Riemannian Graph Tokenizer for Structural Knowledge Transfer

Qimin Zhou
Beijing University of Posts and Telecommunications
Beijing, China
zhouqimin@bupt.edu.cn

Haibo Liu
Beijing University of Posts and Telecommunications
Beijing, China
lhb0410@bupt.edu.cn

Yujie Wang
Beijing University of Posts and Telecommunications
Beijing, China
yujiewang@bupt.edu.cn

Li Sun*
Beijing University of Posts and Telecommunications
Beijing, China
lsun@bupt.edu.cn

Chuan Shi*
Beijing University of Posts and Telecommunications
Beijing, China
shichuan@bupt.edu.cn

Abstract

Foundation models are at the forefront of artificial intelligence. A tokenizer, converting the raw input into discrete representations that the model can understand, plays an important role to the success of foundation models. Unlike the text tokenizer that is well studied in large language models, graph tokenizer is still at its early stage, facing the challenges of tackling the non-Euclidean structures and capturing the structural semantics. **How to design a graph tokenizer for structural knowledge transfer?** To this end, we propose a Riemannian Graph Tokenizer (**RGT**) that bridges the structural knowledge and quantized representations to support cross-domain structural knowledge transfer. The connection is established by Riemannian geometry. Specifically, we first define the geometric vocabulary (trees, cycles and sequences), which captures fundamental structural patterns and reflects the intrinsic geometry of graph. Second, we construct a Riemannian quantizer with Riemannian Straight-Through Estimator to tokenise graph structures across multiple domains into discrete tokens. To ensure consistency and transferability across diverse geometric spaces, RGT further incorporates a geometry-aligned decoder that projects manifold-specific tokens into a unified tangent space. The theoretical analysis and geometric interpretations are provided to support the effectiveness of our proposed method. Extensive experiments across diverse datasets demonstrate that RGT significantly enhances structural knowledge transferability across graph domains.

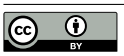
CCS Concepts

• **Computing methodologies** → **Knowledge representation and reasoning**; • **Theory of computation** → **Computational geometry**.

Keywords

Graph Tokenizer, Riemannian Geometry, Structural Knowledge Transfer

*Corresponding author.



WWW '26, Dubai, United Arab Emirates.

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792120

ACM Reference Format:
Qimin Zhou, Haibo Liu, Yujie Wang, Li Sun, and Chuan Shi. 2026. Riemannian Graph Tokenizer for Structural Knowledge Transfer. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792120

1 Introduction

Graph is a universal language for describing and modeling complex systems [51, 61]. Graphs data are ubiquitous in the real world, such as molecular graph [9], social graph [50] and citation networks [18]. Graph machine learning [52] aims to harness machine learning techniques [1] to uncover latent information in graphs and enhance the performance of downstream tasks. Among various graph machine learning approaches, Graph Neural Networks (GNNs) [12, 46, 54] have received increasing attention in recent years due to their broad applicability to various Web-related problems [27] and strong empirical performance [23, 26]. However, most existing GNNs methods are typically tailored to specific datasets or tasks, lacking generalization across domains and tasks. Consequently, Graph Foundation Models (GFMs) [13, 28, 58] are emerging as an interesting research topic in the graph domain.

GFMs [24] refer to pretrained models across diverse graph datasets that support multiple downstream tasks. The GFMs proposed by researchers primarily address the aforementioned limitations of GNNs. Some methods leverage the message-passing mechanism of GNNs to aggregate information. All in One [42] proposes a novel multi-task prompting approach to unify downstream tasks. GCOPE [59] further conducts model training on multi-domain graphs with coordinators in order to improve the generalization capacity to different datasets. Recently, GraphAny [60] models inference on a new graph as an analytical solution to a LinearGNN, which can be naturally applied to graphs with any feature and label. Moreover, some methods align graph structural knowledge with textual spaces. A representative example is OFA [22], which encodes textual attributes using pretrained language models to support various graph tasks. ZeroG [21] further enables zero-shot transfer across graph datasets by aligning semantic and structural features through textual descriptions of nodes and classes. In addition, GFT [49] aligns the structural information of computation tree vocabulary with textual representations through a quantization-based approach. Notably, recent research has begun to explore the

transfer of structural knowledge in graphs. For example, OpenGraph [53] presents a unified graph tokenizer and scalable graph transformer to encode graph structures. It also utilizes large language models to enhance graph features. RiemannGFM [40] learns structural knowledge solely through structural vocabulary, enabling cross-domain generalization without relying on textual features. GFMs have made remarkable progress, yet transferring structural knowledge across graphs remains a significant challenge.

In contrast to GFMs, the foundation models pretrained on large-scale text or image data [29, 48] have demonstrated properties such as emergence and homogenization. Their tokenizers [4, 19] serve as critical bridges between raw data and learned representations, significantly enhancing model generalization and cross-domain transferability [5]. Compared with text or image, graph has richer structural information and topological heterogeneity [11, 38, 41]. GFT and OpenGraph demonstrate that graph tokenizer plays a critical role in addressing the challenge of GFMs. However, research on graph tokenizer remains in its infancy, facing significant challenges in handling multi-domain non-Euclidean structures and capturing structural semantics for effective structural knowledge transfer. Specially, one problem lies in the substantial structural differences among graphs from different domains. Existing models often lack the ability to model and recognize transferable structural patterns. While computation trees [6] and structural vocabulary [40] provide partial solutions, relying solely on trees and cycles proves insufficient to represent the full diversity of graph structures. Another key limitation is that current GFMs cannot effectively perform structure-level knowledge transfer across graph domains. Tokenizer-based mechanisms have shown promise in compressing semantics into discrete and transferable tokens in shallow spaces [44, 47]. But existing efforts [49] either depend heavily on textual alignment to facilitate tokenization and structural transfer, or distill GNNs into lightweight MLPs [55]. Despite these advancements, the potential of tokenizer for enabling structure-level knowledge transfer remains largely underexplored. This naturally leads to a key question: *How can we design a graph tokenizer that supports the structural knowledge transfer?*

