TOWARDS DOWNSTREAM TUNING-FREE GRAPH FOUNDATION MODELS WITH MANIFOLD-BASED STRUCTURAL ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph foundation models (GFMs) demonstrate strong potential in cross-domain graph learning by pre-training on multiple source domains and adapting to unseen target domains. Many existing GFMs are generally prompt-based and address graph domain gaps through downstream tuning of additional prompt embeddings, and therefore focus primarily on representation-level adaptation under a Euclidean geometric assumption. However, real-world graphs often share transferable structural patterns across domains, such as tree-like and cyclic substructures, that are naturally non-Euclidean and thus cannot be sufficiently exploited by existing Euclidean-based prompt-centric GFMs. To address this issue, this paper presents GFManifold, a graph structure-centric framework that adopts hyperbolic and spherical Riemannian manifold branches with learnable curvatures to learn manifold-induced structural relations across source domains for efficient parameter tuning-free GFM adaptation. During pre-training, GFManifold learns cross-domain graph structural patterns through domain alignment and geodesic link prediction across diverse source-domain graphs. During downstream adaptation, GFManifold exploits manifold-induced geodesic relations to adapt the target graph structure for downstream tasks, enabling efficient target-domain parameter tuning-free adaptation and avoiding downstream gradient-based parameter updates in most existing GFMs. Extensive experiments on 14 benchmarks showcase its effectiveness and efficiency on cross-domain graph adaptation tasks. Code is available upon acceptance.

1 INTRODUCTION

Graph foundation models (GFMs) have significantly advanced cross-domain graph learning by pre-training on multiple source domains and adapting to unseen target domains (Liu et al., 2023a; Li et al., 2024; Chen et al., 2024a; Xia et al., 2024; Chen et al., 2024b; Kong et al., 2025; Zhao et al., 2024; Sun et al., 2025b; Yu et al., 2024b; 2025; Wang et al., 2025). To achieve cross-domain generalisability, existing GFMs usually address domain gaps through fine-tuning additional prompts or embeddings (Yu et al., 2024b; 2025; Wang et al., 2025). For example, SAMGPT (Yu et al., 2025) designs a structure prompt for each source domain and fine-tunes the prompt on the target domain to inject structural information into node embeddings. MDGFM (Wang et al., 2025) introduces a domain prompt to align source-domain graphs, which is subsequently fine-tuned on target domains. Despite their progress, the main GFM mechanism for downstream adaptation remains prompt- or embedding-centric, neglecting the importance of graph data. Graph structural information is mainly encoded by

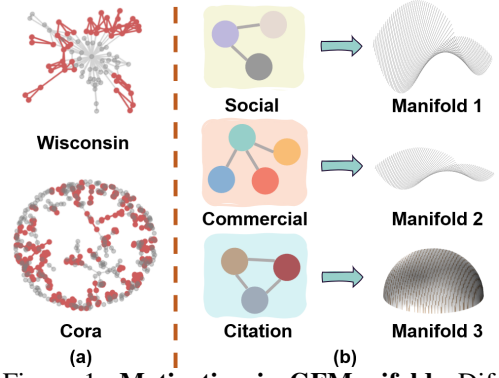


Figure 1: **Motivation in GFManifold.** Different real-world domain graphs, such as Wisconsin (university webpage hyperlink network) and Cora (academic citation network), naturally exhibit diverse non-Euclidean structural tendencies (as in (a)), which can be modelled through complementary hyperbolic and spherical manifold branches (as in (b)). Red indicates similar structural motifs (local subgraph patterns) that commonly exist in both domains.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

additional learnable vectors and adapted through extensive target-domain fine-tuning. Therefore, the graph structural data itself is not explicitly adapted during downstream adaptation, leaving the rich source and target domain graph structure insufficiently explored in existing GFMs.

Graph structure plays an important role in GFM cross-domain generalisation. In real-world applications, cross-domain graphs often share transferable structural patterns, such as hierarchical trees, cycles, cliques, and other motifs, even when their node attributes and domain semantics differ substantially. Such patterns are naturally related to non-Euclidean geometries and cannot always be faithfully represented under a purely Euclidean assumption (Sun et al., 2025b; Xiong et al., 2022; Gu et al., 2019; de Ocáriz Borde et al., 2023; Wang et al., 2024). For example, in Figure 1 (a), subgraphs with identical structures from two domains, i.e., the Wisconsin webpage hyperlink network and the Cora academic citation network, are highlighted in red. The prevalence of these shared structures indicates that GFM cross-domain transfer can benefit from capturing these shared graph structural patterns rather than merely relying on fine-tuning additional target-domain specific embeddings or prompt parameters. Therefore, a natural question arises:

Table 1: **Comparing existing GFMs.** Most GFMs require additional downstream **prompt tuning** for each new target domain. Existing manifold-based GFMs generally use non-learnable fixed curvature. In contrast, GFManifold is a downstream target tuning-free method with learnable curvature.

Method	Tuning	Prompt	Curvature
GCOPE (Zhao et al., 2024)	✓	✗	-
MDGPT (Yu et al., 2024b)	✓	✓	-
SAMGPT (Yu et al., 2025)	✓	✓	-
MDGFM (Wang et al., 2025)	✓	✓	-
RiemannGFM (Sun et al., 2025b)	✓	✗	Fixed
GraphGlue (Sun et al., 2026a)	✓	✓	Induced
GFManifold	✗	✗	Learnable

Can we exploit such non-Euclidean graph structures for parameter tuning-free GFM adaptation?

A straightforward approach is to directly use non-Euclidean Riemannian manifolds, such as hyperbolic (Chami et al., 2019; Ganea et al., 2018; Nickel & Kiela, 2018; Law, 2021) or spherical (Meng et al., 2019; de Ocáriz Borde et al., 2023; Gu et al., 2019) manifolds, which are widely adopted to model tree-like and cyclic structures (Xiong et al., 2022; Sun et al., 2024b;c; Bachmann et al., 2020; Ye et al., 2023; Sun et al., 2024a). However, directly using manifold embeddings as downstream representations is not sufficient for cross-domain GFM adaptation. On the one hand, graphs from different domains may exhibit different non-Euclidean structural tendencies, as illustrated in Figure 1 (b). A citation network with a predominantly hierarchical structure can be better modelled by hyperbolic geometry, whereas a cyclic or clique-rich graph may be more compatible with spherical geometry. Hence, a single fixed pre-defined manifold may be insufficient for capturing diverse graph structures across domains. This motivates complementary hyperbolic and spherical manifold branches, which provide distinct non-Euclidean views of graph structural relations instead of assuming that a single geometry is suitable for all domain graphs. On the other hand, directly replacing graph features or downstream representations with manifold embeddings may distort the original graph information and remove useful target-domain signals. Thus, a conservative graph structural adaptation strategy that adjusts target graph structural weights without reconstructing the graph from scratch is desirable.

In light of the above discussion, we present **GFManifold**, a manifold-based GFM framework for tuning-free graph structural adaptation. During pre-training, hyperbolic and spherical constant-curvature Riemannian manifold branches with learnable curvatures are jointly trained with a GNN encoder on multiple source domains through domain alignment and geodesic link prediction to capture cross-domain transferable graph structural patterns. GFManifold does not simply adopt manifold embeddings for final downstream representations. Instead, the learnt manifold branches provide geodesic relations that guide target graph structural adaptation. During downstream adaptation, GFManifold keeps the pre-trained GNN fixed and transforms the target graph into a geodesic-weighted graph by reweighting edges according to manifold-induced structural similarity. The adapted graph is then encoded by the fixed pre-trained model for downstream tasks, without requiring further prompt, classifier, or model fine-tuning. This design is fundamentally different from existing prompt-centric GFMs (Wang et al., 2025; Yuan et al., 2025; Yu et al., 2025) that rely on representation-level tuning, and also differs from existing Riemannian graph methods (Sun et al., 2025b; 2026a) that mainly operate on manifold embeddings and require downstream tuning, as summarised in Table 1. Experiments on 14 benchmarks validate its effectiveness and efficiency. Our main contributions are:

- **Structure-centric GFManifold**: A novel structure-centric GFM framework is proposed to learn manifold-induced structural relations across source domains. Unlike most existing GFMs, GFManifold exploits manifold-induced geodesic relations for target graph structural adaptation.

- **Target-domain parameter tuning-free adaptation**: Unlike most prompt-based and Riemannian GFMs relying on parameter fine-tuning for target domain adaptation, GFManifold enables parameter tuning-free GFM adaptation by adapting the graph structure from a data centric aspect.
- **Theoretical and empirical validation**: Theoretical analysis characterises cross-domain generalisation under a decomposition into shared and domain-specific structural patterns, while extensive experiments on 14 benchmark datasets validate the effectiveness and efficiency of GFManifold.

2 Preliminary: Riemannian Manifold

A Riemannian manifold $(\mathcal{M}, d)$ is a smooth manifold $\mathcal{M}$ with a metric d that defines distances in a non-Euclidean space. At each point $\boldsymbol{u} \in \mathcal{M}$, the tangent space $T_{\boldsymbol{u}}\mathcal{M}$ is locally Euclidean. The sectional curvature κ characterises how $\mathcal{M}$ deviates from the Euclidean space $\mathbb{R}^d$. A *Constant Curvature Space* (CCS) (Bachmann et al., 2020) is a Riemannian manifold $\mathcal{M}_\kappa$ with constant sectional curvature κ, admitting closed-form distance metrics. Here, **constant curvature refers to curvature being spatially uniform at every point within the same manifold**. This is compatible with learnable curvature in GFManifold: the scalar value of κ is learnt from multi-domain graphs, while each hyperbolic or spherical manifold branch remains a constant-curvature space rather than a spatially varying-curvature manifold. The learnt curvature values parametrise these predefined model spaces and are not intended as estimates of the intrinsic curvature of any individual graph.

In real-world applications, hyperbolic spaces ($\mathbb{H}^d, \kappa < 0$) are commonly used to model tree-like graphs (Chami et al., 2019; Ganea et al., 2018; Nickel & Kiela, 2018; Sun et al., 2025b), spherical spaces ($\mathbb{S}^d, \kappa > 0$) are used to model cyclic or ring-like graphs (Sun et al., 2025b; de Ocáriz Borde et al., 2023; Meng et al., 2019), and Euclidean spaces ($\mathbb{R}^d, \kappa = 0$) for the special flat case. For $\kappa \neq 0$, a d-dimensional constant curvature Riemannian manifold is represented in the ambient space $\mathbb{R}^{d+1}$:

$$\mathcal{M}_\kappa = \left\{ \boldsymbol{u} = (u_t, u_s) \in \mathbb{R}^{d+1} \;\middle|\; \langle \boldsymbol{u}, \boldsymbol{u} \rangle_\kappa = \tfrac{1}{\kappa},\ u_s \in \mathbb{R}^d \right\}, \tag{1}$$

where u_t denotes the first ambient coordinate and $u_s \in \mathbb{R}^d$ denotes the remaining spatial components. For the hyperbolic branch, we additionally restrict $u_t > 0$ to select the upper sheet of the hyperboloid, corresponding to the Lorentz model. The curvature-dependent ambient inner product is defined as:

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_\kappa := \operatorname{sgn}(\kappa)\, u_t v_t + u_s^\top v_s, \quad \boldsymbol{u}, \boldsymbol{v} \in \mathcal{M}_\kappa, \tag{2}$$

which allows computing infinitesimal distances in the tangent space $T_{\boldsymbol{u}}\mathcal{M}_\kappa$. The geodesic distance $d_\kappa(\boldsymbol{u}, \boldsymbol{v})$ is obtained by integrating the distances along the shortest curve connecting $\boldsymbol{u}$ and $\boldsymbol{v}$ on the manifold, and depends explicitly on the value range of the sectional curvature κ, defined as:

$$d_\kappa(\boldsymbol{u}, \boldsymbol{v}) = \begin{cases} \frac{1}{\sqrt{-\kappa}} \operatorname{arccosh}\Big(\kappa \langle \boldsymbol{u}, \boldsymbol{v} \rangle_\kappa\Big), & \kappa < 0 \text{ (Hyperbolic } \mathbb{H}), \\ \|\boldsymbol{u} - \boldsymbol{v}\|_2, & \kappa = 0 \text{ (Euclidean } \mathbb{R}), \\ \frac{1}{\sqrt{\kappa}} \arccos\Big(\kappa \langle \boldsymbol{u}, \boldsymbol{v} \rangle_\kappa\Big), & \kappa > 0 \text{ (Spherical } \mathbb{S}). \end{cases} \tag{3}$$

where $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_\kappa$ is the curvature-aware inner product in the ambient space $\mathbb{R}^{d+1}$ and the $1/\sqrt{|\kappa|}$ factors are scaling induced by the curvature, ensuring geodesic distances are geometrically meaningful within their CCS types. For $\kappa \neq 0$, the manifold origin ("north pole") is defined as:

$$\boldsymbol{o} = \left[\sqrt{|1/\kappa|}, 0, \ldots, 0\right]^\top \in \mathcal{M}_\kappa. \tag{4}$$

The tangent space of the manifold origin, defined by $T_{\boldsymbol{o}}\mathcal{M}_\kappa$, is locally flat and Euclidean, providing a linear approximation. Vectors in this Euclidean space can be mapped to the non-Euclidean manifold via exponential maps and back via logarithmic maps as defined in Appendix C.

