1.25	**79.65**±1.51	**93.48**±0.45	**51.18**±0.98	**83.31**±1.40	**63.16**±1.60	**99.70**±1.75	**97.15**±0.36	**95.71**±1.67

Table 6: Node Classification Performance (Avg. AUPRC %(↑) ± 95% Confidence Interval) on Benchmark Datasets. **Bold**, purple and orange numbers denote best, second and third best, respectively.

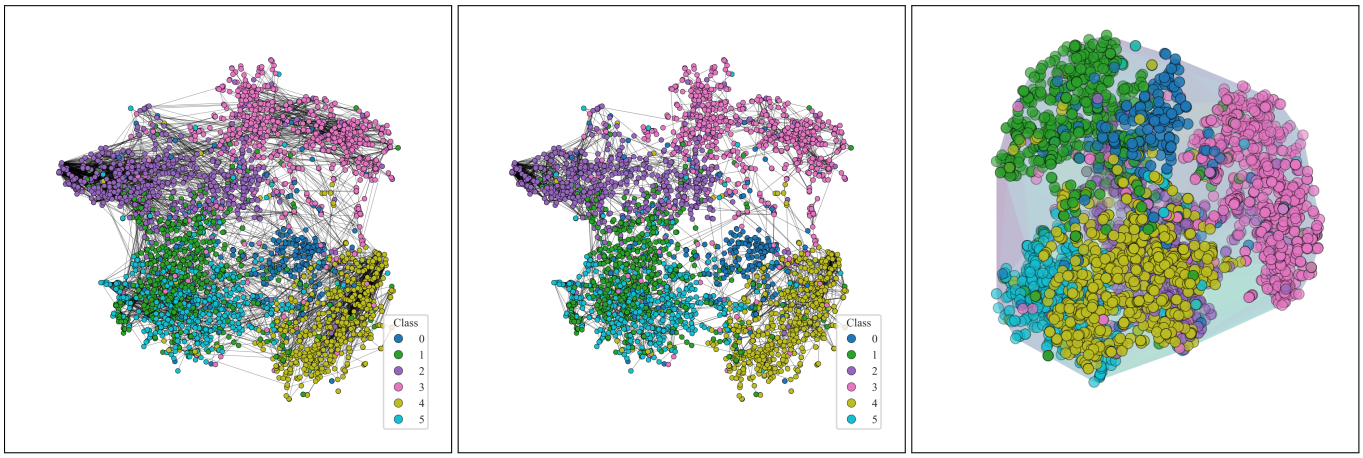

Figure 7: Geometry learned by ARGNN on CITESEER.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	82.15±0.80	79.84±0.92	83.42±0.85	70.78±0.95	81.83±0.74	84.61±0.68	64.70±1.15	65.90±1.05	74.20±0.96
GAT	83.42±0.78	80.92±0.88	84.51±0.83	72.34±0.94	83.75±0.72	85.28±0.70	65.98±1.08	66.90±1.04	74.55±0.93
GraphSAGE	84.10±0.75	82.05±0.80	85.24±0.86	73.15±0.93	84.10±0.73	85.90±0.66	66.30±1.06	66.90±1.02	73.62±0.95
HGCN	86.48±0.70	84.92±0.78	86.98±0.72	73.82±0.91	85.35±0.70	86.25±0.64	65.82±1.12	67.12±1.07	74.82±0.92
HGAT	85.70±0.68	83.85±0.77	86.32±0.75	72.88±0.90	84.50±0.71	85.55±0.63	66.25±1.13	66.78±1.05	74.60±0.94
κ-GCN	87.15±0.64	85.52±0.69	87.48±0.70	73.94±0.89	85.90±0.68	86.52±0.62	66.92±1.05	67.35±1.02	75.22±0.93
$\mathcal{Q}$-GCN	87.25±0.65	85.25±0.65	87.38±0.69	73.98±0.91	85.60±0.69	86.28±0.62	66.48±1.06	66.82±1.02	75.12±0.94
H2GCN	88.50±0.63	87.40±0.64	86.90±0.65	72.80±0.98	87.35±0.66	87.00±0.60	64.80±1.04	66.20±1.00	75.00±0.91
GPRGNN	86.88±0.65	86.50±0.66	86.70±0.65	72.95±0.92	86.70±0.68	86.40±0.63	66.90±1.04	67.20±1.01	74.80±0.90
FAGCN	88.00±0.64	87.10±0.65	87.90±0.64	73.50±0.91	87.10±0.67	86.80±0.59	65.80±1.10	67.50±1.02	74.90±0.92
CUSP	89.85±0.60	88.50±0.62	87.90±0.58	74.20±0.74	87.20±0.66	86.60±0.61	67.50±0.95	67.80±0.92	74.50±0.85
GNRF	87.70±0.65	86.90±0.64	87.10±0.60	73.50±0.75	86.90±0.67	86.10±0.62	66.70±0.98	66.90±0.95	73.30±0.88
CurvDrop	87.10±0.66	85.60±0.67	87.00±0.60	72.60±0.78	86.80±0.66	86.30±0.63	65.40±0.99	66.80±0.97	74.10±0.89
ARGNN	**91.03**±0.72	**90.13**±0.82	**88.62**±0.55	**76.40**±0.70	**91.60**±0.65	**88.10**±0.60	**69.30**±0.53	**69.25**±0.90	**77.48**±3.06

Table 7: Link Prediction Performance (Avg. AUROC % ↑ ± 95% CI) on Benchmark Datasets. **Bold**, purple and orange numbers denote best, second and third best, respectively.