In this paper, inspired by the success of tokenizers in large language models and the expressive power of Riemannian geometry [20, 39] in modeling non-Euclidean graph structures [10, 33, 36, 37]. We propose a novel **R**iemannian **G**raph **T**okenizer (RGT) that quantizes graph structural features into discrete tokens embedded in constant curvature spaces to support cross-domain structural knowledge transfer. Specifically, we sample geometric vocabulary as geometric structural patterns, and utilize a Riemannian encoder to learn the coordinates of trees, cycles and sequences in hyperbolic [34], spherical [30] and Euclidean [43] spaces, respectively. The coordinates of the geometric vocabulary are quantized using codebooks. We develop a Riemannian straight-through estimator to support gradient-based optimization in Riemannian spaces. The node representations are then parallel transported to the tangent space at the pole. Finally, the decoder reconstructs the original structural features, while reconstruction loss and contrastive loss are adopted to encourage the learning of structurally transferable representations. Our main contributions are summarized as follows:

- **Geometric Vocabulary.** The geometric vocabulary (trees, cycles and sequences) serves as fundamental structural patterns that facilitate structure-level knowledge transfer.
- **Riemannian Graph Tokenizer.** We propose RGT that quantizes graph structures across multiple domains within constant curvature spaces. Among its components, the Riemannian quantizer preserves the intrinsic geometry of non-Euclidean manifolds and is supported by theoretical analysis.
- **Extensive Evaluation.** We conduct thorough theoretical analysis and experiments on multiple real-world graph datasets, showing that RGT achieves competitive performance on graph tasks and exhibits significant cross-domain generalization.

2 Preliminaries

Riemannian Geometry. For a smooth manifold $\mathcal{M}$, each point $\boldsymbol{x} \in \mathcal{M}$ is associated with a tangent space $T_{\boldsymbol{x}}\mathcal{M}$, which serves as the first-order approximation of the manifold near $\boldsymbol{x}$ and locally exhibits Euclidean structure. A Riemannian metric $\mathfrak{g}_{\boldsymbol{x}}(\cdot,\cdot) : T_{\boldsymbol{x}}\mathcal{M} \times T_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$ is defined on the tangent space $T_{\boldsymbol{x}}\mathcal{M}$, endowing it with an inner product. This allows geometric concepts such as lengths, angles, and distances to be meaningfully defined on the manifold. The tuple $(\mathcal{M}, \mathfrak{g})$ is thus called a Riemannian manifold. The curvature $\kappa_{\boldsymbol{x}}$ measures the deviation of the manifold at point $\boldsymbol{x}$ from flat Euclidean space. If a manifold has constant curvature $\kappa_{\boldsymbol{x}} = \kappa$ for all $\boldsymbol{x} \in \mathcal{M}$, it is called a Constant Curvature Space (CCS).

Constant Curvature Space. We unify the Lorentz, Spherical and Euclidean space under a unified κ-CCS space, denoted by

$$\mathcal{L}_\kappa^d := \begin{cases} \left\{ \boldsymbol{x} = [x_t; \boldsymbol{x}_s] \in \mathbb{R}^{d+1} \mid \kappa \langle \boldsymbol{x}, \boldsymbol{x} \rangle_\kappa = 1,\ x_t > 0,\ \boldsymbol{x}_s \in \mathbb{R}^d \right\}, & \kappa \neq 0, \\ \left\{ \boldsymbol{x} = [x_t; \boldsymbol{x}_s] \in \mathbb{R}^{d+1} \mid \langle \boldsymbol{x}, \boldsymbol{x} \rangle_\kappa \geq 0, \boldsymbol{x}_t = 0, \boldsymbol{x}_s \in \mathbb{R}^d \right\}, & \kappa = 0, \end{cases} \tag{1}$$

where the inner product is defined as: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\kappa := \mathrm{sgn}(\kappa) x_t y_t + \boldsymbol{x}_s^\top \boldsymbol{y}_s$, d and κ denote the dimension and curvature. The Riemannian metric at $\boldsymbol{x}$ is $\mathfrak{g}_x = \mathrm{diag}(\mathrm{sgn}(k), 1, \ldots, 1)$. x_t corresponds to the axis of symmetry of the hyperboloid and is termed the time-like dimension, while $\boldsymbol{x}_s \in \mathbb{R}^d$ denotes the space-like dimensions. This formulation recovers the following standard models under different curvature regimes. $\kappa < 0$: corresponds to the **Lorentz space** of hyperbolic geometry. $\kappa > 0$: corresponds to the **Spherical space**. $\kappa = 0$: corresponds to the **Euclidean space**. We define the "north pole" of the manifold as: $\boldsymbol{o}_\kappa = [\frac{1}{\sqrt{|\kappa|}}, 0, \cdots, 0]^\top$. **The tangent space at the north pole** is given by $T_{\boldsymbol{o}_\kappa}\mathcal{L}_\kappa^d = \left\{ \boldsymbol{v} = [0; \boldsymbol{v}_s] \in \mathbb{R}^{d+1} \right\} \cong \mathbb{R}^d$. Closed-form exponential and logarithmic maps exist.

Notations & Problem Formulation. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges and $\mathcal{D}$ denotes the domain. Each node $v \in \mathcal{V}$ may optionally be associated with an attribute vector $\boldsymbol{x}_v \in \mathbb{R}^d$. The graph tokenizer $\mathcal{T}_\kappa$ is pretrained on multi-domain graph $\mathcal{G}$ via self-supervised learning. It is parameterized by $\boldsymbol{\theta}$ and associated with a discrete codebook $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m\}$ defined in a constant curvature space, where κ denotes the curvature of the space. Formally, given a subgraph or structural pattern $\mathcal{G}_s \subseteq \mathcal{G}$, the pretrained tokenizer encodes graph features for downstream tasks: $\mathcal{T}_\kappa : \mathcal{G}_s \mapsto \boldsymbol{q}_s \in C$, where the tokenizer maps the input structural

pattern to a geometry-aware token $\boldsymbol{q}_s$ selected from the codebook C, which lies on the manifold $\mathcal{L}_\kappa^d$.

The proposed graph tokenizer is designed with two core capabilities. **Structure Awareness:** The model is expected to learn generalizable representations of fundamental structural patterns such as trees, cycles and sequences. **Structural Knowledge Transfer:** The model should be pretrained on graph data from multiple domains and adapted with minimal parameter updates to new graph domains with varying structural or attribute distributions, thereby enabling generalized graph representations across domains.

3 RGT: Riemannian Graph Tokenizer for Structural Knowledge Transfer

This section introduces the proposed RGT in detail, which is tailored for self-supervised pretraining on graph structures. The model discretizes structural features into geometry-aware representations. It supports both attributed and non-attributed graphs, enabling effective cross-domain transfer. Definition 1 formalizes a complete geometric vocabulary that supports the patterns of arbitrary graph structures.