3 GFManifold: Tuning-Free Structural Adaptation by Manifold

A graph $\mathcal{G}$ is defined by an adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{N \times N}$ (or edge set $\mathcal{E}$) and a feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$. N is the number of nodes and d is the feature dimension. A **graph foundation model (GFM)** is pre-trained on graphs from K source domains $\mathcal{S}$ and adapts to an unseen target domain $\mathcal{T}$. This work focuses on GNN-based GFMs following most GFMs (Zhao et al., 2024; Yu et al., 2024b; 2025; Wang et al., 2025; Sun et al., 2025b; 2026a). **LLM-based ones are not the focus** of this paper since they are limited to text-attributed graphs. The overall framework in Figure 2 comprises two key stages: (i) pre-training on multi-domain and (ii) target domain parameter tuning-free adaptation.

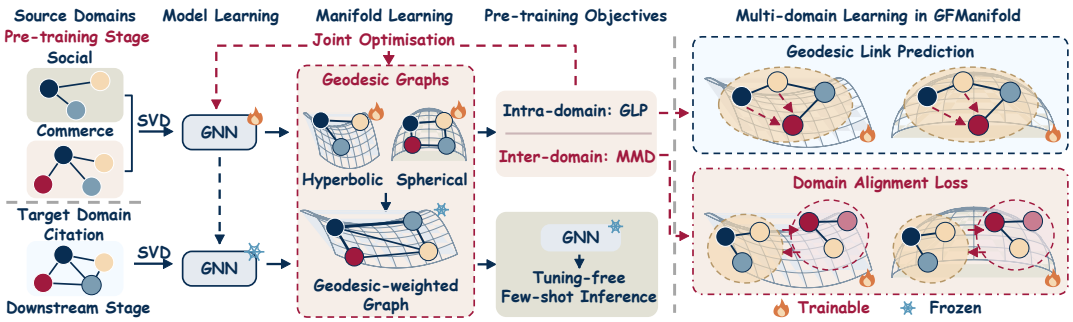


Figure 2: **Overall framework of GFManifold.** (i) During pre-training, spherical and hyperbolic manifold branches with learnable curvatures are jointly trained with a GNN encoder to learn manifold-induced cross-domain structural relations shared across source domains. (ii) During downstream, the target-domain graph is reweighted using the manifold-induced geodesic relations, producing a geodesic-weighted graph that serves as input for few-shot inference without any fine-tuning.

3.1 Multi-domain Pre-training of GFManifold

Feature Alignment and Encoding Process Multi-domain graph pre-training is challenging due to feature and structural misalignment across domains. We follow prior GFMs (Zhao et al., 2024; Yu et al., 2025) to adopt SVD preprocessing to standardise and align node feature dimensionality before GNN encoding. Specifically, let $\mathcal{G}_i = (\boldsymbol{X}, \boldsymbol{A})$ denote a feature-aligned graph with N_i nodes from the i-th domain. A GNN encoder, implemented by default as a GCN following most existing GFMs (Yu et al., 2025; 2024b), with parameters θ encodes it as $\boldsymbol{H}_i = \text{GNN}_\theta(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N_i \times d}$. These embeddings serve as shared tangent representations for the subsequent curvature-specific manifold mappings. GFManifold then models the representations into both hyperbolic and spherical branches and jointly optimises their structural relations across domains. The pre-training stage consists of: (i) Manifold Branch Learning and (ii) Multi-domain Learning.

Cross-domain Learnable Curved Manifold Branches GFManifold adopts complementary hyperbolic and spherical Riemannian branches to model different kinds of non-Euclidean structural tendencies, while learnable curvatures can adapt their geometries during multi-domain pre-training. Note that the curvatures are globally shared across all source domains rather than graph or domain-specific. Yet, Riemannian manifolds with different curvature values induce distinct geometric spaces: positive curvature yields spherical geometry, while negative curvature yields hyperbolic geometry. Using a single learnable curvature parameter for both geometric types can lead to gradient instability, since the geodesic distance changes non-smoothly near zero curvature, as shown in Equation 3.

To address this, GFManifold employs two complementary manifold branches, collectively denoted by $\mathcal{M}_\kappa$: a hyperbolic manifold $\mathbb{H}_\kappa$ and a spherical manifold $\mathbb{S}_\kappa$, each with its own globally shared learnable scalar curvature parameter. These are two distinct global scalars, one for each branch, and each is shared across all source graphs and domains and constant within its corresponding branch. Their signs are constrained during optimisation as detailed in Appendix B. For compactness, κ below denotes the corresponding branch curvature. Here, $\mathcal{M}_\kappa$ is used only as collective notation for the two manifold branches and does not denote a newly constructed or product manifold. Let a manifold embedding matrix be defined as: $\boldsymbol{H}_i^{\mathcal{M}_\kappa} = [\boldsymbol{H}_{i,1}^{\mathcal{M}_\kappa}, \ldots, \boldsymbol{H}_{i,N_i}^{\mathcal{M}_\kappa}]^\top \in \mathbb{R}^{N_i \times (d+1)}$ for N_i nodes in the i-th domain and curvature κ. $\boldsymbol{H}_i^{\mathcal{M}_\kappa} = \boldsymbol{H}_i^{\mathbb{S}_\kappa}$ for $\kappa > 0$, and $\boldsymbol{H}_i^{\mathcal{M}_\kappa} = \boldsymbol{H}_i^{\mathbb{H}_\kappa}$ for $\kappa < 0$. The j-th node embedding produced by the GNN is mapped onto the tangent space at the manifold origin $\boldsymbol{o}$ and then projected onto the corresponding manifold branch via the Riemannian exponential map:

$$\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa} = \begin{cases} \boldsymbol{H}_{i,j}^{\mathbb{S}_\kappa} = \cos\left(\sqrt{\kappa}\,\|\boldsymbol{H}_{i,j}\|\right)\boldsymbol{o} + \dfrac{\sin\left(\sqrt{\kappa}\,\|\boldsymbol{H}_{i,j}\|\right)}{\sqrt{\kappa}\,\|\boldsymbol{H}_{i,j}\|}\,\boldsymbol{H}_{i,j}, & \kappa > 0, \\ \boldsymbol{H}_{i,j}^{\mathbb{H}_\kappa} = \cosh\left(\sqrt{-\kappa}\,\|\boldsymbol{H}_{i,j}\|\right)\boldsymbol{o} + \dfrac{\sinh\left(\sqrt{-\kappa}\,\|\boldsymbol{H}_{i,j}\|\right)}{\sqrt{-\kappa}\,\|\boldsymbol{H}_{i,j}\|}\,\boldsymbol{H}_{i,j}, & \kappa < 0. \end{cases} \tag{5}$$

where $\boldsymbol{H}_{i,j}^{\mathbb{S}_\kappa} \in \mathbb{R}^{d+1}$ and $\boldsymbol{H}_{i,j}^{\mathbb{H}_\kappa} \in \mathbb{R}^{d+1}$ denote the j-th node embedding vector in the i-th domain graph on the spherical and hyperbolic manifold branches. d is the intrinsic manifold dimension, while manifold points are represented in the ambient space $\mathbb{R}^{d+1}$. $\|\cdot\|$ denotes the standard Euclidean norm. $\boldsymbol{H}_{i,j}$ is zero-padded to $(0, \boldsymbol{H}_{i,j})^\top$ prior to the exponential map (de Ocáriz Borde et al., 2023).

Cross-domain Graph Learning. Cross-domain learning requires effective modelling of both inter-domain and intra-domain knowledge. (i) At the inter-domain level, different domain distributions are

aligned within the hyperbolic and spherical manifolds. (ii) At the intra-domain level, manifold-based link prediction models structural relations by encouraging connected nodes to be close.

(i) Multi-domain Alignment via Manifold Branches. Multi-domain learning is challenging because graphs from different domains naturally exhibit distinct distributions. To facilitate multi-domain graph learning, a Maximum Mean Discrepancy (MMD) objective is adopted to align multi-domain distributions. To combine the complementary structural relations from two manifold branches, GFManifold integrates their geodesic relations through a composite manifold-induced dissimilarity measure. For clarity, this measurement does not construct a product manifold or define the geodesic distance of a new manifold. Instead, both spherical and hyperbolic node embeddings are derived from the same Euclidean tangent representation in Equation 5 through curvature-specific exponential maps. The geodesic distance is first computed within its corresponding constant-curvature manifold, and the manifold-induced structural relations are then aggregated as a unified structural dissimilarity:

$$d_\kappa(\boldsymbol{u}, \boldsymbol{v}) = \alpha_0 \, d_{\mathbb{H}_\kappa}(\boldsymbol{u}^{\mathbb{H}}, \boldsymbol{v}^{\mathbb{H}}) + \alpha_1 \, d_{\mathbb{S}_\kappa}(\boldsymbol{u}^{\mathbb{S}}, \boldsymbol{v}^{\mathbb{S}}), \tag{6}$$

where α_0 and α_1 are learnable weighting parameters. $d_{\mathbb{H}_\kappa}(\cdot,\cdot)$ and $d_{\mathbb{S}_\kappa}(\cdot,\cdot)$ denote the geodesic distance functions within the hyperbolic and spherical spaces as in Equation 3. For each node, $\boldsymbol{u}$ denotes the paired branch representations $(\boldsymbol{u}^{\mathbb{H}}, \boldsymbol{u}^{\mathbb{S}})$. $\boldsymbol{u}^{\mathbb{H}}$ and $\boldsymbol{v}^{\mathbb{H}}$ are the hyperbolic embeddings of two nodes, while $\boldsymbol{u}^{\mathbb{S}}$ and $\boldsymbol{v}^{\mathbb{S}}$ are their spherical counterparts. Intuitively, $d_\kappa(\boldsymbol{u}, \boldsymbol{v})$ represents a composite manifold-induced structural dissimilarity between two nodes, rather than the geodesic distance of a newly constructed manifold. Based on this composite relation, a kernel function is defined as:

$$k_\kappa(\boldsymbol{u}, \boldsymbol{v}) = \exp\Big(-\frac{d_\kappa(\boldsymbol{u}, \boldsymbol{v})^2}{2b^2}\Big), \tag{7}$$

where b denotes the bandwidth. To align source-domain representation distributions, the domain with the smallest average composite manifold-induced dissimilarity among connected nodes is selected as the pivot domain p, defined as: $p = \arg\min_{i\in\{1,\dots,K\}} \frac{1}{|\mathcal{E}_i|} \sum_{(\boldsymbol{u},\boldsymbol{v})\in\mathcal{E}_i} d_\kappa(\boldsymbol{u}, \boldsymbol{v})$, where $\mathcal{E}_i$ denotes the edge set in domain i, $d_\kappa(\boldsymbol{u}, \boldsymbol{v})$ is the composite manifold-induced dissimilarity, and K is the number of source domains. The MMD loss aligns each source domain $i \neq p$ with the pivot domain p:

$$\mathcal{L}_{\text{MMD}} = \sum_{i\in\{1,\dots,K\}\setminus p} \left[\frac{1}{N_i^2} \sum_{j,j'=1}^{N_i} k_\kappa(\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa}, \boldsymbol{H}_{i,j'}^{\mathcal{M}_\kappa}) + \frac{1}{N_p^2} \sum_{k,k'=1}^{N_p} k_\kappa(\boldsymbol{H}_{p,k}^{\mathcal{M}_\kappa}, \boldsymbol{H}_{p,k'}^{\mathcal{M}_\kappa}) - \frac{2}{N_i N_p} \sum_{j=1}^{N_i} \sum_{k=1}^{N_p} k_\kappa(\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa}, \boldsymbol{H}_{p,k}^{\mathcal{M}_\kappa}) \right], \tag{8}$$

where $\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa}$ is the j-th node embedding in the i-th domain and $\boldsymbol{H}_{p,k}^{\mathcal{M}_\kappa}$ is the k-th node embedding in the pivot domain. N_i and N_p are the numbers of nodes in domains i and p, respectively. Intuitively, the MMD loss encourages alignment between source-domain representation distributions under the kernel induced by the composite manifold relations. To reduce computational overhead, the same number of nodes is uniformly sampled from each source domain.