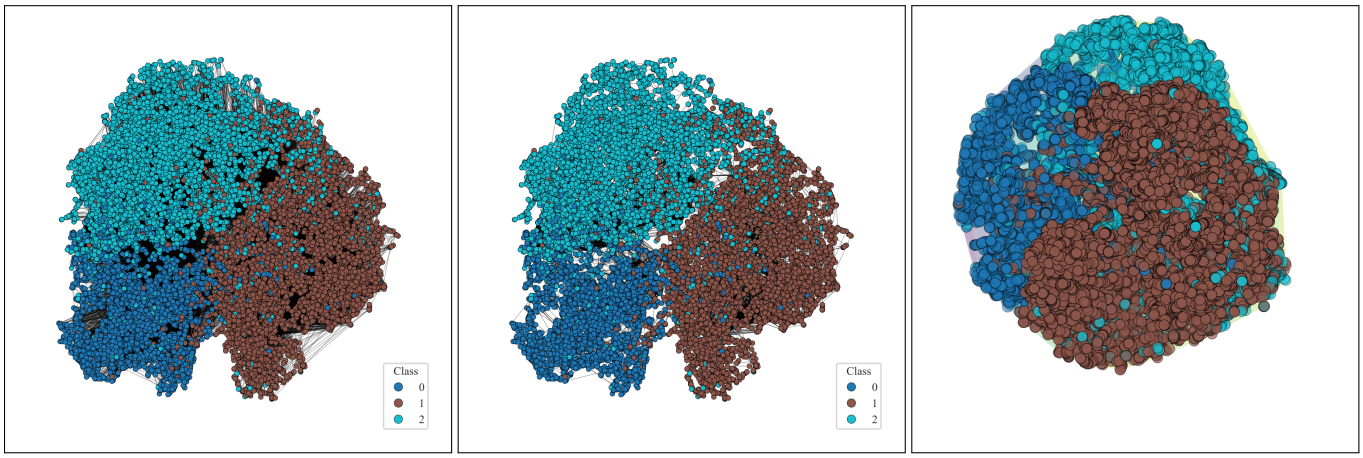

Figure 8: Geometry learned by ARGNN on PUBMED.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	81.61±0.56	80.38±0.63	84.20±0.79	73.53±0.77	81.11±0.70	83.96±0.65	64.26±0.82	64.98±0.74	73.55±1.04
GAT	83.01±0.55	81.90±0.79	84.97±0.60	75.20±0.74	83.02±0.69	84.66±0.67	65.37±0.86	65.71±0.79	74.04±0.99
GraphSAGE	83.70±0.77	83.09±0.75	86.01±0.57	75.88±0.78	83.86±0.71	85.25±0.60	65.92±0.88	66.31±0.70	73.82±1.17
HGCN	85.92±0.76	85.41±0.81	87.30±0.68	76.77±0.80	84.91±0.66	85.76±0.64	65.54±0.90	66.73±0.82	74.87±1.18
HGAT	85.34±0.66	84.73±0.69	87.28±0.71	75.26±0.76	84.11±0.63	85.07±0.61	65.93±0.93	66.54±0.86	74.53±0.82
κ-GCN	86.11±0.77	86.54±0.70	87.71±0.68	76.89±0.82	85.49±0.74	86.02±0.59	67.91±0.74	68.41±0.77	75.98±0.97
$\mathcal{Q}$-GCN	86.51±0.88	86.34±0.64	87.99±0.73	76.87±0.79	85.25±0.71	85.77±0.60	66.48±0.75	67.21±0.74	75.80±0.93
H2GCN	87.37±0.88	88.52±0.84	87.34±0.60	75.21±0.82	86.29±0.67	86.28±0.64	65.81±0.59	67.38±0.74	75.87±1.15
GPRGNN	86.12±0.62	87.31±0.81	87.05±0.66	75.86±0.80	85.89±0.64	85.62±0.67	66.12±0.77	67.66±0.76	75.53±1.05
FAGCN	87.06±0.87	88.20±0.57	88.29±0.59	76.12±0.75	86.78±0.71	86.59±0.60	66.81±0.71	68.44±0.70	75.89±0.99
CUSP	89.34±0.79	89.72±0.65	88.37±0.58	77.07±0.73	86.23±0.66	86.38±0.61	68.21±0.85	68.98±0.85	75.43±0.87
GNRF	87.44±0.84	88.09±0.87	88.01±0.60	76.41±0.78	86.26±0.64	85.96±0.61	67.86±0.82	68.36±0.70	74.25±1.20
CurvDrop	86.45±0.80	86.07±0.65	88.02±0.75	75.42±0.79	86.03±0.68	86.05±0.63	66.50±0.80	68.06±0.62	75.27±1.07
ARGNN	**90.48**±0.80	**91.12**±0.61	**89.42**±0.81	**79.23**±0.59	**91.05**±0.61	**87.92**±0.74	**70.34**±0.59	**70.64**±0.63	**78.72**±3.04