Definition 1 (Geometric Vocabulary). *A set of graph substructures $S = \{S_1, S_2, \ldots, S_K\}$ is called* geometric vocabulary *if it satisfies the following two conditions:*

- ***Structural Completeness:*** *For any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, there exists a multiset of substructures $\{S_i^{(j)}\}_{j=1}^M$, where $S_i^{(j)} \in S$, such that: $\mathcal{G} = \bigcup_{i=1}^K \bigcup_{j=1}^M S_i^{(j)}$. The value of M can vary across different S_i, as each structural type may contribute a different number of instances in the composition of the graph.*
- ***Geometric Representability:*** *Each element $S_i \in S$ can be embedded into a Riemannian manifold $\mathcal{M}_\kappa^d$ with constant curvature $\kappa \in \mathbb{R}$, such that the embedding preserves its structural geometry (e.g., positive, negative or zero curvature).*

Then, the set S is defined as geometric vocabulary *for graph structures.*

In constant curvature spaces [2], trees are modeled in hyperbolic space, cycles in spherical space and sequences in Euclidean space. This modeling paradigm effectively bridges topological heterogeneity [11, 13]. Based on the definition of geometric vocabulary, we leverage simple trees, cycles and sequences as the core structural patterns (detailed in Appendix A).

Overall Architecture. This work proposes the Riemannian graph tokenizer. The overall architecture is illustrated in Figure 1. First, geometric vocabulary is sampled from the input graph and structural coordinates are encoded in constant curvature space via Riemannian Encoder (Fréchet mean method) [57]. A novel Riemannian quantizer is designed to discretize the structural features and learn the structural codebooks. The quantized tokens are subsequently parallel transported or projected to the tangent space at the north pole to enable unified geometric fusion and decoding. Finally, the entire model is trained using a joint objective that combines reconstruction loss and contrastive loss.

3.1 Riemannian Graph Tokenizer

RGT consists of three core components: the geometric vocabulary encoder, the Riemannian quantizer and the geometry-aligned decoder. We describe each of these modules in detail as follows.

3.1.1 Geometric Vocabulary Encoder. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the Laplacian matrix is $L \in \mathbb{R}^{n\times n}$. We perform eigenvalue decomposition: $L = U\Lambda U^\top$, where $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$ is the orthogonal matrix of eigenvectors, the $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues. We select the eigenvectors corresponding to the top d eigenvalues with the smallest magnitudes to construct a structural feature matrix $\boldsymbol{X}$, where each row of $\boldsymbol{x}$ serves as the initial structural feature of a node.

The structural feature $\boldsymbol{x}$ is projected via a linear transformation ψ to obtain $\boldsymbol{x}^{\mathcal{R}}$ in Euclidean space. For the hyperbolic and spherical spaces, we apply exponential mapping to project $\boldsymbol{x}^{\mathcal{R}}$ into the corresponding manifold as the initial coordinate. Therefore, the coordinate of a node can be uniformly denoted as $\boldsymbol{x}^{\mathcal{M}}, \mathcal{M} \in \{\mathcal{R}, \mathcal{H}, \mathcal{S}\}$ and its feature in the tangent space at the current point is denoted as $\boldsymbol{e}^{\mathcal{M}}$. We adopt the BFS algorithm [7] to sample geometric vocabulary (trees, cycles and sequences) from the graph. The coordinates of these structural features are then updated via $\boldsymbol{z}^{\mathcal{M}} = \mathbf{E}(\boldsymbol{x}^{\mathcal{M}})$, where $\boldsymbol{x}^{\mathcal{M}}, \boldsymbol{z}^{\mathcal{M}} \in \mathcal{M}$, $\mathbf{E}$ denotes the geometric vocabulary encoder. The encoding procedure is detailed as follows:

For trees and cycles, we adopt the Fréchet Mean to update the coordinates of geometric vocabulary [25, 40]. Specifically, attention weights are computed by cross-geometry graph attention mechanism defined in Eq.(2). The coordinate of each node is aggregated via the Fréchet mean on the corresponding manifold. The encoded coordinate is given by the geometric center $\boldsymbol{z}_i^{\mathcal{M}}, \mathcal{M} \in \{\mathcal{H}, \mathcal{S}\}$.

$$\alpha_{ij} = \frac{\exp\left(\phi\left([\boldsymbol{r}_i \,\|\, \boldsymbol{k}_j]\right)\right)}{\sum_{(i,t)\in S_i} \exp\left(\phi\left([\boldsymbol{r}_i \,\|\, \boldsymbol{k}_t]\right)\right)}, \tag{2}$$

$$\boldsymbol{z}_i^{\mathcal{M}} = \mathrm{mid}_\kappa\left(\left\{\boldsymbol{z}_j^{\mathcal{M}}, \alpha_{ij}\right\}_{(i,j)\in S_i}\right), \tag{3}$$

where j denotes a descendant node of node i, ϕ is an arbitrary scalar-valued function and $\mathrm{mid}_\kappa(\cdot)$ denotes the Fréchet mean on the manifold. In the cross-geometry attention mechanism, the key, query and value are computed through manifold-preserving linear operations f_* [40] are denoted as:

$$\boldsymbol{k}_i = f_V(\boldsymbol{z}_i^{\mathcal{H}}), \quad \boldsymbol{r}_i = f_Q(\boldsymbol{z}_i^{\mathcal{S}}), \quad \boldsymbol{v}_i = f_V(\boldsymbol{z}_i^{\mathcal{H}}), \tag{4}$$

the coordinate $\boldsymbol{z}_i^{\mathcal{M}}$ of a node in the manifold space is obtained via Eq.(3).

Geometric Center in Euclidean Space. For a sequence of length L, let the structural features of node v_j be denoted as $\{\boldsymbol{z}_j^{\mathcal{R}}\}_{j=1}^L \subset \mathbb{R}^d$, with corresponding attention weights $\{w_{ij}\}_{j=1}^L$ satisfying $w_{ij} \geq 0$ and $\sum_{j=1}^L w_{ij} = 1$. Eq.(2) is used to compute w_{ij} within a single geometric space. The weighted geometric center (Fréchet Mean) in Euclidean space is then defined as:

$$\boldsymbol{z}_i^{\mathcal{R}} = \frac{\sum_{j=1}^L w_{ij}\boldsymbol{z}_j^{\mathcal{R}}}{\left\|\sum_{j=1}^L w_{ij}\boldsymbol{z}_j^{\mathcal{R}}\right\|_2}. \tag{5}$$

As a result, we obtain the coordinates of the graph in three different spaces: $\boldsymbol{z}^{\mathcal{H}}, \boldsymbol{z}^{\mathcal{S}}$ and $\boldsymbol{z}^{\mathcal{R}}$.