(ii) Geodesic-based Link Prediction via Manifold Branches (GLP). To learn intra-domain graph structural relations from source domains, a manifold-based link prediction objective is proposed to encourage connected node pairs to have small composite manifold-induced dissimilarities:

$$\mathcal{L}_{\text{GLP}} = -\sum\nolimits_{i=1}^{K} \sum\nolimits_{(j,k)\in\mathcal{E}_i} \log \frac{\exp(-d_\kappa(\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa}, \boldsymbol{H}_{i,k}^{\mathcal{M}_\kappa})/\tau)}{\sum_{l\in\mathcal{C}_i(j)} \exp(-d_\kappa(\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa}, \boldsymbol{H}_{i,l}^{\mathcal{M}_\kappa})/\tau)}, \tag{9}$$

where $\mathcal{E}_i$ denotes the set of connected nodes in the i-th domain, τ is the temperature, and $\mathcal{C}_i(j)$ contains the positive neighbour and uniformly sampled negative nodes following prior methods (Yu et al., 2024b; Chen et al., 2025b). $\boldsymbol{H}_{i,j}^{\mathcal{M}_\kappa}$ and $\boldsymbol{H}_{i,k}^{\mathcal{M}_\kappa}$ are the j-th and k-th rows of the manifold embedding matrix for domain i.

Overall GFManifold Pre-training Objective. By leveraging both manifold-based link prediction and domain alignment, the GNN encoder and the two manifold branches are jointly optimised across multiple source-domain graphs. The final learning objective is defined as a weighted combination:

$$\mathcal{L}_{\text{GFManifold}} = \lambda \, \mathcal{L}_{\text{MMD}} + (1 - \lambda) \, \mathcal{L}_{\text{GLP}}, \tag{10}$$

where λ is a hyperparameter controlling the relative contributions of the two loss terms and is determined on the validation set. This objective encourages alignment between source-domain representation distributions while preserving connected-node structural relations.

3.2 Parameter Tuning-free Downstream Adaptation in GFManifold

During downstream stage, GFManifold enables efficient parameter tuning-free adaptation without incurring any model or prompt fine-tuning cost. This differs from prior GFMs that generally adopt model or prompt-centric approaches, such as fine-tuning learnable prompts to capture graph structures.

Manifold-induced Graph Structural Adaptation Leveraging the learnt manifold-induced relations, GFManifold adapts the target-domain graph structure by edge reweighting. For a target graph $\mathcal{G}_{Tar} = (\boldsymbol{X}, \boldsymbol{A})$ with N nodes, the pre-trained GNN encodes first latent embeddings as $\boldsymbol{H}_{Tar} = \text{GNN}_\theta(\boldsymbol{X}, \boldsymbol{A})$, and the manifold embeddings $\boldsymbol{H}_{\text{Tar}}^{\mathcal{M}_\kappa}$ can be obtained as in Equation 5. Then, a geodesic-weighted adjacency matrix $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ can be defined by the composite manifold-induced dissimilarity. Each evaluated entry is scaled to $[0, 1]$ and represents a normalised structural similarity weight between nodes $\boldsymbol{u}$ and $\boldsymbol{v}$: $\boldsymbol{S}_{\boldsymbol{u},\boldsymbol{v}} = \frac{1-d_\kappa(\boldsymbol{u},\boldsymbol{v})-\min(1-d_\kappa)}{\max(1-d_\kappa)-\min(1-d_\kappa)}$, where $d_\kappa(\boldsymbol{u}, \boldsymbol{v})$ is the composite manifold-induced dissimilarity defined in Equation 6, and $\min(1 - d_\kappa)$ and $\max(1 - d_\kappa)$ are computed over the evaluated node pairs $\boldsymbol{u}', \boldsymbol{v}' \in \boldsymbol{H}_{\text{Tar}}^{\mathcal{M}_\kappa}$.

To adapt the graph structure, a straightforward approach is to construct a dense graph from all pairwise manifold-induced relations. However, this may distort the original topology and remove valuable structural information. In addition, computing a full pairwise weighted adjacency matrix incurs a complexity of $\mathcal{O}(N^2)$, which becomes prohibitive for large-scale graphs. To address this issue, GFManifold constrains the weight computation to existing edges, reducing the complexity to $\mathcal{O}(E)$, where E is the number of edges. This strategy operates on existing graph edges while using manifold-induced similarities as edge weights. The resulting geodesic-weighted graph is defined as $\mathcal{G}_{\text{Geo}} = (\boldsymbol{X}, \boldsymbol{S})$ and serves as input to the pre-trained model for encoding test node embeddings.

Downstream Few-shot Inference GFManifold performs downstream classification tasks following prior work (Zhao et al., 2024; Yu et al., 2025; Wang et al., 2025). The classification is performed in a parameter tuning-free manner without updating the parameters of the pre-trained GNN by comparing the similarity between test-node embeddings and the mean few-shot embeddings (Sun et al., 2022b; Chen et al., 2025b; Yu et al., 2024a). Given a geodesic-weighted graph $\mathcal{G}_{\text{Geo}} = (\boldsymbol{X}, \boldsymbol{S})$, the pre-trained GNN encodes embeddings as $\boldsymbol{H} = \text{GNN}_\theta(\boldsymbol{X}, \boldsymbol{S})$. Let $\mathcal{H}_c$ denote the support embeddings of class $c \in \mathcal{C}$, and let $\boldsymbol{h}$ denote a test-node embedding. The predicted label $\hat{y}$ is obtained by maximising the cosine similarity between $\boldsymbol{h}$ and the mean support embedding $\bar{\boldsymbol{h}}_c = \frac{1}{|\mathcal{H}_c|} \sum_{\boldsymbol{h}_c \in \mathcal{H}_c} \boldsymbol{h}_c$, calculated as $\hat{y} = \text{argmax}_{c \in \mathcal{C}} \cos(\boldsymbol{h}, \bar{\boldsymbol{h}}_c)$. As GFManifold enables data-centric downstream adaptation by reweighting target graphs using source-learnt manifold-induced structural relations, the pre-trained model does not require additional fine-tuning or adaptation. **The entire downstream process requires no target-domain parameter optimisation, improving efficiency compared with existing GFMs that rely on fine-tuning either prompts or the model during downstream adaptation.** We provide a detailed efficiency analysis in Section 5.5 and a complexity analysis in Appendix B.

4 Theoretical Insights

Definition 1: Shared and Domain Specific Patterns in Graphs. Following prior graph invariant learning and multi-domain graph generalisation studies (Li et al., 2022; Yuan et al., 2025), we assume that graphs from multiple domains comprise cross-domain shared and domain-specific structural patterns. Let Φ_{shared} and Φ_{specific} denote conceptual mappings for shared and domain-specific structural information, where $\Phi_{\text{shared}}(\mathcal{G})$ captures generalisable structural patterns shared across domains, while $\Phi_{\text{specific}}(\mathcal{G})$ represents domain-specific variations.

In cross-domain scenarios, shared structural patterns may be neglected or inadequately captured by existing GFMs (Zhao et al., 2024; Yu et al., 2024b; 2025), limiting their generalisability. GFManifold is designed to exploit such manifold-induced relations that reflect cross-domain transferable structural patterns in the target-domain graph, thereby supporting effective cross-domain generalisation.

Definition 2: Cross-domain Graph Structural Adaptation. We define GFManifold as a structure-centric framework for cross-domain GFM adaptation. Let f_{θ^*} denote the source-learnt GNN encoder and $\mathcal{M}_{\kappa^*}$ denote the source-learnt manifold-induced structural transformation, where θ^* and κ^* are learnt during source-domain pre-training and remain fixed during downstream adaptation. Given an unseen target-domain graph $\mathcal{G}$, GFManifold first obtains the downstream representations as $\boldsymbol{H} = f_{\theta^*}\big(\mathcal{M}_{\kappa^*}(\mathcal{G})\big)$, where $\mathcal{M}_{\kappa^*}(\mathcal{G})$ denotes the geodesic-weighted graph by reweighting the existing edges of $\mathcal{G}$. For a test representation $\boldsymbol{h} \in \boldsymbol{H}$, GFManifold predictor g outputs:

$$g(\boldsymbol{h}) = \hat{y} = \text{argmax}_{c \in \mathcal{C}} \cos\big(\boldsymbol{h}, \bar{\boldsymbol{h}}_c\big), \tag{11}$$

where $\bar{\boldsymbol{h}}_c$ is the mean few-shot embedding from class c. No prompt or model parameters are updated during downstream. Following prior GFMs, for the conditional analysis, we introduce a shared structural component $\Phi_{\text{shared}}(\mathcal{G})$ under the assumptions detailed in Appendix E.

Given a pre-trained GNN encoder f_θ and manifold branches $\mathcal{M}_\kappa$, the cross-domain generalisation of GFManifold can be characterised as follows:

Theorem 4.1 (**Conditional Cross-domain Risk Objective**). *Consider the GFManifold graph predictor $g_{\theta,\kappa}(\mathcal{G}) = f_\theta(\mathcal{M}_\kappa(\mathcal{G}))$. Its optimal cross-domain parameters are characterised by:*

$$(\theta^*, \kappa^*) \in \arg\min_{\theta,\kappa} \sup_{i \in \mathcal{S} \cup \mathcal{T}} R(g_{\theta,\kappa} \mid i), \tag{12}$$

where $R(g_{\theta,\kappa} \mid i)$ denotes the expected risk on domain i.

Detailed proof with the adopted assumptions from prior GFMs are in Appendix E due to page limit.

5 Experiments

Datasets. 14 benchmarks from diverse domains are employed, including large-scale datasets, as in Table 4 and Table 5. These include: (i) **Social Networks**: Cornell, Texas, Wisconsin, Chameleon, and Squirrel, where nodes represent individuals or web pages and edges denote social or hyperlink connections; (ii) **Citation Networks**: Cora, Citeseer, PubMed, and Arxiv-year, where nodes are papers and edges indicate citation; (iii) **Commercial Systems**: Amazon-Photo and Computers, where nodes are products and edges are co-purchasing relations; (iv) **Biological Networks**: BZR, COX2, and PROTEINS, where nodes are biological entities and edges represent molecular interactions.

Baselines. 14 baselines are compared with GFManifold and can be categorised into: (i) Single-domain supervised learning **(SL)**, such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), SAGE (Hamilton et al., 2017), H2GCN (Zhu et al., 2020a), and GPR (Chien et al., 2021); (ii) Single-domain self-supervised pre-training and fine-tuning **(SSL+FT)**, including Link Prediction (LP) (Lu et al., 2021), DGI (Velickovic et al., 2019), and GCL (You et al., 2020), where fine-tuning (FT) adapts the model on the target graph using few-shot samples; (iii) GFM Methods by multi-domain self-supervised pre-training and cross-domain adaptation. GFMs that rely on fine-tuning on target domain datasets **(GFM+FT)** include GCOPE (Zhao et al., 2024), MDGFM (Wang et al., 2025), MDGPT (Yu et al., 2024b), SAMGPT (Yu et al., 2025), RiemannGFM (Sun et al., 2025b) and GraphGlue (Sun et al., 2026a). Unlike the evaluated **GFM+FT** baselines, GFManifold requires no downstream parameter updates.

Implementation. Following most GFMs (Yu et al., 2024b; 2025), GCN is the default backbone for all methods except for RiemannGFM and SL methods, which employ their specific backbones. **Analysis on more backbones is in Appendix F.4.** Following prior GFM settings (Zhao et al., 2024), for each target domain, after selecting few-shot samples for inference, the remaining data are randomly split 1:9 for validation and testing. Further implementation details are provided in Appendix B. Notably, GFMs perform **multi-domain pre-training on all domains except the one for target domain** evaluation, consistent with existing GFMs for a fair comparison (Zhao et al., 2024).

5.1 Overall Performance

Following the cross-domain evaluation settings in most GFMs (Yu et al., 2024b; 2025; Wang et al., 2025), the overall performance is evaluated on one-shot node classification (Table 2), graph classification (Table 6), and few-shot node classification (Figure 6). Our main findings are threefold:

(i) Our parameter tuning-free GFManifold significantly outperforms fine-tuning based baselines. GFManifold performs downstream adaptation without target-domain parameter updates, whereas the evaluated GFM baselines require downstream model, classifier, or prompt tuning. Among GFM methods, GFManifold consistently outperforms existing Euclidean-based GFMs through its learnable manifold branches and data-centric downstream adaptation. These results showcase the effectiveness of exploiting non-Euclidean, cross-domain transferable structure for downstream graph adaptation.