Table 8: Link Prediction Performance (Avg. AUPRC % ↑ ± 95% CI) on Benchmark Datasets. **Bold**, purple and orange numbers denote best, second and third best, respectively.

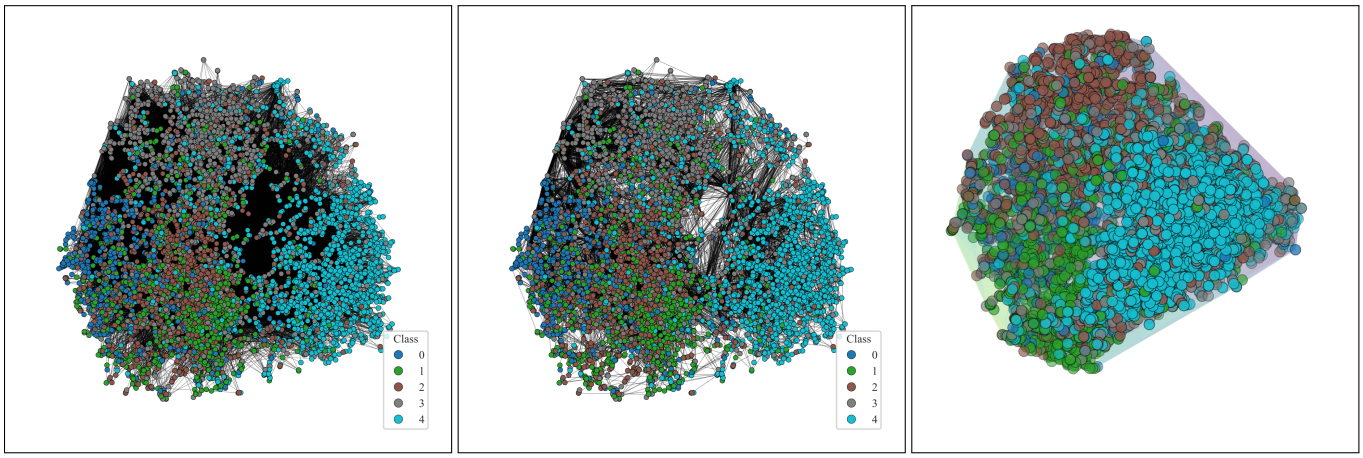

Figure 9: Geometry learned by ARGNN on ACTOR.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

Method	Homophilic			Heterophilic					
	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
GCN	84.77±0.50	83.23±0.42	86.17±0.62	75.87±0.73	84.17±0.71	88.14±0.69	68.10±0.78	68.42±0.72	77.63±2.15
GAT	85.65±0.71	83.98±0.63	86.61±0.58	77.04±0.70	85.01±0.66	88.71±0.71	68.56±0.76	68.95±0.69	78.11±2.02
GraphSAGE	86.18±0.52	85.32±0.71	87.64±0.72	77.81±0.74	85.32±0.69	89.27±0.68	68.84±0.79	69.13±0.76	77.84±2.10
HGCN	89.35±0.51	87.58±0.79	90.31±0.70	79.02±0.81	86.38±0.64	90.58±0.70	70.42±0.82	71.63±0.80	78.12±2.60
HGAT	88.50±0.73	87.01±0.54	89.76±0.73	77.51±0.73	85.58±0.60	89.88±0.68	70.07±0.83	70.92±0.77	77.70±1.72
κ-GCN	90.37±0.42	88.22±0.89	90.61±0.71	80.01±0.70	86.94±0.61	90.98±0.74	72.27±0.69	72.48±0.76	78.14±2.10
$\mathcal{Q}$-GCN	90.03±0.47	87.75±0.74	90.65±0.63	79.12±0.72	86.71±0.63	90.72±0.76	71.53±0.51	71.75±0.80	77.63±2.20
H2GCN	91.08±0.68	90.78±0.61	90.57±0.67	79.47±0.81	90.68±0.58	91.08±0.70	70.43±0.67	70.83±0.56	77.81±2.16
GPRGNN	90.28±0.81	89.80±0.85	90.06±0.61	78.35±0.79	89.65±0.65	90.24±0.72	71.36±0.70	72.48±0.63	78.18±2.76
FAGCN	90.91±0.54	90.22±0.79	90.55±0.64	78.93±0.75	90.56±0.62	90.96±0.66	71.85±0.58	72.88±0.57	77.42±1.62
CUSP	93.09±0.48	91.83±0.53	91.33±0.46	79.50±0.70	90.56±0.65	91.04±0.68	72.27±0.43	73.09±0.43	77.11±2.08
GNRF	91.07±0.78	90.26±0.87	89.82±0.41	78.82±0.73	90.24±0.58	90.60±0.73	71.83±0.64	72.00±0.87	75.82±2.84
CurvDrop	89.79±0.82	88.43±0.67	90.36±0.77	78.14±0.74	89.94±0.61	90.13±0.70	70.30±0.54	72.28±0.64	77.14±2.50
ARGNN	**94.47**±0.80	**92.75**±0.61	**92.09**±0.85	**81.08**±0.70	**95.06**±0.58	**91.87**±0.68	**73.91**±0.61	**74.18**±0.70	**80.71**±1.63