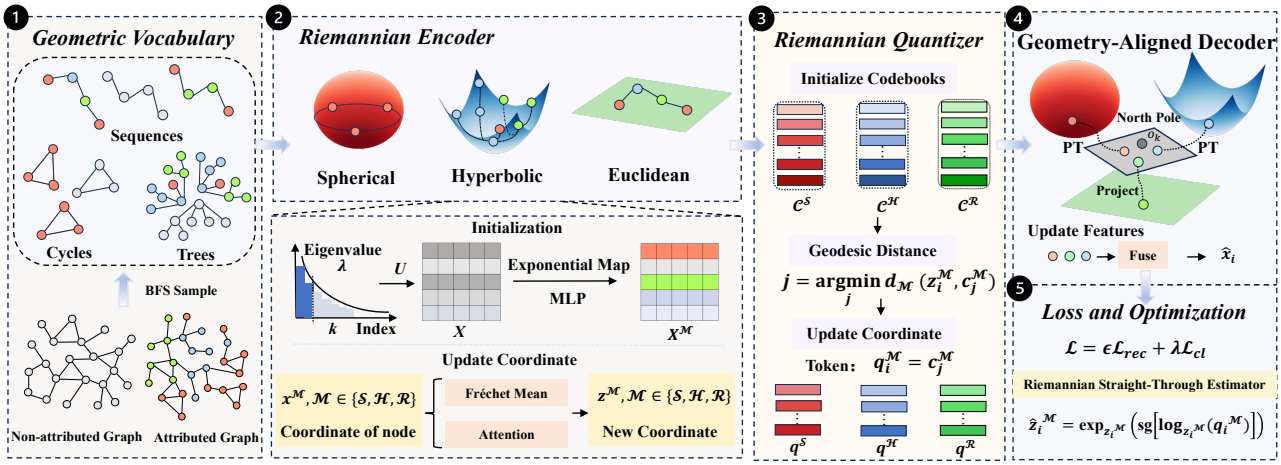


Figure 1: Overall architecture of the proposed Riemannian Graph Tokenizer.

3.1.2 *Riemannian Quantizer.*

Riemannian Quantizer. To enhance the model's ability to transfer structural knowledge across diverse graph domains, we design Riemannian Quantizer (RQ) that discretizes geometric vocabulary at the structural level. Specifically, for each constant curvature space $\mathcal{M} \in \{\mathcal{R}, \mathcal{H}, \mathcal{S}\}$, we initialize the codebook $C^{\mathcal{M}}$ with m codes, where each code vector corresponds to a point on the manifold. For the i-th node, its coordinate in space $\mathcal{M}$ is denoted as $z_i^{\mathcal{M}}$. Following Eq.(6), we select the code $\boldsymbol{q}_i^{\mathcal{M}} \in C^{\mathcal{M}}$ from the codebook that has the smallest geodesic distance to $z_i^{\mathcal{M}}$. The core procedure is as follows:

$$\text{RQ}: \begin{cases} j = \arg\min_j d_{\mathcal{M}}(z_i^{\mathcal{M}}, \boldsymbol{c}_j^{\mathcal{M}}), \\ \boldsymbol{q}_i^{\mathcal{M}} = \boldsymbol{c}_j^{\mathcal{M}}, \end{cases} \tag{6}$$

where $d_{\mathcal{M}}(z_i^{\mathcal{M}}, \boldsymbol{c}_j^{\mathcal{M}})$ is geodesic distance. $\boldsymbol{c}_j^{\mathcal{M}} \in C^{\mathcal{M}}$ denotes the j-th code in the codebook. The nearest code is selected as $\boldsymbol{q}_i^{\mathcal{M}} = \boldsymbol{c}_j^{\mathcal{M}}$.

To provide a theoretical foundation for the feasibility of RQ, we present two theorems: Cross-domain Encoding Consistency and Optimality of Riemannian Quantization. The details are as follows:

Theorem 1 (Cross-domain Encoding Consistency). *Let $z_i^{\mathcal{M}} \in D_1$, $z_j^{\mathcal{M}} \in D_2$, where $D_1, D_2 \in \mathcal{D}$, if $d(z_i^{\mathcal{M}}, z_j^{\mathcal{M}}) \le \epsilon$, then the following holds:*

$$d(RQ(z_i^{\mathcal{M}}), RQ(z_j^{\mathcal{M}})) \le 3\epsilon.$$

The theorem 1 establishes that RQ preserves local consistency, we now show that RQ also enjoys a global optimality on Riemannian manifolds.

Theorem 2 (Optimality of Riemannian Quantization). *Let $(\mathcal{M}, g)$ be a smooth Riemannian manifold equipped with a geodesic distance function $d_{\mathcal{M}}(\cdot,\cdot)$. Given a set of geometric vocabulary $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$ and a structural codebook $C = \{\boldsymbol{c}_j^{\mathcal{M}}\}_{j=1}^{m} \subset \mathcal{M}$, the quantizer $RQ : \{\mathcal{S}_i^{(j)}\}_{j=1}^{M} \to C$ is defined as:*

$$\boldsymbol{c}_j^{\mathcal{M}} = \arg\min_{\boldsymbol{c} \in C} \mathbb{E}\left[d_{\mathcal{M}}(\boldsymbol{z}, \boldsymbol{c}) \,\middle|\, \boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M} \right], \tag{7}$$

the quantizer RQ minimizes the overall expected geodesic distortion $\mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, RQ(\boldsymbol{z}))]$, where $\boldsymbol{z} \sim P(\{\mathcal{S}_i^{(j)}\}_{j=1}^{M})$ denotes the probability distribution over structural tokens within the geometric vocabulary set $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$.

Geometric Interpretation 1. The theorem 1 establishes that RQ preserves the neighborhood relations of geometric vocabularies across different domains at the local level, thereby ensuring that structurally similar patterns remain close after quantization. Furthermore, the theorem 2 provides a global guarantee by showing that selecting the codeword for each geometric vocabulary $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$ as its geodesic center on the manifold $\mathcal{M}$ minimizes the quantization error. Together, these theoretically validate that the RQ formulation in Eq. (6) offers an optimal structure-level discretization strategy: structurally similar geometric vocabularies (such as trees, cycles, and sequences) can be consistently mapped to the same code in the corresponding codebook, thereby enabling cross-domain structural alignment. The detailed proofs are provided in Appendix A.