(ii) GFManifold consistently outperforms manifold-based GFMs such as RiemannGFM and GraphGlue. RiemannGFM operates on pre-defined fixed-curvature manifolds and therefore has limited flexibility to adapt its geometric relations across multiple source domains. Moreover, it relies on a fine-tuned classification head for downstream tasks. Another manifold GFM, GraphGlue, requires further training of additional prompt embeddings for each target domain, whereas GFManifold supports tuning-free downstream adaptation without model fine-tuning for each new target domain.

(iii) Furthermore, single-domain methods SL and SSL+FT cannot compete with GFMs, owing to GFMs' multi-domain pre-training. GFManifold is further evaluated on **graph classification**. As in

Table 2: **Cross-domain transfer learning performance of one-shot node classification.** Average accuracy (%) over five runs is reported. Results are highlighted by **best** and runner-up. "OOM" indicates out-of-memory on a 48 GB A6000 GPU. "–" indicates official code issues as in Appendix B. Results for graph classification and different few-shot are in Appendix F.1 and Appendix F.2.

	Methods	Texas	Cornell	Wisconsin	Squirrel	Chameleon	Citeseer	Computers	Pubmed	Cora	Photo	Arxiv-year
SL	GCN	38.82±9.79	24.58±10.09	43.36±13.17	19.96±5.89	24.30±6.59	33.39±11.86	23.96±6.20	24.08±12.17	29.85±8.98	47.09±5.81	18.64±6.91
	GAT	39.96±9.62	25.84±12.26	45.99±10.86	20.77±4.48	22.81±7.38	35.51±9.70	23.57±5.68	24.90±9.89	33.25±9.72	47.33±4.97	19.93±5.40
	SAGE	40.38±8.21	32.55±13.36	47.41±11.36	22.34±4.32	31.29±8.87	38.80±9.73	25.08±7.71	25.45±10.36	35.76±8.89	49.72±3.81	20.75±4.99
	GPR	37.60±9.54	30.77±13.42	49.56±15.58	21.06±6.14	28.44±6.65	29.77±13.22	20.93±8.47	22.35±13.30	38.99±15.77	43.39±4.66	10.81±9.94
	H2GCN	47.75±12.89	35.58±15.02	45.55±10.61	21.10±3.06	28.01±9.50	30.91±12.79	22.58±5.43	21.04±7.87	30.90±9.98	45.81±6.09	15.54±7.33
SSL+FT	LP+FT	36.93±11.90	23.88±11.43	41.37±14.22	20.04±5.61	25.34±7.19	34.92±12.08	29.08±9.62	46.99±8.10	35.59±9.74	48.82±6.55	17.94±6.06
	GCL+FT	28.81±15.32	20.56±13.36	35.70±17.73	19.97±5.62	24.69±8.82	36.05±13.44	24.01±10.82	46.72±9.98	33.27±9.50	47.33±5.89	15.53±8.89
	DGI+FT	34.46±12.92	22.89±11.32	38.89±15.77	20.85±6.57	25.78±7.34	33.96±11.57	26.17±13.05	45.96±10.40	32.38±8.86	45.17±7.34	18.06±6.22
GFM+FT	GCOPE	64.76±14.84	60.98±14.66	54.66±12.86	22.16±5.77	30.58±7.44	42.26±14.19	44.70±11.03	41.10±5.59	39.06±12.52	55.69±4.68	17.98±5.51
	MDGPT	59.76±12.44	59.14±13.08	50.40±15.07	24.41±7.01	28.04±4.28	41.98±12.24	48.98±11.39	49.76±8.18	44.52±11.39	54.96±10.25	OOM
	SAMGPT	66.79±10.77	59.34±9.82	52.29±14.40	25.75±6.29	38.12±8.90	47.76±10.55	48.02±7.68	49.89±11.04	52.83±12.04	56.33±9.04	OOM
	MDGFM	–	40.77±5.96	–	24.30±3.26	28.36±3.65	42.18±6.41	–	46.84±7.31	44.83±7.41	–	–
	RiemannGFM	58.60±15.27	46.35±11.92	47.32±16.58	20.13±8.58	29.68±9.95	38.02±9.58	40.29±9.51	44.50±5.37	37.91±16.13	49.69±13.32	OOM
	GraphGlue	39.94±11.60	25.47±5.30	29.73±9.58	22.38±1.05	28.67±3.91	31.33±5.06	47.79±14.52	41.40±3.79	38.86±5.57	56.85±8.29	OOM
	GFManifold	**72.92**±5.98	**64.97**±11.05	**61.26**±7.56	**34.17**±2.56	**41.12**±8.06	**62.53**±8.01	**51.72**±8.07	**52.59**±3.04	**61.71**±8.32	**63.27**±3.38	**26.99**±1.94

Table 6 in Appendix F.1, GFManifold outperforms GFMs with significantly lower standard deviations. This indicates that GFManifold can effectively transfer from learning on node classification to evaluating on graph classification, as investigated in Appendix F.1.

5.2 Ablation Study

The ablation study verifies the contribution of each key component in Figure 3. *w/o GeoGraph* represents GFManifold without geodesic-weighted graphs, where the target graph is instead weighted by GNN embedding similarity in Euclidean space. *w/o Manifold* refers to the baseline without non-Euclidean manifolds, which pre-trains a GNN and performs few-shot inference via cosine similarity. *w/o* κ denotes GFManifold with a fixed manifold parameter, specifically $\kappa = -1$ for hyperbolic and $\kappa = 1$ for spherical manifolds. Results show that **all components contribute positively to GFManifold and the non-Euclidean manifold component has the most significant impact.**

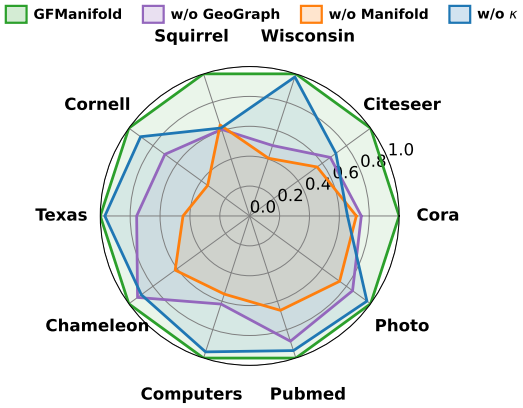


Figure 3: **Ablation study for GFManifold.**

5.3 Visualisation

t-SNE is employed to visualise the effect of the manifold component on the separability of test-node embeddings for classification. As shown in Figure 4, "original" indicates that the manifold component is removed, corresponding to "w/o manifold" in Section 5.2. The embeddings produced by GFManifold exhibit markedly clearer class separation than the original embeddings without the manifold component. These results suggest that source-learnt manifold-induced structural relations can support more separable downstream representations. In contrast, raw graphs may contain noisy and domain-specific patterns that hinder the class separability of the GNN-encoded embeddings. Additional results are provided in Appendix F.7.

5.4 Analysis of Cross-domain Graph Structural Relations

This section examines whether the geodesic edge reweighting in GFManifold reduces graph structural discrepancies between source and target graphs, using edge homophily (Jiang et al., 2026b) as a descriptive statistic of graph connectivity patterns. First, a multi-domain source pre-training graph is constructed by aggregating different source domain graphs from Table 4 except the one for target domain, yielding a weighted average edge homophily of 58.26% in the source domain graph set. The pre-trained GFManifold is subsequently evaluated on unseen target-domain graphs, as reported in Table 3. The results show that the geodesic-weighted target graphs exhibit smaller homophily gaps relative to the source graphs. **This reduction is accompanied by improved downstream accuracy, showcasing that source-learnt manifold-induced structural relations can support cross-domain graph adaptation.** In contrast, those GFM baselines do not explicitly perform data-centric reweighting of the target graph structure and therefore could not directly address source-target domain graph structural discrepancies through data-centric transformation.

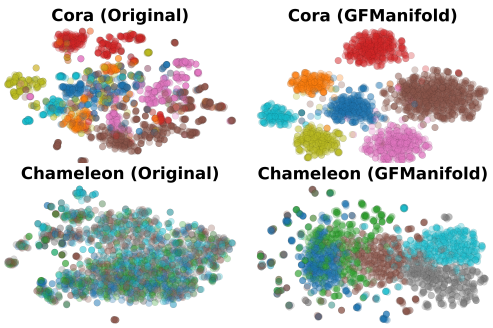


Figure 4: **t-SNE visualisation of embeddings**. Clearer class separation is observed for GFManifold. More results are provided in Appendix F.7.

Method	Chameleon	Wisconsin	Citeseer	Pubmed
Before Hom.	23.50	19.61	73.55	80.24
Homophily Gap	34.76	38.65	15.29	21.98
Before Accuracy	38.15	30.36	41.69	46.36
After Hom.	36.36	23.65	68.18	78.23
Homophily Gap	21.90 ↓	34.61 ↓	9.92 ↓	19.97 ↓
After Accuracy	41.12 ↑	61.26 ↑	62.53 ↑	52.59 ↑

Table 3: **Cross-domain Structural Relation Analysis.** Edge homophily (Hom.) (%) is used as a descriptive statistic of graph connectivity patterns. GFManifold is pre-trained on the mixed homophily source graph set with an average edge homophily of 58.26%. After geodesic edge reweighting, the target graphs exhibit smaller homophily gaps relative to the source graphs, resulting in improved downstream performance.

5.5 Efficiency of Tuning-free Adaptation

This experiment evaluates GFManifold's efficiency against other GFMs in Figure 5. For a fair comparison, all statistics are measured during downstream by each method's pre-trained model. GFManifold, highlighted in green, supports target-domain parameter tuning-free adaptation with zero tunable model parameters. This results in substantially lower time costs on the x-axis compared to existing GFMs. Furthermore, GFManifold demonstrates lower memory consumption, as indicated by the GPU peak memory values. The results verify the superior time and space efficiency of GFManifold through data-centric graph adaptation.

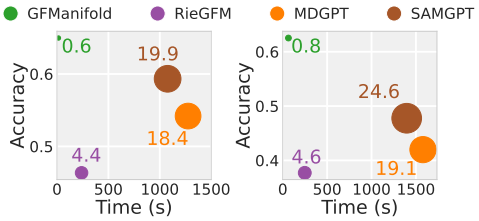


Figure 5: **Efficiency analysis** on Cornell (Left) and Citeseer (Right). The size of each point is proportional to the GPU peak memory and the numbers indicate GPU peak memory in GB. GFManifold (top-left in green) is tuning free for efficient downstream adaptation.

6 Related Work

GFManifold intersects with three major domains. **(i) Graph Pre-training** captures intrinsic patterns via self-supervised learning, including graph reconstruction (Hu et al., 2020b; Hou et al., 2022; Kipf & Welling, 2016; Lu et al., 2021) and contrastive learning (Zhu et al., 2020b; Zeng & Xie, 2021; You et al., 2020; Velickovic et al., 2019; Sun et al., 2019), yet often suffers from objective discrepancies (Sun et al., 2023b; Zhao et al., 2024). **(ii) Graph Foundation Models (GFMs)** adapt multi-domain pre-training to unseen targets (Liu et al., 2025). LLM-based GFMs (Liu et al., 2023a; Li et al., 2024; Chen et al., 2024a; Xia et al., 2024; Chen et al., 2024b; Kong et al., 2025) require text attributes, while GNN-based GFMs (Zhao et al., 2024; Yu et al., 2024b; 2025; Sun et al., 2025b; Wang et al., 2025; Yuan et al., 2025; Chen et al., 2025a; Jiang et al., 2026a; Wang et al., 2024; Sun et al., 2026a) apply to general graphs and are our focus. **(iii) Manifold Learning Methods on Graphs**, which mostly models graph structure via Riemannian manifolds (Xiong et al., 2022; Sun et al., 2024b;c; 2022a; 2023a; Ye et al., 2023; Sun et al., 2024a;c;b; 2026c; Wang et al., 2024), including spherical, hyperbolic (Meng et al., 2019; Bachmann et al., 2020; Chami et al., 2019; Ganea et al., 2018; Nickel & Kiela, 2018), product (de Ocáriz Borde et al., 2023; Gu et al., 2019), or quotient (Xiong et al., 2022; Law, 2021) spaces. Recent manifold-based GFM studies further explore Riemannian geometry for graph foundation modelling through manifold representations, geometric transfer, and structural learning (Sun et al., 2025b; 2026a;b; 2025a; Sun & Yu, 2026), while GFManifold focuses on tuning-free structure transformation for downstream cross-domain GFM adaptation. Additional details with more discussion are deferred to Appendix D due to page limit.

7 Conclusion

In this paper, we propose GFManifold, a structure-centric GFM framework for cross-domain GFM adaptation. By leveraging hyperbolic and spherical manifold branches with learnable curvatures, GFManifold learns manifold-induced structural relations shared across multiple domains. By transforming target graphs into geodesic-weighted graphs while keeping the pre-trained GNN fixed, GFManifold enables efficient GFM downstream adaptation without target-domain parameter tuning. Extensive experiments on 14 benchmarks demonstrate its superior effectiveness and efficiency.