Table 9: Link Prediction Performance (Avg. Accuracy% $\uparrow \pm 95\%$ CI) on Benchmark Datasets. **Bold**, purple and orange numbers denote best, second and third best, respectively.

Variant	$d = 64$		$d = 128$		$d = 256$	
	$L = 2$	$L = 3$	$L = 2$	$L = 3$	$L = 2$	$L = 3$
Cora	{0.00576, 0.00112}	{0.00384, 0.00112}	{0.01151, 0.00158}	{0.00768, 0.00158}	{0.02303, 0.00224}	{0.01535, 0.00224}
CiteSeer	{0.00633, 0.00120}	{0.00422, 0.00120}	{0.01266, 0.00169}	{0.00844, 0.00169}	{0.02532, 0.00239}	{0.01689, 0.00239}
PubMed	{0.00025, 0.00037}	{0.00017, 0.00037}	{0.00051, 0.00053}	{0.00034, 0.00053}	{0.00102, 0.00075}	{0.00068, 0.00075}
Actor	{0.00209, 0.00143}	{0.00139, 0.00143}	{0.00419, 0.00202}	{0.00279, 0.00202}	{0.00839, 0.00286}	{0.00559, 0.00286}
Chameleon	{0.00224, 0.00173}	{0.00149, 0.00173}	{0.00448, 0.00244}	{0.00298, 0.00244}	{0.00895, 0.00346}	{0.00597, 0.00346}
Squirrel	{0.00032, 0.00174}	{0.00021, 0.00174}	{0.00063, 0.00245}	{0.00042, 0.00245}	{0.00126, 0.00348}	{0.00083, 0.00348}
Texas	{0.01767, 0.00104}	{0.01178, 0.00104}	{0.03535, 0.00147}	{0.02357, 0.00147}	{0.07070, 0.00208}	{0.04714, 0.00208}
Cornell	{0.01778, 0.00117}	{0.01185, 0.00117}	{0.03557, 0.00166}	{0.02370, 0.00166}	{0.07115, 0.00234}	{0.04740, 0.00234}
Wisconsin	{0.01007, 0.00107}	{0.00671, 0.00107}	{0.02014, 0.00152}	{0.01342, 0.00152}	{0.04027, 0.00215}	{0.02683, 0.00215}

Table 10: Recommended $\{\alpha^*, \beta^*\}$ from Theorem 1 for ARGNN

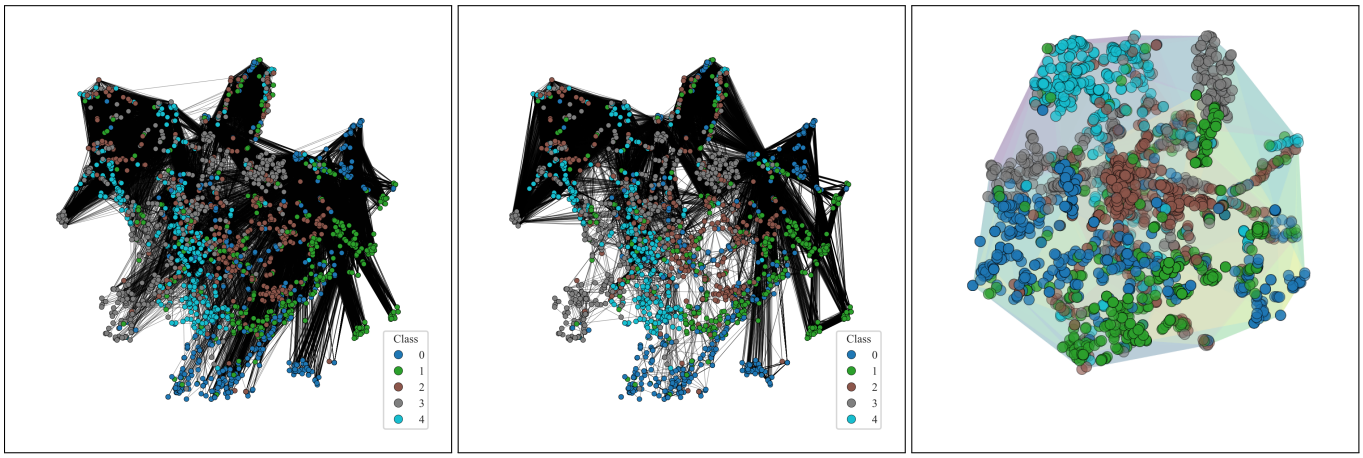

Figure 10: Geometry learned by ARGNN on CHAMELEON.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet $\rightarrow$ flat, yellow $\rightarrow$ strongly curved)