Riemannian Straight-Through Estimator. To maintain the intrinsic structure of the manifold while supporting end-to-end differentiable training, we propose the Riemannian Straight-Through Estimator (RSTE), formulated as Eq.(8):

$$\hat{z_i}^{\mathcal{M}} = \exp_{z_i^{\mathcal{M}}}\left(\text{sg}\left[\log_{z_i^{\mathcal{M}}}(\boldsymbol{q}_i^{\mathcal{M}})\right]\right), \quad \boldsymbol{q}_i^{\mathcal{M}} \in \mathcal{M}, \tag{8}$$

where the operator $\log_{z_i^{\mathcal{M}}}(\boldsymbol{q}_i^{\mathcal{M}})$ denotes the logarithmic map that projects the point $\boldsymbol{q}_i^{\mathcal{M}}$ onto the tangent space at $z_i^{\mathcal{M}}$, while $\exp_{z_i^{\mathcal{M}}}(\cdot)$ denotes the exponential map that lifts a tangent vector back onto the manifold. $\text{sg}[\cdot]$ represents the stop-gradient operator.

Geometric Interpretation 2. In Euclidean space, vector quantizer [44] typically adopts $\boldsymbol{z}^{\mathcal{R}} = \boldsymbol{z}^{\mathcal{R}} + \text{sg}[\boldsymbol{q}^{\mathcal{R}} - \boldsymbol{z}^{\mathcal{R}}]$ as a Straight-Through Estimator (STE) [3], where $\boldsymbol{q}^{\mathcal{R}}$ is treated as a constant during back-propagation and does not receive gradient updates. However, this estimator fails to preserve the intrinsic structure of non-Euclidean manifolds. We propose the RSTE to address this limitation. The advantage of this method lies in the fact that the quantization error $\log_{z_i}^{\mathcal{M}}(q_i^{\mathcal{M}})$ is retained in the tangent space, where the geometric structure is more stable. Since gradients are propagated within the tangent space, the manifold $\mathcal{M}$ maintains geometric consistency.

It prevents the distortion typically introduced by Euclidean additive operations. This effectively alleviates the issue of codebook collapse [8, 62]. The multiple similar geometric vocabularies are aligned to the same code in the codebook. Quantization in Riemannian space facilitates structural knowledge transfer. Detailed analyses are provided in Appendix B.

3.1.3 Geometry-Aligned Decoder. As the quantized tokens reside in different geometric spaces, we propose a geometry-aligned decoder $\hat{\boldsymbol{x}}_i^{\mathcal{M}} = \mathrm{D}(\boldsymbol{q}_i^{\mathcal{M}})$ to obtain consistent representation and ensure spatially aligned reconstruction and contrast. For the Euclidean-space token $\boldsymbol{q}_i^{\mathcal{R}}$, we first project it to the tangent space at the north pole to obtain $\boldsymbol{e}_i^{\mathcal{R}}$. Then, for hyperbolic and spherical spaces, the Euclidean feature $\boldsymbol{q}_i^{\mathcal{R}}$ is projected into the tangent space at $\boldsymbol{q}_i^{\mathcal{M}}(\mathcal{M} \in \mathcal{H}, \mathcal{S})$, resulting in $\boldsymbol{e}_i^{\mathcal{M}}$. Riemannian parallel transport is utilized to align $\boldsymbol{e}_i^{\mathcal{M}}$ from the point $\boldsymbol{q}_i^{\mathcal{M}}(\mathcal{M} \in \mathcal{H}, \mathcal{S})$ to the tangent space $T_{\boldsymbol{o_k}}\mathcal{M}$ at the north pole. The aligned representation in the tangent space, $\boldsymbol{e}_i^{T_{\boldsymbol{o_k}}\mathcal{M}}$, is defined as follows:

$$\boldsymbol{e}_i^{T_{\boldsymbol{o_k}}\mathcal{M}} = PT_{\boldsymbol{q}_i^{\mathcal{M}} \to \boldsymbol{o_k}}\left(\boldsymbol{e}_i^{\mathcal{M}}\right), \tag{9}$$

where $PT_{\boldsymbol{q}_i^{\mathcal{M}} \to \boldsymbol{o_k}}(\cdot)$ denotes the parallel transport operation from a point $\boldsymbol{q}_i^{\mathcal{M}}$ on the manifold to the north pole $\boldsymbol{o_k}$. The resulting embeddings in the tangent space at the north pole are denoted as $\boldsymbol{e}_i^{T_{\boldsymbol{o_k}}\mathcal{H}}$ and $\boldsymbol{e}_i^{T_{\boldsymbol{o_k}}\mathcal{S}}$ for the hyperbolic and spherical spaces, respectively. Next, the aligned embeddings from different manifolds are concatenated and projected back to the original space. Geometry-aligned representations from all spaces are fused into a single unified representation:

$$\hat{x}_i = \mathbf{Fuse}\left(\boldsymbol{e}_i^{\mathcal{R}}, \boldsymbol{e}_i^{T_{\boldsymbol{o_k}}\mathcal{H}}, \boldsymbol{e}_i^{T_{\boldsymbol{o_k}}\mathcal{S}}\right), \tag{10}$$

where $\mathbf{Fuse}(\cdot)$ denotes a simple fusion operation, such as concatenation followed by a linear transformation, and $\hat{\boldsymbol{x}}_i$ represents the reconstructed structural feature. The geometry-aligned decoder enables unified decoding of embeddings across geometric spaces with varying curvatures, ensuring consistency in representation.