AI USE STATEMENT

AI was used exclusively for grammar checking. All content, including the writing, analysis, interpretation, and presentation of the results, was completed by us. This manuscript was reviewed and revised by us to ensure its accuracy and consistency.

ETHICS STATEMENT

This work does not involve human subjects, private or sensitive personal data, or the collection of new user data. All experiments are conducted using publicly available graph benchmark datasets and established models and methods, as detailed in the full reproducibility statement in Appendix B. Therefore, we do not identify any specific ethical risks arising from the research methodology or experimental setting. All datasets, models, and code used in this work comply with their corresponding licences and terms of use.

REFERENCES

Gregor Bachmann, Gary Bécigneul, and Octavian Ganea. Constant curvature graph convolutional networks. In *ICML*, 2020.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21, 2005.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *NeurIPS*, 2019.

Haibo Chen, Xin Wang, Zeyang Zhang, Haoyang Li, Ling Feng, and Wenwu Zhu. Autogfm: Automated graph foundation model with adaptive architecture customization. In *ICML*, 2025a.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. Graphwiz: An instruction-following language model for graph problems. *arXiv preprint arXiv:2402.16029*, 2024a.

Qin Chen, Liang Wang, Bo Zheng, and Guojie Song. Dagprompt: Pushing the limits of graph prompting with a distribution-aware graph prompt tuning approach. *arXiv preprint arXiv:2501.15142*, 2025b.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024b.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *ICLR*, 2021.

Haitz Sáez de Ocáriz Borde, Anees Kazi, Federico Barbero, and Pietro Liò. Latent graph inference using product manifolds. In *ICLR*, 2023.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *NeurIPS*, 2018.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *ICLR*, 2019.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *SIGKDD*, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *SIGKDD*, 2020b.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. PRODIGY: Enabling In-context Learning Over Graphs. In *NeurIPS*, 2023.

Yan Jiang, Ruihong Qiu, and Zi Huang. GFMate: Empowering Graph Foundation Models with Test-time Prompt Tuning. In *ICML*, 2026a.

Yan Jiang, Ruihong Qiu, and Zi Huang. Does homophily help in robust test-time node classification? In *WSDM*, 2026b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. GOFA: A generative one-for-all model for joint graph language modeling. In *ICLR*, 2025.

Marc T. Law. Ultrahyperbolic neural networks. In *NeurIPS*, 2021.

Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning invariant graph representations for out-of-distribution generalization. In *NeurIPS*, 2022.

Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. Zerog: Investigating cross-dataset zero-shot transferability in graphs. *arXiv preprint arXiv:2402.11235*, 2024.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023a.

Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, and Chuan Shi. Graph foundation models: Concepts, opportunities and challenges. *TPAMI*, 2025.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*, 2023b.

Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. Learning to pre-train graph neural networks. In *AAAI*, 2021.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance M. Kaplan, and Jiawei Han. Spherical text embedding. In *NeurIPS*, 2019.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*, 2018.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9, 2021.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Li Sun and Philip S. Yu. A Riemannian Perspective on Graph Foundation Models: Curvature as a Guiding Principle. *Frontiers of Computer Science*, 20(12):2012370, 2026.

Li Sun, Junda Ye, Hao Peng, and Philip S. Yu. A Self-Supervised Riemannian GNN with Time Varying Curvature for Temporal Graph Learning. In *CIKM*, 2022a.

Li Sun, Junda Ye, Hao Peng, Feiyang Wang, and Philip S. Yu. Self-Supervised Continual Graph Learning in Adaptive Riemannian Spaces. In *AAAI*, 2023a.

Li Sun, Jingbin Hu, Suyang Zhou, Zhenhao Huang, Junda Ye, Hao Peng, Zhengtao Yu, and Philip S. Yu. RicciNet: Deep Clustering via A Riemannian Generative Model. In *WWW*, 2024a.

Li Sun, Zhenhao Huang, Qiqi Wan, Hao Peng, and Philip S. Yu. Spiking Graph Neural Network on Riemannian Manifolds. In *NeurIPS*, 2024b.

Li Sun, Zhenhao Huang, Zixi Wang, Feiyang Wang, Hao Peng, and Philip S. Yu. Motif-Aware Riemannian Graph Neural Network with Generative-Contrastive Learning. In *AAAI*, 2024c.

Li Sun, Zhenhao Huang, Ming Zhang, and Philip S. Yu. Deeper with Riemannian Geometry: Overcoming Oversmoothing and Oversquashing for Graph Foundation Models. In *NeurIPS*, 2025a.

Li Sun, Zhenhao Huang, Suyang Zhou, Qiqi Wan, Hao Peng, and Philip S. Yu. Riemanngfm: Learning a graph foundation model from riemannian geometry. In *WWW*, 2025b.

Li Sun, Zhenhao Huang, Silei Chen, Lanxu Yang, Junda Ye, Sen Su, and Philip S. Yu. Multi-Domain Riemannian Graph Gluing for Building Graph Foundation Models. In *ICLR*, 2026a.

Li Sun, Zhenhao Huang, Yiding Wang, Qin Chen, Pietro Liò, and Philip S. Yu. Are Common Substructures Transferable? Riemannian Graph Foundation Model with Neural Vector Bundles. In *ICML*, 2026b.

Li Sun, Qiqi Wan, Suyang Zhou, Zhenhao Huang, and Philip S. Yu. RiemannGL: Riemannian Geometry Changes Graph Deep Learning. *CoRR*, abs/2602.10982, 2026c.

Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *SIGKDD*, 2022b.

Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *SIGKDD*, 2023b.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. *CoRR*, abs/2502.02017, 2025.

Zehong Wang, Zheyuan Zhang, Nitesh V. Chawla, Chuxu Zhang, and Yanfang Ye. Gft: Graph foundation model with transferable tree vocabulary. In *NeurIPS*, 2024.

Lianghao Xia, Ben Kao, and Chao Huang. Opengraph: Towards open graph foundation models. *arXiv preprint arXiv:2403.01121*, 2024.

Bo Xiong, Shichao Zhu, Nico Potyka, Shirui Pan, Chuan Zhou, and Steffen Staab. Pseudo-riemannian graph convolutional networks. In *NeurIPS*, 2022.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Junda Ye, Zhongbao Zhang, Li Sun, Yang Yan, Feiyang Wang, and Fuxin Ren. SINCERE: Sequential Interaction Networks Representation Learning on Co-Evolving Riemannian Manifolds. In *WWW*, 2023.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *NeurIPS*, 2020.

Xingtong Yu, Jie Zhang, Yuan Fang, and Renhe Jiang. Non-homophilic graph pre-training and prompt learning. *arXiv preprint arXiv:2408.12594*, 2024a.

Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. Text-free multi-domain graph pre-training: Toward graph foundation models. *arXiv preprint arXiv:2405.13934*, 2024b.

Xingtong Yu, Zechuan Gong, Chang Zhou, Yuan Fang, and Hui Zhang. Samgpt: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *WWW*, 2025.

Haonan Yuan, Qingyun Sun, Junhua Shi, Xingcheng Fu, Bryan Hooi, Jianxin Li, and Philip S. Yu. How much can transfer? bridge: Bounded multi-domain graph foundation model with generalization guarantees. In *ICML*, 2025.

Jiaqi Zeng and Pengtao Xie. Contrastive self-supervised learning for graph classification. In *AAAI*, 2021.

Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *SIGKDD*, 2024.

Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael Bronstein, and Jian Tang. Fully-inductive node classification on arbitrary graphs. In *ICLR*, 2025.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *NeurIPS*, 2020a.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020b.

Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. Prog: A graph prompt learning benchmark. In *NeurIPS*, 2024.

A Appendix Overview

This appendix provides supplementary materials, theoretical proofs, and comprehensive experimental details. We summarise the structure as follows:

- **Appendix B** provides reproducibility details, including:
  - Baseline settings for supervised, self-supervised, and GFM methods in our experiments.
  - Dataset statistics for node classification in Table 4.
  - Dataset statistics for graph classification in Table 5.
  - The pseudo-code of GFManifold in Algorithm 1.
  - Complexity analysis of GFManifold.
  - Implementation details, including software versions, hardware environment, hyperparameter search space, and evaluation settings.
- **Appendix C** provides details on mapping between Euclidean and non-Euclidean spaces, including:
  - The exponential map from the tangent space at the manifold origin to the non-Euclidean manifold.
  - The logarithmic map from the manifold back to the tangent space.
  - The role of the tangent space in transferring representations between Euclidean and non-Euclidean spaces.
- **Appendix D** reviews detailed related work, including:
  - Graph pre-training methods in Appendix D.1.
  - General graph foundation models in Appendix D.2.
  - Euclidean-based GFM methods in Appendix D.3.
  - Manifold learning on graphs in Appendix D.4.
  - Comparison to existing manifold-based GFMs in Appendix D.5.
- **Appendix E** presents proof for theoretical analysis, including:
  - Additional detailed notation for graph components.
  - The Independence and Sufficiency assumptions used in the conditional analysis following existing GFMs.
  - The conditional cross-domain risk characterisation in Theorem 4.1 under these assumptions.
- **Appendix F** provides additional experimental results, including:
  - Results on graph classifications in Appendix F.1.
  - Results on more few-shot scenarios in Appendix F.2.
  - Effectiveness of parameter tuning-free adaptation in Appendix F.3.
  - Effectiveness upon different backbones in Appendix F.4.
  - Robustness on pre-training domains in Appendix F.5.
  - Hyperparameter analysis of λ in Appendix F.6.
  - Further visualisation results in Appendix F.7.
- **Appendix G** discusses the limitations of GFManifold.
- **Appendix H** presents the impact statement of GFManifold.

B Reproducibility Statement

To facilitate reproducible research, we will release the code upon acceptance. Our efforts can be summarised as follows:

- **Baselines.** We adopt baseline methods from existing GFM methods, including SAMGPT (Yu et al., 2025), MDGPT (Yu et al., 2024b), GCOPE (Zhao et al., 2024), RiemannGFM (Sun et al., 2025b) and GraphGlue (Sun et al., 2026a), and carefully select their hyperparameters following the recommendations in their respective papers for a fair comparison. Specifically:
  - For GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), SAGE (Hamilton et al., 2017), H2GCN (Zhu et al., 2020a), and GPR (Chien et al., 2021), the hidden dimension is chosen from 32, 64, 128, 256, with the number of layers searched in the range 2–4. The dropout rate is tuned within [0, 1].
  - For Link Prediction (Lu et al., 2021), DGI (Velickovic et al., 2019), and GCL (You et al., 2020), GCN with a hidden dimension of 256 is used as the default backbone. PReLU is applied, following prior work (Yu et al., 2024a; 2025), and edge dropping with a drop rate of 0.2 is adopted as the default augmentation, in line with SAMGPT (Yu et al., 2025).
  - For GCOPE (Zhao et al., 2024), a 2-layer GCN with 100 hidden dimensions is used as the backbone. Downstream adaptation is performed via fine-tuning, as in the original implementation. Following GCOPE, the feature dimension of all source datasets is aligned to 100 via SVD across all GFM methods for consistency.
  - For MDGPT (Yu et al., 2024b) and SAMGPT (Yu et al., 2025), a 3-layer GCN is used as the base model for all datasets, with the hidden dimension set to 256, as recommended in their papers.
  - For MDGFM (Wang et al., 2025), although the code repository has been made public recently, it cannot facilitate reimplementation on our datasets, as the core component for few-shot data generation is missing. Consequently, the reported performance from their paper is adopted instead.
  - For RiemannGFM (Sun et al., 2025b), the number of Riemannian layers is fixed at 2. The parameterised scalar map in cross-geometry attention is implemented as an MLP with one hidden layer of dimension 256, following the instructions in the paper.
  - For GraphGlue (Sun et al., 2026a), we adapt the official implementation to our dataset pipeline while preserving its original training protocol. We use the default 2-layer GCN with a hidden dimension of 512 and 32 generators, ensuring that all baselines are evaluated under a consistent and fair experimental setting.