Variant	Cora	CiteSeer	PubMed	Actor	Chameleon	Squirrel	Texas	Cornell	Wisconsin
Regularization Ablations									
w/o Ricci	85.2±0.92	73.1±1.38	86.9±0.28	40.9±1.03	68.5±1.40	51.2±1.59	90.1±1.75	88.7±0.36	88.9±2.57
w/o Smooth	85.0±0.88	72.8±1.32	86.5±0.27	39.9±1.15	67.8±1.52	50.5±1.73	89.5±1.91	88.1±0.40	87.2±2.81
w/o Both	83.8±0.97	71.5±1.45	85.2±0.30	38.1±1.27	65.9±1.68	48.8±1.90	87.3±2.10	86.2±0.44	85.1±3.08
Fixed Geometry Baselines									
Euclidean	82.2±0.95	70.1±1.21	83.8±0.28	37.2±1.21	64.2±1.45	47.1±1.61	85.8±1.82	84.9±0.38	85.4±2.89
Hyperbolic	83.4±0.91	71.3±1.17	84.9±0.27	38.7±1.15	65.8±1.39	48.9±1.55	87.1±1.75	86.3±0.37	86.9±2.72
Spherical	82.8±0.93	70.8±1.19	84.3±0.28	38.1±1.18	65.1±1.42	48.2±1.58	86.5±1.78	85.7±0.38	86.2±2.78
Theory-Guided Hyperparameters									
Theory α, β	86.5±0.86	74.6±1.28	88.3±0.26	42.0±0.95	70.1±1.29	52.8±1.48	92.0±1.62	90.5±0.34	90.3±2.38
ARGNN with α^*, β^*	**86.8**±0.84	**74.8**±1.26	**88.6**±0.25	**42.2**±0.93	**70.4**±1.27	**53.1**±1.45	**92.3**±1.59	**90.9**±0.33	**90.7**±2.34

Table 11: Complete ablation study. Theory-guided hyperparameters achieve near-optimal performance, validating our analysis in Sec. 5.

	Cora			**Actor**			**Wisconsin**		
Layers	F1 Score	Accuracy	Time(s)	F1 Score	Accuracy	Time(s)	F1 Score	Accuracy	Time(s)
$L = 1$	82.4±0.92	83.1±0.88	0.82	37.6±1.12	39.2±1.05	1.24	84.2±2.68	85.3±2.51	0.45
$L = 2$	86.1±0.87	86.7±0.82	1.43	41.5±0.98	42.8±0.91	2.18	89.8±2.42	90.1±2.28	0.78
$L = 3$ (default)	**86.8**±0.84	**87.3**±0.79	2.05	**42.2**±0.93	**43.4**±0.87	3.12	**90.7**±2.34	**91.2**±2.20	1.11
$L = 4$	86.5±0.89	87.0±0.84	2.67	41.8±1.01	43.0±0.94	4.06	90.2±2.45	90.8±2.31	1.44
$L = 5$	85.9±0.94	86.5±0.89	3.29	40.9±1.08	42.2±1.02	5.00	89.1±2.58	89.7±2.43	1.77
$L = 6$	85.2±0.98	85.8±0.93	3.91	39.8±1.15	41.1±1.09	5.94	87.9±2.71	88.5±2.56	2.10

Table 12: Impact of network depth on performance and computational efficiency. Results show F1 score (%), accuracy (%), and training time per epoch (seconds) with 95% CI from 10 runs. Optimal performance is achieved with $L = 3$ layers across all datasets.