3.2 Loss Function and Optimization

Loss Function. We design a unified loss function to jointly optimize the Riemannian graph tokenizer across constant curvature spaces. Each geometric space maintains an independent codebook. The loss function integrates both reconstruction loss $\mathcal{L}_{rec}$ and geometric contrastive loss $\mathcal{L}_{cl}$.

$$\mathcal{L} = \epsilon\mathcal{L}_{rec} + \lambda\mathcal{L}_{cl}, \tag{11}$$

The reconstruction loss and geometric contrastive loss are formulated in Eq.(12) and Eq.(13):

$$\begin{aligned}\mathcal{L}_{rec} = \frac{1}{n}\sum_{i=1}^{n}\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 + \frac{1}{n}\sum_{\mathcal{M}}\Bigg[\beta\sum_{i=1}^{n} d^2_{\mathcal{M}}(\boldsymbol{z}_i^{\mathcal{M}}, \mathrm{sg}(\boldsymbol{q}_i^{\mathcal{M}})) \\ + \gamma\sum_{i=1}^{n} d^2_{\mathcal{M}}(\mathrm{sg}(\boldsymbol{z}_i^{\mathcal{M}}), \boldsymbol{q}_i^{\mathcal{M}})\Bigg],\end{aligned} \tag{12}$$

$$\mathcal{L}_{cl} = \underbrace{[\mathcal{J}(\mathcal{R},\mathcal{H}) + \mathcal{J}(\mathcal{R},\mathcal{S}) + \mathcal{J}(\mathcal{H},\mathcal{S})]}_{\text{Cross-geometry contrastive loss}}, \tag{13}$$

$$\mathcal{J}(\mathcal{H},\mathcal{S}) = -\sum_{i=1}^{N}\log\frac{\exp\left(\left\langle PT_{\boldsymbol{q}_i^{\mathcal{H}} \to \boldsymbol{o_k}}\left(\boldsymbol{e}_i^{\mathcal{H}}\right), PT_{\boldsymbol{q}_i^{\mathcal{S}} \to \boldsymbol{o_k}}\left(\boldsymbol{e}_i^{\mathcal{S}}\right)\right\rangle\right)}{\sum_{j=1}^{N}\exp\left(\left\langle PT_{\boldsymbol{q}_i^{\mathcal{H}} \to \boldsymbol{o_k}}\left(\boldsymbol{e}_i^{\mathcal{H}}\right), PT_{\boldsymbol{q}_j^{\mathcal{S}} \to \boldsymbol{o_k}}\left(\boldsymbol{e}_j^{\mathcal{S}}\right)\right\rangle\right)}, \tag{14}$$

where the $\boldsymbol{q}_i^{\mathcal{M}}$ denotes the representation of the i-th node in the geometric space $\mathcal{M}$. After parallel transport, it is projected into the shared tangent space to obtain the fused embedding $\hat{\boldsymbol{x}}_i$. The term $\mathcal{J}(H, S)$ in Eq.(13) denotes the contrastive learning loss computed in the shared tangent space, encouraging alignment between representations from different geometric spaces. The other two follow similarly. The reconstruction loss is computed by comparing the reconstructed representation $\hat{\boldsymbol{x}}_i$ with the original input $\boldsymbol{x}_i$.

Optimization. We update each variable. First, Eq.(15) is the gradient propagation for the $\boldsymbol{z}_i^{\mathcal{M}}$:

$$\begin{aligned}\boldsymbol{z}_i^{\mathcal{M},(t+1)} =& \boldsymbol{z}_i^{\mathcal{M},(t)} - \eta\Bigg[\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i}\cdot\frac{\partial\hat{\boldsymbol{x}}_i}{\partial\hat{\boldsymbol{z}}_i^{\mathcal{M}}}\cdot\frac{\partial\exp^{\mathcal{M}}_{\boldsymbol{z}_i^{\mathcal{M}}}\left(\mathrm{sg}\left[\log_{\boldsymbol{z}_i^{\mathcal{M}}}(\boldsymbol{q}_i^{\mathcal{M}})\right]\right)}{\partial\boldsymbol{z}_i^{\mathcal{M}}} \\ &+ \beta\cdot\frac{\partial}{\partial\boldsymbol{z}_i^{\mathcal{M}}}d^2_{\mathcal{M}}(\boldsymbol{z}_i^{\mathcal{M}}, \mathrm{sg}(\boldsymbol{q}_i^{\mathcal{M}})) + \lambda\cdot\frac{\partial\mathcal{L}_{cl}}{\partial\boldsymbol{z}_i^{\mathcal{M}}}\Bigg].\end{aligned} \tag{15}$$

Eq.(16) is the gradient propagation for the code $\boldsymbol{q}_i^{\mathcal{M}}$:

$$\boldsymbol{q}_i^{\mathcal{M},(t+1)} = \boldsymbol{q}_i^{\mathcal{M},(t)} - \eta(\gamma\frac{\partial}{\partial\boldsymbol{q}_i^{\mathcal{M}}}d^2_{\mathcal{M}}(\mathrm{sg}(\boldsymbol{z}_i^{\mathcal{M}}), \boldsymbol{q}_i^{\mathcal{M}}) + \lambda\frac{\partial\mathcal{L}_{cl}}{\partial\boldsymbol{z}_i^{\mathcal{M}}}), \tag{16}$$

where the t is the iteration. The above design ensures geometric consistency between structural tokens and discrete codes, with the training process strictly following curvature-aware geodesic paths. This enhances the model's stability and performance in structural transfer and cross-domain generalization.

The proposed model is pretrained using the combined loss Eq.(11). The overall training process is summarized in Algorithm 1 (Appendix C), with a computational complexity of $O(|\mathcal{V}|^2 + |\mathcal{E}|)$, where $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of nodes and edges, respectively. The Riemannian graph tokenizer effectively captures globally consistent structural patterns across diverse domains.

4 Experiment

In this section, we aim to answer the following research questions:
RQ1. How does RGT perform in cross-domain transfer learning?
RQ2. How effective is RGT under few-shot learning scenarios?
RQ3. How important is the use of geometric vocabulary in RGT?
RQ4. How does the choice of pretraining dataset influence the performance of RGT?
RQ5. How does RGT scale to web-scale graphs? Additional experiments are provided in Appendix D. **Codes** are available at https://github.com/Miyla77/RGT.git.

4.1 Experimental Setups

Hyperparameter settings: The number of layers for tree-based sampling is set to 2, cycles are constructed as triangles (3-node), and the sequence length is set to L. The embedding dimension d for structural embedding is 32. The curvatures of the constant curvature spaces κ are set to 1 (spherical), -1 (hyperbolic) and 0

Table 1: Cross-domain transfer learning performance of node classification on four datasets.