Table 4: **Datasets statistics for node classification.**

Dataset	#Nodes	#Edges	#Features	Homophily degree	#Classes	#Full-shot	Domain	Source
Texas	183	558	1703	17.25	5	87	Social Network	(Pei et al., 2020)
Cornell	183	554	1703	20.27	5	87	Social Network	(Pei et al., 2020)
Wisconsin	251	900	1703	19.80	5	120	Social Network	(Pei et al., 2020)
Chameleon	2277	36101	2325	24.90	5	1092	Social Network	(Rozemberczki et al., 2021)
Squirrel	5201	217073	2089	21.90	5	2496	Social Network	(Rozemberczki et al., 2021)
Arxiv-year	169343	1166243	128	30.97	5	100	Citation Network	(Hu et al., 2020a)
Cora	2708	10556	1433	84.99	7	140	Citation Network	(Yang et al., 2016)
Citeseer	3327	9104	3703	71.66	6	120	Citation Network	(Yang et al., 2016)
Pubmed	19717	88648	500	79.24	3	60	Citation Network	(Yang et al., 2016)
Amazon-computers	13752	491722	767	80.17	10	200	Commercial System	(Shchur et al., 2018)
Amazon-photo	7650	238162	745	84.93	8	160	Commercial System	(Shchur et al., 2018)

Table 5: **Datasets statistics for graph classification.**

Dataset	#Graphs	Avg.#Nodes	Avg.#Edges	#Features	#Classes	Domain	Source
BZR	405	35.8	38.4	3	2	Biochemical Molecules	(Rossi & Ahmed, 2015)
COX2	467	41.2	43.5	3	2	Biochemical Molecules	(Rossi & Ahmed, 2015)
PROTEINS	1113	39.1	72.8	3	2	Protein–Protein Interaction	(Borgwardt et al., 2005)

- **Datasets.** We utilise 14 publicly available graph benchmark datasets as in Table 4 and 5. These include:
  - Cora and Citeseer (Yang et al., 2016) are academic citation networks where nodes denote publications and edges represent citation links.
  - PubMed (Yang et al., 2016) consists of 19,717 biomedical publications related to diabetes, each classified into one of three categories: Diabetes Mellitus, Experimental Diabetes Mellitus and Type 1 Diabetes Mellitus.
  - Amazon-Photo (Shchur et al., 2018) (Photo) is a co-purchasing network whose nodes correspond to products and edges indicate frequent co-purchases.
  - Amazon-Computers (Computers) (Shchur et al., 2018) includes 13,752 computer-related products divided into ten categories. The co-purchase network consists of 245,861 edges representing products frequently bought together. Each product is described by a feature vector generated from metadata and reviews, with labels according to its category.
  - Arxiv-year (Arxiv) (Hu et al., 2020a) contains 169,343 computer science papers, forming a large-scale citation network, with each paper represented by a 128-dimensional feature vector averaged from title and abstract word embeddings.
  - Chameleon and Squirrel (Rozemberczki et al., 2021) are Wikipedia-derived page-page networks in which nodes are web pages, edges are mutual hyperlinks, and features encode informative nouns extracted from the pages.
  - Cornell (Pei et al., 2020) is a webpage network comprising 183 nodes and 295 hyperlinks, with node features obtained from a bag-of-words representation.
  - Texas (Pei et al., 2020) is a webpage network containing nodes representing university webpages and edges denoting hyperlinks.
  - Wisconsin (Pei et al., 2020) is also a webpage network whose nodes represent webpages and edges denote hyperlinks. Features are from a bag-of-words representation, and the webpages are classified into categories such as student, course and staff.
  - BZR (Rossi & Ahmed, 2015) is a dataset of ligand graphs associated with the benzodiazepine receptor, where each graph represents a molecule. The molecules are classified into two categories.
  - COX2 (Rossi & Ahmed, 2015) is a dataset of molecular graphs representing cyclooxygenase-2 inhibitors. Nodes correspond to atoms, edges represent types of chemical bonds, and the molecules are classified into two categories.
  - PROTEINS (Borgwardt et al., 2005) is a dataset of protein graphs where nodes represent secondary structures and edges indicate spatial or sequential proximity. Node features capture structural and biochemical properties.
- **Algorithm.** The GFManifold framework is comprehensively detailed in the method section. Additionally, pseudo-code is provided in Algorithm 1.
- **Complexity Analysis.** The complexity of GFManifold consists of two main components: (i) multi-domain pre-training, which includes GNN encoding, mapping nodes to the manifold via the exponential map, and computing geodesic distances for MMD and geodesic link prediction. With K source-domain graphs, each having N nodes and E edges, the complexity is dominated by $\mathcal{O}(K(N^2 + E + Nd))$, where N^2 accounts for pairwise computations in MMD, E for edge-based operations, and Nd for manifold mapping. (ii) Tuning-free downstream adaptation for a target graph with N nodes and E edges involves manifold mapping, edge-wise geodesic reweighting, and GNN inference, requiring $\mathcal{O}(dL(N + E) + Ed + Nd)$ in total. For fixed d and L, pre-training scales linearly with the number of source domains K, while downstream adaptation scales linearly with the target graph size $N + E$. Therefore, GFManifold introduces no additional target-domain parameter optimisation cost during downstream adaptation.
- **Implementation Details.** We use publicly available benchmark datasets listed in Table 4 and Table 5 in accordance with their respective licences and terms of use. The datasets are sourced from PyTorch Geometric 2.6.1 and OGB 1.3.6, both under the MIT license. Experiments are performed with Python 3.8.2 and PyTorch 2.4.1 using CUDA 12.2 on a single NVIDIA A6000 GPU (48GB memory). Baseline codes are obtained from official GitHub repositories, with hyperparameters carefully selected to ensure fair comparison according to the recommendations in their papers. Hidden dimensions are selected from 32, 64, 128, 256, and layer counts range from 2 to 5. The branch curvatures are parameterised as $\kappa_{\mathbb{S}} = \mathrm{clamp}(\rho_{\mathbb{S}}, \min = \epsilon)$ and $\kappa_{\mathbb{H}} = -\mathrm{clamp}(\rho_{\mathbb{H}}, \min = \epsilon)$, where $\epsilon = 10^{-5}$ keeps both branches away from zero ($\kappa_{\mathbb{S}} \geq \epsilon > 0$ and $\kappa_{\mathbb{H}} \leq -\epsilon < 0$). For each MMD computation, the same number of nodes is uniformly sampled from each source domain, with the sample size set to the minimum source-domain node count. The Adam optimiser (Kingma & Ba,

Algorithm 1: GFManifold Overview

Input: Source domains $\{\mathcal{G}_i\}_{i=1}^{K}$, where $\mathcal{G}_i = (\boldsymbol{X}_i, \boldsymbol{A}_i)$; target domain $\mathcal{G}_{\text{Tar}} = (\boldsymbol{X}_{\text{Tar}}, \boldsymbol{A}_{\text{Tar}})$; support set $\mathcal{H}_c$ for each class $c \in \mathcal{C}$.
Output: Predicted label $\hat{y}$ for target test nodes.

% Stage 1: Multi-domain Pre-training on Source Domains
while not converged **do**
  for $i = 1, \ldots, K$ **do**
    $\boldsymbol{H}_i \leftarrow \text{GNN}_\theta(\boldsymbol{X}_i, \boldsymbol{A}_i)$.
    Map $\boldsymbol{H}_i$ to the spherical and hyperbolic manifold branches, collectively denoted by $\boldsymbol{H}_i^{\mathcal{M}_\kappa}$, via Eq. 5.
  end for
  Compute the composite manifold-induced dissimilarity d_κ via Eq. 6.
  Compute $\mathcal{L}_{\text{MMD}}$ (Eq. 8) and $\mathcal{L}_{\text{GLP}}$ (Eq. 9).
  $\mathcal{L}_{\text{GFManifold}} \leftarrow \lambda\mathcal{L}_{\text{MMD}} + (1-\lambda)\mathcal{L}_{\text{GLP}}$.
  Update θ, $\kappa_{\mathbb{H}}$, $\kappa_{\mathbb{S}}$, α_0, and α_1 by minimising $\mathcal{L}_{\text{GFManifold}}$.
end while
% Stage 2: Tuning-free Downstream Adaptation on an Unseen Target Domain
Freeze the source-learnt parameters θ, $\kappa_{\mathbb{H}}$, $\kappa_{\mathbb{S}}$, α_0, and α_1.
$\boldsymbol{H}_{\text{Tar}} \leftarrow \text{GNN}_\theta(\boldsymbol{X}_{\text{Tar}}, \boldsymbol{A}_{\text{Tar}})$.
Map $\boldsymbol{H}_{\text{Tar}}$ to the spherical and hyperbolic manifold branches, collectively denoted by $\boldsymbol{H}_{\text{Tar}}^{\mathcal{M}_\kappa}$, via Eq. 5.
Compute structural weights $\boldsymbol{S}$ for the existing target-domain edges.
Refine $\boldsymbol{S}_{u,v} \leftarrow 0$ for edges with $\boldsymbol{S}_{u,v} < 0.1$ and construct $\mathcal{G}_{\text{Geo}}$.
$\boldsymbol{H} \leftarrow \text{GNN}_\theta(\boldsymbol{X}_{\text{Tar}}, \boldsymbol{S})$.
% Few-shot Classification Setting
for each class $c \in \mathcal{C}$ **do**
  $\bar{\boldsymbol{h}}_c \leftarrow \frac{1}{|\mathcal{H}_c|}\sum_{\boldsymbol{h}_c \in \mathcal{H}_c} \boldsymbol{h}_c$.
end for
Predict $\hat{y} \leftarrow \text{argmax}_{c \in \mathcal{C}} \cos(\boldsymbol{h}, \bar{\boldsymbol{h}}_c)$.
return $\hat{y}$

2015) is used with learning rates searched over $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. Edges with small geodesic weights below 0.1 are pruned. GFManifold's hyperparameter λ is selected from $\{0.1, 0.2, \ldots, 1.0\}$ using source-domain validation data and remains fixed for all target-domain evaluations. The default backbone is a GCN, consistent with SAMGPT (Yu et al., 2025) and MDGPT (Yu et al., 2024b). All experiments are repeated five times. Mean accuracy and standard deviation are reported in percentage.

- **Open Source.** The code will be released upon acceptance to facilitate reproduction.
- **Limitation.** The potential limitations are discussed in Section G.

C Mapping between Euclidean and Non-Euclidean Spaces

In this context, Euclidean space refers specifically to the tangent space at the manifold origin o, which provides a locally flat approximation for transferring representations between Euclidean and non-Euclidean spaces. For a vector on the manifold origin tangent space $\boldsymbol{t} \in T_{\boldsymbol{o}}\mathcal{M}_\kappa$, the exponential map projects it from the Euclidean space to the non-Euclidean manifold as:

$$\exp_{\boldsymbol{o}}(\boldsymbol{t}) = \begin{cases} \cos\left(\sqrt{\kappa}\,\|\boldsymbol{t}\|\right)\boldsymbol{o} + \dfrac{\sin\left(\sqrt{\kappa}\,\|\boldsymbol{t}\|\right)}{\sqrt{\kappa}\,\|\boldsymbol{t}\|}\,\boldsymbol{t}, & \kappa > 0, \\ \cosh\left(\sqrt{-\kappa}\,\|\boldsymbol{t}\|\right)\boldsymbol{o} + \dfrac{\sinh\left(\sqrt{-\kappa}\,\|\boldsymbol{t}\|\right)}{\sqrt{-\kappa}\,\|\boldsymbol{t}\|}\,\boldsymbol{t}, & \kappa < 0. \end{cases} \tag{13}$$

For any point vector on the manifold $\boldsymbol{u} \in \mathcal{M}_\kappa$, the logarithmic map to the tangent space at the origin $\boldsymbol{o}$ is defined as:

$$\log_{\boldsymbol{o}}(\boldsymbol{u}) = \begin{cases} \dfrac{\arccos\left(\kappa\langle\boldsymbol{o},\boldsymbol{u}\rangle_\kappa\right)}{\sqrt{\kappa}\,\|\boldsymbol{u}-\kappa\langle\boldsymbol{o},\boldsymbol{u}\rangle_\kappa\boldsymbol{o}\|}\,(\boldsymbol{u}-\kappa\langle\boldsymbol{o},\boldsymbol{u}\rangle_\kappa\boldsymbol{o}), & \kappa > 0, \\ \dfrac{\operatorname{arccosh}\left(\kappa\langle\boldsymbol{o},\boldsymbol{u}\rangle_\kappa\right)}{\sqrt{-\kappa}\,\|\boldsymbol{u}-\kappa\langle\boldsymbol{o},\boldsymbol{u}\rangle_\kappa\boldsymbol{o}\|}\,(\boldsymbol{u}-\kappa\langle\boldsymbol{o},\boldsymbol{u}\rangle_\kappa\boldsymbol{o}), & \kappa < 0, \end{cases} \tag{14}$$

where $\|\cdot\|$ denotes the standard Euclidean norm. These maps enable effective transfer between Euclidean representations in the tangent space and points on the non-Euclidean manifold.

D Detailed Related Work

D.1 Graph Pre-training Methods.