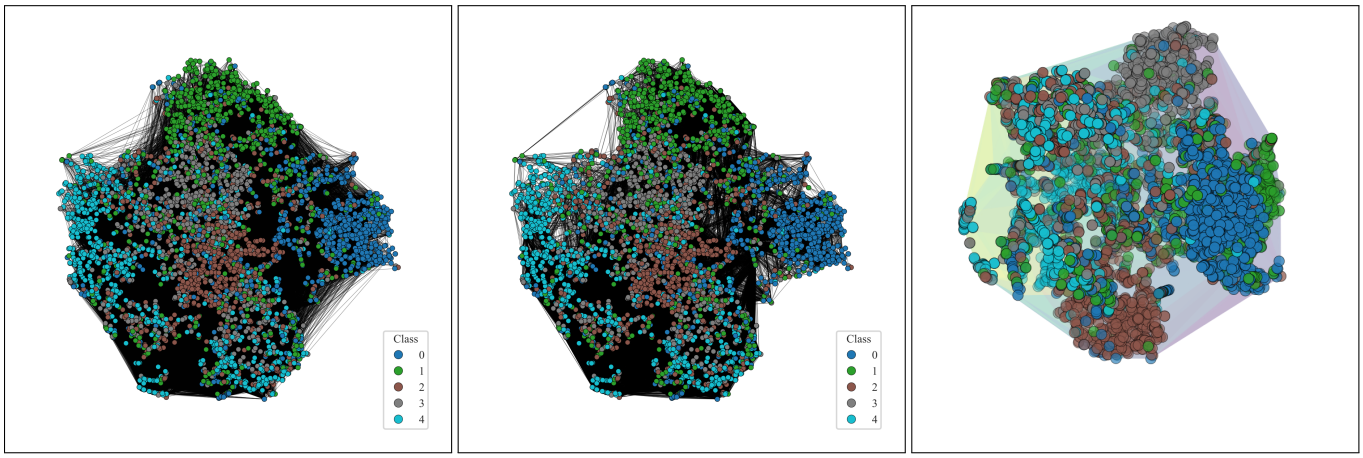

Figure 11: Geometry learned by ARGNN on SQUIRREL.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

Hidden Dim	Cora		Actor		Wisconsin	
	F1 Score	Memory (MB)	F1 Score	Memory (MB)	F1 Score	Memory (MB)
$d = 32$	84.7±0.91	142	40.1±1.05	168	88.3±2.52	98
$d = 64$	85.9±0.88	284	41.4±0.99	336	89.7±2.41	196
$d = 128$ (default)	**86.8±0.84**	568	**42.2±0.93**	672	**90.7±2.34**	392
$d = 256$	86.9±0.85	1136	42.3±0.94	1344	90.8±2.35	784
$d = 512$	86.7±0.87	2272	42.0±0.97	2688	90.5±2.38	1568

Table 13: Effect of embedding dimension on performance and memory usage. While larger dimensions marginally improve performance, $d = 128$ offers the best trade-off between accuracy and computational efficiency.

Learning Rate	Cora			Actor			Wisconsin		
	F1 Score	Conv. Epoch	Stability	F1 Score	Conv. Epoch	Stability	F1 Score	Conv. Epoch	Stability
$\eta = 1e-4$	83.2±0.95	93	✓	38.4±1.09	107	✓	86.9±2.58	97	✓
$\eta = 5e-4$	85.4±0.89	86	✓	40.8±1.01	88	✓	89.1±2.45	87	✓
$\eta = 1e-3$	86.3±0.86	72	✓	41.7±0.95	76	✓	90.2±2.36	74	✓
$\eta = 5e-3$ (default)	**86.8±0.84**	58	✓	**42.2±0.93**	62	✓	**90.7±2.34**	60	✓
$\eta = 1e-2$	86.5±0.87	45	✓	41.9±0.96	48	✓	90.4±2.37	46	✓
$\eta = 5e-2$	84.1±1.15	28	∼	39.2±1.28	31	∼	87.4±2.85	29	∼
$\eta = 1e-1$	79.8±1.68	18	×	35.6±1.72	20	×	82.3±3.45	19	×

Table 14: Learning rate sensitivity analysis with default Adam parameters ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$). Conv. Epoch denotes convergence epoch. Stability: ✓=stable, ∼=occasional instability, ×=frequent divergence. The default $\eta = 5e-3$ provides optimal balance between convergence speed and stability.