Method	Node Classification Results							
	Pubmed		Github		USA		Europe	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DGI	75.66±0.86	73.66±1.46	74.10±0.15	42.56±0.05	38.90±1.57	33.06±3.98	24.48±0.97	9.83±0.31
GraphMAE	**79.47±0.45**	**78.75±0.40**	67.57±2.76	63.14±1.81	32.78±6.63	21.42±5.39	31.28±2.10	20.96±3.47
OFA	52.23±6.45	50.35±3.34	-	-	-	-	-	-
ZeroG	69.12±0.89	68.35±1.23	-	-	-	-	-	-
Graphany	74.18±2.57	70.87±0.18	79.26±0.20	76.93±0.42	45.73±1.70	45.35±0.39	34.54±1.57	27.54±0.42
OpenGraph	68.11±1.98	65.45±3.33	47.85±0.54	46.34±0.78	40.65±1.38	42.89±1.14	27.36±1.63	25.74±0.73
GFT	70.30±1.76	70.38±1.83	-	-	-	-	-	-
RiemannGFM	73.94±5.66	74.25±4.25	85.51±0.39	85.24±0.15	55.63±2.33	55.92±2.22	36.50±4.68	30.77±4.21
RGT(Ours)	76.22±0.80	76.14±0.77	**86.35±0.14**	**85.67±0.24**	**56.72±2.51**	**55.80±3.29**	**37.75±3.39**	**32.17±1.48**

Table 2: Cross-domain transfer learning performance of link prediction on four datasets.

Method	Link Prediction Results							
	Pubmed		Github		USA		Europe	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DGI	56.19±0.19	61.56±0.35	55.75±8.68	53.66±7.79	56.17±2.31	48.39±1.35	58.82±5.38	51.38±3.30
GraphMAE	84.40±0.38	82.36±0.36	65.94±0.18	67.51±0.14	65.81±6.51	67.36±1.27	45.58±1.73	46.28±0.81
OFA	85.32±1.67	83.28±1.39	-	-	-	-	-	-
OpenGraph	65.04±0.98	64.31±3.98	78.34±0.92	79.76±1.20	77.89±2.34	79.32±1.32	72.34±1.51	70.32±1.22
RiemannGFM	89.84±0.32	88.99±0.27	88.49±0.22	87.78±0.17	**93.57±0.23**	**96.02±0.16**	**85.49±0.98**	**82.06±0.80**
RGT(Ours)	**93.28±0.01**	**91.92±0.01**	**90.31±0.03**	**88.90±0.02**	93.32±0.03	92.89±0.03	85.17±0.07	82.84±0.07

(Euclidean space). The feature dimension of each code is 32, the number of code in codebook $m = 128$. All hyperparameters are selected via grid search. For baselines, the hyperparameters strictly follow those reported in the original papers.

Pretraining and Evaluation: The pretraining datasets used for our model is the same as in RiemannGFM. During evaluation, the pretrained codebooks are kept frozen while the GNN classifier is fine-tuned on the target test datasets. Other models are evaluated on the same test datesets. The results are averaged over multiple runs to reduce the impact of randomness.

Datasets. We conduct experiments using several graph datasets, which are categorized based on their attribute types and application contexts. Specifically, we select three datasets ogbn-arxiv [15], Computers and Physics [35] for the pretraining phase. For the main evaluation experiments, we employ seven datasets: three text-attributed graphs (Cora, PubMed and Citeseer) [18], one mix-attributed graph (GitHub) [32], three non-attributed graphs (Airport of USA, Europe and Brazil) [31] and one Web-scale graph ogbn-products [16].

Baselines & Metrics. We choose seven competitive baselines, categorized into two groups. Self-supervised models: DGI [45] and GraphMAE2 [14]. Graph foundation models: OFA [22], ZeroG [21], GraphAny [60], OpenGraph [53], GFT [49], RiemannGFM [40]. Experiments cover node-classification and link-prediction settings. For node classification we report overall accuracy (ACC) together with Micro-F1 (F1). For link prediction we measure area under the AUC and average precision (AP). We report the mean and standard deviation over 5 seeds for each result. Best results are bolded. The dash means the model cannot handle non-attributed graphs.

4.2 Results and Discussion

Cross-domain Transfer Learning Performance (RQ1). The experimental results of node classification and link prediction are shown in Table 1 and Table 2, respectively. RGT demonstrates significant advantages. On citation networks dataset, RGT achieves comparable node classification performance to traditional self-supervised methods that are trained and evaluated within the same domain. Due to the incorporation of structural vocabulary, RGT yields superior results in link prediction. Meanwhile, RGT achieves better performance than existing GFMs on both node classification and link prediction tasks. This holds true even when models like GFT utilize multi-domain quantization on text-attributed graphs and OpenGraph incorporates graph tokenizer. The major strength of RGT lies in its ability to handle non-attributed graphs by structural features, where it outperforms RiemannGFM. These findings validate the effectiveness of structure-level quantization and the cross-domain transferability enabled by RGT.

Few-shot Learning Performance(RQ2). The results of one-shot and five-shot node classification are reported in Table 3 and Table 4, respectively. RGT demonstrates strong transferability in this low-resource setting. On the Pubmed dataset, methods such as ZeroG, Graphany and GFT leverage test-domain data during pretraining. In contrast, RGT performs strict cross-domain transfer without any exposure to the target domain. Nevertheless, RGT still exhibits competitive overall results. On the Github dataset, RGT achieves competitive performance on both evaluation metrics, showcasing its strong capacity to model complex graph structures. In the five-shot setting, as more structural information becomes available, the performance of RGT improves significantly. Except for the GraphMAE and GFT, RGT outperforms other methods, particularly on non-attributed graphs, where its advantage becomes more evident. These experimental results suggest that RGT effectively leverages limited supervision to capture semantic correlations between graph structures and class labels, making it well-suited for graph learning tasks under low-label regimes.

Table 3: One-shot learning performance of node classification on four datasets.

Method	Node Classification Results							
	Pubmed		Github		USA		Europe	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DGI	54.09±1.92	45.66±7.34	44.26±0.14	44.72±3.03	26.69±2.94	13.58±4.84	25.62±0.00	10.20±0.00
GraphMAE	47.03±7.27	42.31±9.40	55.25±2.03	46.78±1.74	23.69±1.70	14.73±2.83	27.07±3.20	18.54±2.30
OFA	41.25±0.94	38.45±2.96	-	-	-	-	-	-
ZeroG	**74.52±0.52**	**74.15±0.15**	-	-	-	-	-	-
Graphany	72.46±6.89	70.55±4.27	63.69±6.39	43.67±0.47	26.64±0.06	27.00±0.32	26.57±0.57	23.01±4.37
OpenGraph	66.96±5.34	63.86±4.88	32.67±0.12	45.23±1.21	30.45±3.71	29.32±2.46	22.46±3.89	20.65±3.22
GFT	73.56±3.15	74.55±4.70	-	-	-	-	-	-
RiemannGFM	44.92±4.28	37.28±3.54	64.32±5.25	63.96±4.94	34.34±5.53	34.29±5.55	27.58±4.89	21.57±5.27
RGT(Ours)	52.18±7.97	47.45±9.74	**72.28±6.74**	**70.39±8.00**	**36.13±5.67**	**33.34±3.09**	**28.75±6.57**	**21.94±6.75**

Table 4: Five-shot learning performance of node classification on four datasets.