Graph pre-training methods aim to capture intrinsic graph characteristics, typically via self-supervised learning. They fall into two categories: (i) Graph Reconstruction Methods, which recover specific graph attributes (Hu et al., 2020b; Hou et al., 2022; Kipf & Welling, 2016; Lu et al., 2021); (ii) Graph Contrastive Learning Methods, which learn representations by contrasting different views (Zhu et al., 2020b; Zeng & Xie, 2021; You et al., 2020; Velickovic et al., 2019; Sun et al., 2019). Unlike GFMs that adapt to diverse downstream tasks, these methods often face objective discrepancies between pre-training and fine-tuning, limiting generalisation (Sun et al., 2023b; Zhao et al., 2024).

D.2 General Graph Foundation Models.

GFMs aim to learn generalisable graph representations by leveraging pre-training on large-scale source graphs to adapt to a wide range of downstream tasks and unseen domains (Liu et al., 2025). Current GFMs can be broadly categorised based on backbones into (i) LLM-based GFMs (Liu et al., 2023a; Li et al., 2024; Chen et al., 2024a; Xia et al., 2024; Chen et al., 2024b; Kong et al., 2025) and (ii) GNN-based GFMs (Zhao et al., 2024; Yu et al., 2024b; 2025; Sun et al., 2025b; Wang et al., 2025; Yuan et al., 2025; Zhao et al., 2025; Huang et al., 2023). LLM-based GFMs utilise textual information in graphs by large language models (LLMs) (Brown et al., 2020), but are inherently limited to text-attributed graphs (TAGs). In contrast, GNN-based GFMs operate in general continuous feature spaces and are applicable to both text-free graphs (Zhao et al., 2024; Sun et al., 2025b; Wang et al., 2025; Yu et al., 2024b; 2025) and text-attributed graphs by pre-encoding the text into embeddings. Other approaches explore tuning-free graph transfer from complementary perspectives, including fully inductive inference with GraphAny (Zhao et al., 2025), cross-dataset zero-shot transfer with ZeroG (Li et al., 2024), and in-context graph learning with PRODIGY (Huang et al., 2023). **Therefore, we focus on enhancing GNN-based GFMs for general graph types.**

D.3 Euclidean-based GFM Methods.

Most GFM methods align source and unseen target domains by tuning prompts (Liu et al., 2023b; Zi et al., 2024), which are additional learnable embeddings in Euclidean space. These prompts are mainly injected into the encoding process to adapt node or graph representations. For instance,

MDGPT (Yu et al., 2024b) uses unified and mixed prompts, while SAMGPT (Yu et al., 2025) employs dual-stage structural prompts for source-target domain alignment. GFT (Wang et al., 2024) captures transferable graph structures through discrete computation-tree tokenisation, whereas GFManifold models cross-domain structural relations continuously through manifold-induced geodesic dissimilarities. BRIDGE (Yuan et al., 2025) pre-trains a graph feature aligner and facilitates feature adaptation through a lightweight Mixture-of-Experts (MoE) network, which is further leveraged for downstream classification via prompt tuning. These methods may encode structural signals into prompts, but the dominant adaptation mechanism is still representation-level tuning. In contrast, GFManifold focuses on the graph structure aspect: it learns manifold-induced cross-domain graph structural relations during multi-domain pre-training and then adapts the target graph by reweighting its edges. Moreover, existing prompt-based GFMs rely heavily on prompt or model fine-tuning with target few-shot samples, which significantly hinders efficiency as analysed in Section 5.5. Without target-domain tuning, their designs lead to performance degradation as showcased in Appendix F.3.

D.4 Manifold Learning on Graphs.

Recent research has explored learning graph representations on non-Euclidean Riemannian manifolds (Xiong et al., 2022; Sun et al., 2024b;c; 2022a; 2023a; Ye et al., 2023; Sun et al., 2024a;c;b; Bachmann et al., 2020), leveraging geometric properties to model complex graph topologies. Some methods embed nodes in a single constant-curvature space (Bachmann et al., 2020), such as spherical or hyperbolic manifolds (Meng et al., 2019; Chami et al., 2019; Ganea et al., 2018; Nickel & Kiela, 2018), while others employ product (de Ocáriz Borde et al., 2023; Gu et al., 2019) and quotient manifolds (Xiong et al., 2022; Law, 2021) to capture mixed-curvature characteristics. These approaches are typically developed for graph representation learning, where manifold embeddings are the main objective.

Recent work further extends Riemannian geometry to graph foundation models and cross-domain graph learning (Sun et al., 2025b; 2026a;b; 2025a; Sun & Yu, 2026), while RiemannGL (Sun et al., 2026c) provides a broader Riemannian perspective on graph learning. RiemannGFM (Sun et al., 2025b) relies on pre-defined fixed-curvature manifolds, while GraphGlue (Sun et al., 2026a) requires further training of additional prompt embeddings for each downstream domain, following a similar prompt-based fine-tuning paradigm as previous GFMs (Yu et al., 2025; Wang et al., 2025). In contrast, GFManifold uses manifolds as a graph structure learner for cross-domain GFM adaptation without relying on additional prompt tuning or model fine-tuning. The learnt geodesic relations are used to transform graph structure, rather than merely replacing Euclidean embeddings with manifold embeddings, and enable tuning-free downstream adaptation without further fine-tuning for each new target domain.

D.5 Comparison with Existing Manifold-based GFMs

We further clarify the relationship between GFManifold and existing manifold-based GFMs, particularly RiemannGFM (Sun et al., 2025b) and GraphGlue (Sun et al., 2026a). GFManifold does not aim to provide a universally more expressive geometric model than these methods. Instead, it differs mainly in its curvature parameterisation, multi-domain learning objective, and the use of manifold-induced structural relations for target-domain graph adaptation.

Difference from RiemannGFM. GFManifold differs from RiemannGFM (Sun et al., 2025b), a recent GFM method that adopts Riemannian manifolds, in the following aspects.

(1) Curvature learning. RiemannGFM operates on hyperbolic and spherical manifolds with fixed curvature settings, where curvature is not learnt but set to predefined standard constants, e.g., -1 for hyperbolic space and 1 for hyperspherical space. In contrast, GFManifold learns one globally shared curvature value for each of its hyperbolic and spherical branches from multi-domain source graphs, allowing the corresponding model-space geometries to adapt during pre-training while keeping each manifold branch a constant-curvature space. Specifically, constant curvature means that curvature is spatially uniform within each manifold, rather than varying across nodes or locations. These learnt curvatures parameterise the predefined manifold spaces and are not intended to estimate the intrinsic curvature of individual graphs.

(2) Downstream adaptation. RiemannGFM requires downstream fine-tuning of the classifier on each target domain, whereas GFManifold performs target-domain parameter tuning-free adaptation without prompt, classifier, curvature, or model parameter updates.

(3) Structure-centric design. RiemannGFM uses hyperbolic and spherical manifold representations as part of its representation learning framework. GFManifold instead uses the manifold branches primarily to derive structural relations rather than directly using manifold embeddings as final downstream representations. These manifold-induced relations are used to reweight target-domain edges, producing geodesic-weighted graphs that are subsequently encoded by the fixed pre-trained GNN.

(4) Cross-domain training. Under the official implementation used in our experiments, RiemannGFM performs pre-training sequentially on individual source-domain graphs. In contrast, GFManifold is jointly pre-trained across multiple source domains through MMD-based domain alignment and geodesic link prediction, learning manifold-induced structural relations across source domains.

Difference from GraphGlue. GraphGlue (Sun et al., 2026a) is another recent manifold-based graph method that constructs a smooth manifold by gluing local manifold pieces. GFManifold addresses a different aspect of manifold-based graph learning, focusing on jointly learning two globally shared constant-curvature branches and using their induced structural relations for cross-domain graph adaptation.

(1) Learning objective. GraphGlue focuses on gluing local manifold pieces through geometric smoothness and consistency, whereas GFManifold jointly optimises hyperbolic and spherical constant-curvature manifold branches with a GNN encoder through MMD-based domain alignment and geodesic link prediction. The two methods therefore use manifold geometry for different learning objectives.

(2) Curvature parameterisation. GraphGlue does not explicitly parameterise curvature as independently learnable scalar variables for hyperbolic and spherical spaces. Its geometry is instead induced by learnable quantities within the glued manifold construction. In contrast, GFManifold explicitly learns two globally shared scalar curvature parameters, one for the hyperbolic branch and one for the spherical branch, during multi-domain source pre-training.

(3) Metric construction. GraphGlue constructs a Riemannian metric from learnable tangent-space quantities, e.g., $G = W^\top W$, such that its metric and geometric properties are induced by its learnable variables. GFManifold does not construct a new unified Riemannian metric from its two manifold branches. Instead, it computes valid geodesic distances within the hyperbolic and spherical branches separately and aggregates them as a composite manifold-induced structural relation for domain alignment and graph reweighting.

(4) Downstream graph transformation. GraphGlue primarily focuses on manifold gluing and geometric consistency, and performs downstream adaptation by further optimising additional prompt parameters on the target domain. In contrast, GFManifold derives hyperbolic and spherical representations from a shared Euclidean tangent representation through curvature-specific exponential maps and uses the resulting manifold-induced relations to reweight the target graph structure. The pre-trained GNN and manifold parameters remain fixed during target-domain adaptation, without introducing additional learnable prompt or model parameters.

E RISK ANALYSIS FOR GFMANIFOLD

Notation. For the ideal theoretical analysis, let $\Phi_{\text{shared}}(\mathcal{G}) = \mathcal{M}_\kappa(\mathcal{G})$ denote the shared structural component represented by GFManifold under the assumptions stated below. Let $\Phi_{\text{specific}}(\mathcal{G}) = \mathcal{G} \setminus \Phi_{\text{shared}}(\mathcal{G})$ denote the remaining domain-specific component. This notation is introduced only for the conditional analysis and does not assert that the practical objectives exactly recover such a decomposition.

Proof. We characterise the expected cross-domain risk of a predictor that operates on the shared structural component under the following assumptions in consistent with existing GFMs (Yuan et al., 2025; Wang et al., 2025).

ASSUMPTION E.1 (**INDEPENDENCE**)
For any domain $i \in \mathcal{S} \cup \mathcal{T}$, the shared and domain-specific structural components are independent:

$$\Phi_{\text{shared}}(\mathcal{G}) \perp \Phi_{\text{specific}}(\mathcal{G}). \tag{15}$$

ASSUMPTION E.2 (**SUFFICIENCY**)
Conditioned on the shared structural component, the label $\boldsymbol{Y}$ is independent of the domain-specific component:

$$\boldsymbol{Y} \perp \Phi_{\text{specific}}(\mathcal{G}) \mid \Phi_{\text{shared}}(\mathcal{G}). \tag{16}$$

Let $\boldsymbol{Z}_{\text{shared}} = \Phi_{\text{shared}}(\mathcal{G})$ and $\boldsymbol{Z}_{\text{specific}} = \Phi_{\text{specific}}(\mathcal{G})$. Under Assumptions E.1 and E.2, their joint distribution factorises as

$$P^i(\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Z}_{\text{specific}}, \boldsymbol{Y}) = P^i(\boldsymbol{Z}_{\text{specific}}) P^i(\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Y}). \tag{17}$$

For a predictor g that operates on the shared structural component, its expected risk on domain i is therefore

$$\begin{aligned} R(g \mid i) &= \sum_{\boldsymbol{Z}_{\text{specific}}} \sum_{\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Y}} P^i(\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Z}_{\text{specific}}, \boldsymbol{Y}) \mathcal{L}\big(g(\boldsymbol{Z}_{\text{shared}}), \boldsymbol{Y}\big) \\ &= \sum_{\boldsymbol{Z}_{\text{specific}}} P^i(\boldsymbol{Z}_{\text{specific}}) \sum_{\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Y}} P^i(\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Y}) \mathcal{L}\big(g(\boldsymbol{Z}_{\text{shared}}), \boldsymbol{Y}\big) \\ &= \sum_{\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Y}} P^i(\boldsymbol{Z}_{\text{shared}}, \boldsymbol{Y}) \mathcal{L}\big(g(\boldsymbol{Z}_{\text{shared}}), \boldsymbol{Y}\big). \end{aligned} \tag{18}$$

Equation 18 shows that, under the adopted assumptions, the expected risk depends on the shared structural component and the label, while the domain-specific component is marginalised out. Note that these assumptions follow prior GFM analyses (Yuan et al., 2025; Wang et al., 2025) and provide a consistent theoretical setting for GFM comparison.