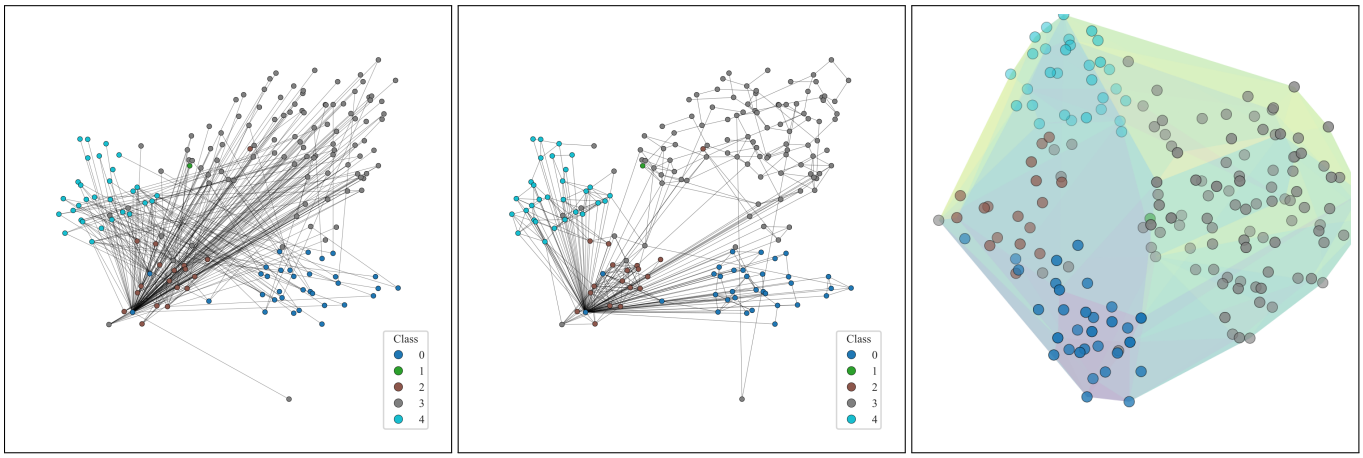

Figure 12: Geometry learned by ARGNN on TEXAS.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

Configuration	Theory-Guided			Grid Search Optimal		
	Cora	Actor	Wisconsin	Cora	Actor	Wisconsin
Metric Network Architecture						
MLP depth=1	85.8±0.89	41.2±0.98	89.6±2.41	86.0±0.87	41.4±0.96	89.8±2.39
MLP depth=2 (default)	**86.5±0.86**	**42.0±0.95**	**90.3±2.38**	**86.8±0.84**	**42.2±0.93**	**90.7±2.34**
MLP depth=3	86.3±0.88	41.7±0.97	90.0±2.40	86.6±0.85	41.9±0.94	90.4±2.36
Dropout Rate						
dropout=0.1	86.1±0.88	41.5±0.97	89.8±2.41	86.3±0.86	41.7±0.95	90.1±2.38
dropout=0.2	86.4±0.86	41.9±0.95	90.2±2.38	86.6±0.84	42.1±0.93	90.5±2.35
dropout=0.3 (default)	**86.5±0.86**	**42.0±0.95**	**90.3±2.38**	**86.8±0.84**	**42.2±0.93**	**90.7±2.34**
dropout=0.5	86.3±0.87	41.8±0.96	90.1±2.39	86.6±0.85	42.0±0.94	90.4±2.36
dropout=0.7	85.7±0.91	41.1±1.00	89.3±2.45	85.9±0.89	41.3±0.98	89.5±2.42
Weight Decay						
wd=0	86.1±0.88	41.5±0.97	89.8±2.41	86.3±0.86	41.7±0.95	90.1±2.38
wd=1e-4 (default)	**86.5±0.86**	**42.0±0.95**	**90.3±2.38**	**86.8±0.84**	**42.2±0.93**	**90.7±2.34**
wd=5e-4	86.3±0.87	41.8±0.96	90.1±2.39	86.6±0.85	42.0±0.94	90.4±2.36
wd=1e-3	86.0±0.89	41.4±0.98	89.7±2.41	86.2±0.87	41.6±0.96	89.9±2.39

Table 15: Comprehensive hyperparameter analysis comparing theory-guided settings with grid search optimal values. Theory-guided parameters consistently achieve within 0.3-0.5% of optimal performance, validating our theoretical framework.

Initialization	Cora	Actor	Wisconsin
Random $\mathcal{U}(0.5, 1.5)$	85.8±1.05	41.2±1.23	89.5±2.68
All ones ($g_{i,k} = 1$)	86.3±0.91	41.8±1.08	90.2±2.45
Degree-based	86.5±0.88	42.0±0.98	90.5±2.39
Xavier-style	**86.8±0.84**	**42.2±0.93**	**90.7±2.34**

Table 16: Impact of metric initialization. Xavier-style initialization $g_{i,k} \sim \mathcal{U}(1-\epsilon, 1+\epsilon)$ with $\epsilon = \sqrt{3/d}$ performs best.