Let $\mathcal{F}_{\text{shared}}$ denote the class of GFManifold predictors operating on the shared structural component. Define the conditional minimax predictor as

$$g^* \in \arg \min_{g \in \mathcal{F}_{\text{shared}}} \sup_{i \in \mathcal{S} \cup \mathcal{T}} R(g \mid i). \tag{19}$$

By the definition of g^*, for any $g \in \mathcal{F}_{\text{shared}}$,

$$\sup_{i \in \mathcal{S} \cup \mathcal{T}} R(g^* \mid i) \leq \sup_{i \in \mathcal{S} \cup \mathcal{T}} R(g \mid i). \tag{20}$$

Therefore, under the Independence and Sufficiency assumptions, the cross-domain risk of the idealised GFManifold predictor can be characterised through the shared structural component. This analysis is conditional and does not claim that the practical MMD and GLP objectives exactly recover $\Phi_{\text{shared}}(\mathcal{G})$ or exactly solve the minimax optimisation problem. □

F Additional Experimental Results

F.1 Results on Graph Classifications

Table 6 further evaluates whether the model learnt from node classification can be transferred to graph classification tasks. It is worth noting that several existing GFMs are not directly applicable to graph classification. For example, MDGFM (Wang et al., 2025) and RiemannGFM (Sun et al., 2025b) are mainly designed for node-level tasks and their official implementations do not support graph classification. In contrast, GFManifold can be freely extended to graph-level tasks by applying the same tuning-free structural adaptation strategy to graph data, without requiring additional model fine-tuning. As shown in Table 6, GFManifold consistently achieves the best performance on all three graph classification datasets, improving over the strongest baselines on PROTEINS, COX2, and BZR. These results demonstrate that the learnt non-Euclidean shared cross-domain structural relations are not limited to node classification, but can also provide effective and generalisable structural information for graph-level prediction.

Table 6: **Performance of graph classification.** MDGFM (Wang et al., 2025) and RiemannGFM (Sun et al., 2025b) do not support graph classification.

Category	Methods	PROTEINS	COX2	BZR
Supervised	GCN	51.24±7.96	45.59±9.62	44.89±16.27
	GAT	50.89±8.04	50.71±7.26	46.58±15.83
	SAGE	51.39±6.92	55.74±5.98	52.39±12.98
	GPR	51.09±9.26	48.15±9.69	45.16±17.63
	H2GCN	52.28±8.75	53.67±7.82	49.88±13.35
SSL+FT	LP+FT	52.29±7.50	53.91±12.24	52.29±16.73
	DGI+FT	49.68±8.37	54.42±10.03	49.79±14.95
	GCL+FT	50.75±7.86	46.61±9.85	50.48±16.89
GFM+FT	GCOPE	53.21±6.44	54.90±13.25	58.15±16.09
	MDGPT	54.70±6.69	50.47±12.35	53.48±12.92
	SAMGPT	55.19±5.52	52.07±7.82	58.59±17.04
Tuning-free	**GFManifold**	**62.10**±1.15	**56.47**±4.26	**61.49**±4.24

F.2 Results on More Few-Shot Scenarios

This experiment evaluates GFManifold and GFMs on few-shot settings with various numbers of few-shot labelled data in Figure 6. **GFManifold (green) with tuning-free adaptation, consistently outperforms existing GFMs** as the number of shots increases and reaches higher performance with a lower number of shots. GFManifold exhibits lower variability, as indicated by the shaded area, reflecting stable performance compared with existing GFMs.

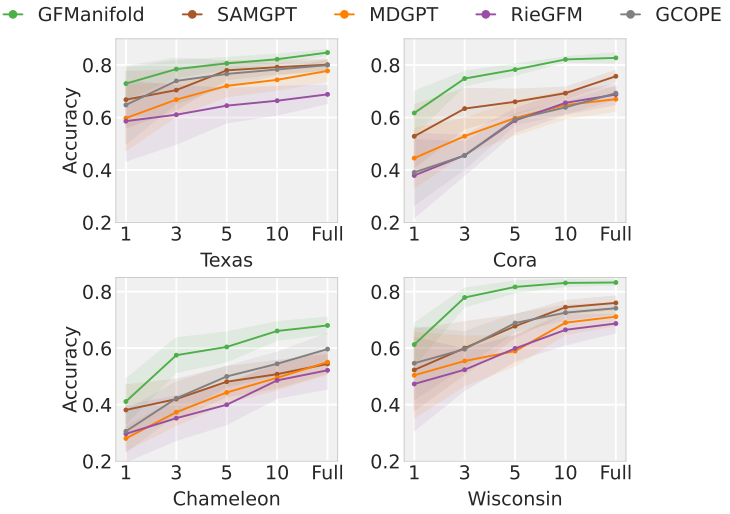


Figure 6: **Various labelled data in few-shot node classification.** RieGFM is short for RiemannGFM. The shaded regions indicate the standard deviation for each method.

F.3 Effectiveness of Tuning-free Adaptation

During the downstream stage, GFManifold enables tuning-free data-centric adaptation by transforming the target-domain graph into a geodesic-weighted graph using the learnt manifold-induced structural relations. As shown in Table 7, in a consistent tuning-free downstream setting, removing target-domain fine-tuning reduces the performance of existing tuning-based GFMs MDGPT and SAMGPT. In contrast, **with geodesic-based graph adaptation, GFManifold significantly outperforms all baselines, both with and without fine-tuning.** This showcases the effectiveness of GFManifold's tuning-free downstream adaptation.

Table 7: **Effectiveness of GFManifold tuning-free downstream adaptation** $*$ refers to the GFMs without target domain fine-tuning in a consistent tuning-free scenario.

Method	Texas	Cornell	Cora	Photo
MDGPT	59.76±12.44	59.14±13.08	44.52±11.39	54.96±10.25
SAMGPT	66.79±10.77	59.34±9.82	52.83±12.04	56.33±9.04
MDGPT*	49.65±9.82	42.18±10.47	42.07±10.60	47.60±9.73
SAMGPT*	50.10±8.17	45.39±11.65	45.88±9.50	50.39±8.94
GFManifold	**72.92±5.98**	**64.97±11.05**	**61.71±8.32**	**63.27±3.38**

F.4 Effectiveness upon Different Backbones

To evaluate the versatility of the proposed framework, GFManifold is implemented across diverse backbones, including GCN, SAGE, GAT, and H2GCN. As illustrated in Table 8, GFManifold consistently outperforms existing GFMs regardless of the backbone architecture, while existing GFMs typically adhere to a fixed GCN backbone. Results confirm that the performance gains of GFManifold stem from the manifold-based data-centric graph transformation rather than specific backbone choice.

Table 8: **Backbone Analysis.** GFManifold supports various backbones and remains insensitive to backbone choice, whereas existing GFMs are typically designed upon a fixed GCN backbone.

Method	Texas	Cornell	Cora	Photo
MDGPT (GCN)	59.76±12.44	59.14±13.08	44.52±11.39	54.96±10.25
SAMGPT (GCN)	66.79±10.77	59.34±9.82	52.83±12.04	56.33±9.04
GFManifold-GCN	72.92±5.98	64.97±11.05	61.71±8.32	63.27±3.38
GFManifold-SAGE	70.69±6.13	65.50±9.68	62.35±6.94	63.31±4.07
GFManifold-GAT	73.03±7.16	62.25±10.60	60.12±7.03	62.59±3.58
GFManifold-H2GCN	73.21±7.76	65.82±11.90	60.02±9.71	61.24±4.31

Table 9: **Robustness Analysis.** Compared with state-of-the-art GFM SAMGPT, GFManifold is robust to domain changes in the pre-training domains and consistently outperforms SAMGPT.

Method	Texas	Cornell	Cora	Photo
SAMGPT	66.79±10.77	59.34±9.82	52.83±12.04	56.33±9.04
– w/o Social	59.98±12.14	52.88±9.36	55.29±13.03	57.40±10.68
– w/o Citation	62.26±8.77	62.15±12.13	47.38±9.64	58.82±8.94
– w/o Commercial	60.30±9.03	63.79±11.55	52.96±10.91	49.89±9.63
GFManifold	72.92±5.98	64.97±11.05	61.71±8.32	63.27±3.38
– w/o Social	66.75±8.19	60.47±9.72	65.35±8.11	62.96±5.31
– w/o Citation	73.24±6.98	65.54±4.98	55.47±7.79	65.13±6.20
– w/o Commercial	72.85±5.23	66.08±13.35	62.24±5.31	55.39±5.52

F.5 Robustness on Pre-training Domains

The robustness of GFManifold is evaluated by changing the pre-training domain, as shown in Table 9. **GFManifold remains robust to variations in pre-training domains and consistently outperforms SAMGPT in the same cross-domain adaptation setting.** Notably, removing certain domains, such as excluding commercial domains from the pre-training datasets for the target domain Cornell, results

in higher performance. This can be due to cross-domain graphs introducing conflicts or interferences rather than synergistic effects (Yu et al., 2025), which is a common phenomenon consistently observed in prior GFMs (Wang et al., 2025; Yu et al., 2025).

F.6 Hyperparameter Analysis

The hyperparameter λ in GFManifold is determined by the validation set. As in Figure 7, GFManifold with $\lambda = 1$ or $\lambda = 0$ exhibits performance degradation. This demonstrates that **removing either the domain alignment MMD loss or the geodesic link prediction loss degrades the performance**, thereby validating the effectiveness of GFManifold's multi-domain pre-training strategies.

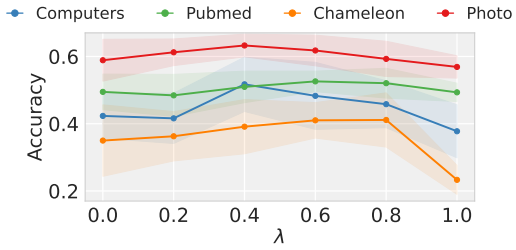


Figure 7: **Hyperparameter analysis.** GFManifold achieves optimal performance when λ is neither 0 nor 1, demonstrating that both the domain alignment MMD loss and the geodesic link prediction loss contribute to its overall effectiveness.

F.7 Further Visualisation

This section presents additional visualisation results for GFManifold to verify the effectiveness of the adopted manifolds. As shown in Figure 8, GFManifold effectively improves the class separation of the GFM output test embeddings, particularly on Citeseer and Photo. Even on a challenging heterophilic dataset, Squirrel, the embeddings produced by GFManifold remain more compact within each class, as illustrated by the blue class, whereas the embeddings without the manifold are more evenly distributed.

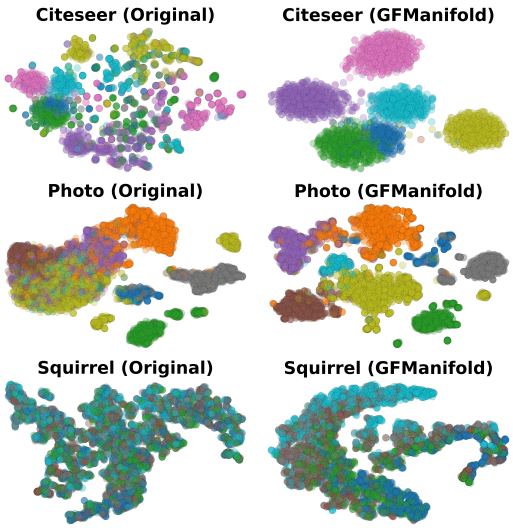


Figure 8: **Further visualisations for the effect of the manifold on the embedding separability.** Clearer class separation is consistently observed in the GFManifold output test node embeddings for target domain graphs. Datasets such as Texas, Wisconsin and Cornell are excluded because they are too small for visualisation.

G LIMITATIONS

GFManifold employs two globally shared constant-curvature manifold branches and does not explicitly model node- or region-dependent curvature. This design prioritises globally shared source-learnt manifold parameters and tuning-free downstream inference. GFManifold uses manifold geometry to derive structural edge weights while retaining the GNN message-passing backbone and restricting structural adaptation to existing graph connections. Extending GFManifold with conditional local geometric relations without introducing target-specific optimisation is a meaningful direction yet beyond the current scope.

Moreover, GFManifold follows the consistent paradigm of prior GFM, such as SAMGPT (Yu et al., 2025) and MDGFM (Wang et al., 2025), to focus on GNN-based architectures as the backbone. Consequently, the proposed GFManifold is primarily designed for general graph data where node features are continuous. LLM-based GFMs are not the target in this work, as they are typically restricted to text-attributed graphs and rely on language modelling capabilities that are not directly applicable to broader graph modalities. In contrast, GNN-based GFMs exhibit wider applicability across diverse graph domains, including social, biological, and citation networks. Therefore, this work prioritises the study of GNN-based GFMs to ensure broader generalisability. Extending GFManifold to integrate LLM-based GFMs, particularly for text-rich graph settings, could be a potential direction for future research, yet it is beyond the scope of this work.

H IMPACT STATEMENT

This paper introduces GFManifold, a novel graph foundation model framework utilising non-Euclidean Riemannian manifolds to enable efficient, tuning-free adaptation across diverse graph domains. As this work focuses on theoretical advancements in cross-domain graph representation learning, it does not present any foreseeable ethical risks or negative societal impacts.