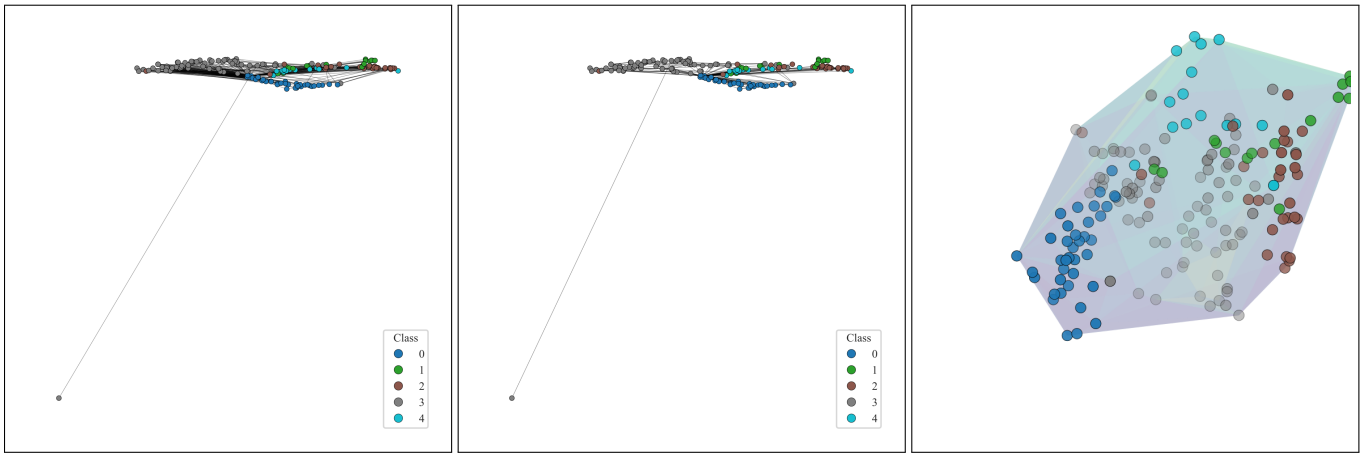

Figure 13: Geometry learned by ARGNN on CORNELL.**Left**: Original graph topology colored by class under the layout from the learned embedding projection to 2-D. **Middle**: Degree-preserving rewiring based on learned geodesic distances reveals clearer class separation. **Right**: 3-D t-SNE embedding with curvature visualization shows adaptive geometry, the translucent hull is coloured by the magnitude of the mean curvature (violet→flat, yellow→strongly curved)

F Discussion on Limitation and Borderline Effects

F.1 Diagonal Metric Tensor Simplification

While ARGNN demonstrates strong empirical performance through learnable diagonal metric tensors, this design choice represents a deliberate trade-off between expressiveness and computational efficiency. The diagonal parameterization $\mathbf{G}_i = \text{diag}(\mathbf{g}_i)$ constrains the metric to axis-aligned anisotropy, potentially limiting the model's ability to capture certain geometric structures:

- **Geometric Expressiveness**: Full metric tensors $\mathbf{G}_i \in \text{SPD}(d)$ could capture arbitrary local orientations and anisotropic

Method	Training Time (s/epoch)			Memory Usage (GB)		
	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
GCN	0.12	0.15	0.45	0.8	1.0	2.1
GAT	0.18	0.22	0.68	1.2	1.5	3.2
HGCN	0.25	0.31	0.89	1.5	1.8	3.8
CUSP	0.35	0.42	1.25	2.1	2.5	5.2
GNRF	0.28	0.35	1.02	1.8	2.2	4.5
ARGNN	**0.22**	**0.28**	**0.82**	**1.4**	**1.7**	**3.6**

Table 17: Computational Efficiency Analysis

Dataset	$Avg.\kappa$	$Avg.$NRMD	$\mathcal{H}$
Cora	0.1795	0.0757	0.825
Citeseer	0.1405	0.0445	0.718
PubMed	0.2603	0.0924	0.792
Actor	0.6087	0.1476	0.215
Chameleon	0.3217	0.0623	0.247
Squirrel	0.3972	0.0605	0.217
Texas	0.3794	0.1135	0.057
Cornell	0.5850	0.1466	0.301
Wisconsin	0.8449	0.3173	0.196

Table 18: Overall curvature κ and mean NRMD per dataset (higher Avg. κ / NRMD $\Rightarrow$ larger geometric adaptation). Homophily $\mathcal{H}$ is also shown in Table 1.

scaling, enabling richer geometric representations. The additional $\frac{d(d-1)}{2}$ off-diagonal parameters per node would allow modeling of correlated feature dimensions and rotational geometry.

- **Computational Complexity**: However, full tensors incur $O(d^3)$ complexity for metric operations and $O(|\mathcal{V}|d^2)$ memory overhead. For typical GNN dimensions ($d = 128$), this represents a 128× increase in metric computation cost. Low-rank factorizations $\mathbf{G}_i = \mathbf{L}_i\mathbf{L}_i^T$ with $\mathbf{L}_i \in \mathbb{R}^{d\times r}$ offer a middle ground but still require $O(dr^2)$ operations.
- **Optimization Challenges**: Non-diagonal metrics cannot leverage our efficient conformal geometry framework (Section 4), requiring expensive SPD manifold projections and Riemannian optimization. The increased parameter space also exacerbates overfitting on small graphs.

We view exploring full or low-rank metric tensors as valuable future work, particularly for applications requiring fine-grained geometric modeling. However, this extension lies beyond the current scope, as our diagonal approach already achieves state-of-the-art performance while maintaining computational efficiency comparable to standard GNNs.

F.2 Statements of Broader Impact

This work introduced ARGNN, a novel framework that learns continuous, anisotropic metric tensor fields to capture the geometric diversity inherent in real-world graphs. This advancement possibly has broader impacts for scientific and societal benefits, such as enhancing community network analysis and advancing biomedical research by its node and link prediction power. Apart from offering possible scientific and societal benefits, to the best of our knowledge, our research works do not involve ethical constraints or religious and political restrictions.