Method	Node Classification Results							
	Pubmed		Github		USA		Europe	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DGI	64.07±8.43	61.88±4.84	75.56±1.37	51.21±7.40	35.84±1.28	28.02±3.20	25.62±0.00	10.20±0.00
GraphMAE	72.09±1.86	71.62±2.28	57.12±2.16	52.39±1.97	31.42±5.58	21.11±4.59	26.74±1.59	14.20±3.00
OFA	41.26±4.36	39.67±5.35	-	-	-	-	-	-
ZeroG	72.26±3.21	69.48±2.39	-	-	-	-	-	-
Graphany	62.82±6.16	60.28±4.15	54.52±10.07	44.66±0.47	27.69±0.08	28.93±0.25	29.54±4.24	23.21±5.12
OpenGraph	65.62±3.68	62.78±3.28	45.93±0.92	42.46±1.21	40.23±3.89	39.56±2.92	32.21±2.14	23.42±3.22
GFT	**73.89±2.76**	**74.26±4.39**	-	-	-	-	-	-
RiemannGFM	64.54±5.32	63.84±4.42	82.34±2.46	82.14±2.15	37.41±1.25	32.41±1.31	30.18±5.54	21.87±5.14
RGT(Ours)	70.90 ±5.20	70.73 ±5.52	**85.12±2.12**	**84.96±1.98**	**40.34±3.24**	**33.43±3.15**	**33.10±3.62**	**26.55±2.97**

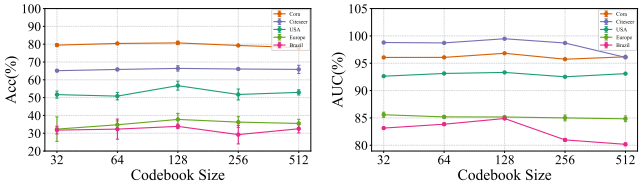
Importance of Geometric Vocabulary(RQ3). Graphs can be regarded as discrete manifolds. Therefore, modeling trees in hyperbolic space, cycles in spherical space and sequences in Euclidean space aligns well with the mathematical formulation of Riemannian geometry. To verify the effectiveness of the proposed geometric vocabulary, we assign trees, cycles and sequences to different manifolds: $\mathcal{H}_{-1}^{32}$ (hyperbolic), $\mathcal{S}_{1}^{32}$ (spherical) and $\mathcal{R}_{0}^{32}$ (Euclidean), resulting in nine possible combinations. The results across various datasets are reported in Table 5. The results show that assigning trees to hyperbolic space, cycles to spherical space and sequences to Euclidean space yields the best performance among all configurations. In contrast, combinations where all three vocabularies are embedded in mismatched spaces (Row 4 and Row 6 in the table) or one space (Row 7-9 in the table) lead to significantly worse results. These findings confirm the effectiveness of the geometry vocabulary in this work.

Table 5: Link prediction results of geometric ablation are reported in terms of AUC.

Trees	Cycles	Sequences	Cora	Citeseer	Pubmed	USA
$\mathcal{H}_{-1}^{32}$	$\mathcal{S}_{1}^{32}$	$\mathcal{R}_{0}^{32}$	**80.72±0.97**	**66.40±1.53**	**76.26±0.73**	**56.72±2.51**
$\mathcal{H}_{-1}^{32}$	$\mathcal{R}_{0}^{32}$	$\mathcal{S}_{1}^{32}$	78.53±0.78	64.98 ± 2.22	74.52±0.81	52.69±1.47
$\mathcal{S}_{1}^{32}$	$\mathcal{H}_{-1}^{32}$	$\mathcal{R}_{0}^{32}$	79.42±0.83	65.77±1.58	74.34±0.82	52.77±1.05
$\mathcal{S}_{1}^{32}$	$\mathcal{R}_{0}^{32}$	$\mathcal{H}_{-1}^{32}$	77.98±1.21	64.58±2.56	73.88±0.89	51.13±1.35
$\mathcal{R}_{0}^{32}$	$\mathcal{S}_{1}^{32}$	$\mathcal{H}_{-1}^{32}$	78.32±0.53	65.64±1.20	75.26±0.63	53.71±1.24
$\mathcal{R}_{0}^{32}$	$\mathcal{H}_{-1}^{32}$	$\mathcal{S}_{1}^{32}$	76.85±1.21	63.69±1.32	74.26±0.93	52.21±1.37
$\mathcal{H}_{-1}^{32}$	$\mathcal{H}_{-1}^{32}$	$\mathcal{H}_{-1}^{32}$	78.80±0.48	64.80±1.11	73.36±0.83	53.57±1.02
$\mathcal{S}_{1}^{32}$	$\mathcal{S}_{1}^{32}$	$\mathcal{S}_{1}^{32}$	77.12±0.77	63.41±1.09	74.38±0.45	53.79±1.34
$\mathcal{R}_{0}^{32}$	$\mathcal{R}_{0}^{32}$	$\mathcal{R}_{0}^{32}$	77.80±0.56	64.60±1.46	74.21±0.86	52.93±1.26

Impact of Hyperparameters. To analyze the sensitivity of model parameters, we focus on the impact of the codebook size on model performance. For each setting of the m, the model is pretrained from scratch and evaluated across multiple datasets. The test results for node classification and link prediction under various datasets are illustrated in Figure 2a and Figure 2b. The results indicate that a limited number of structural tokens in the codebook is sufficient to achieve competitive performance. However, when the codebook size becomes too large, performance degrades due to the increased risk of codebook collapse. This highlights the importance of selecting an appropriate codebook size for stable and effective structure-level quantization.

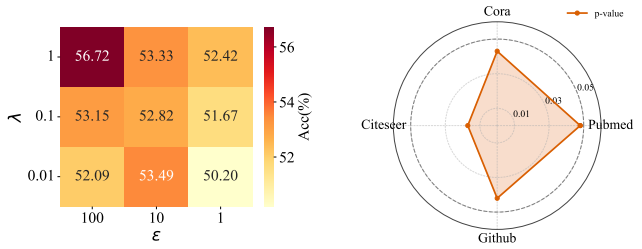
We also perform an hyperparameter study about λ, ϵ. The results of node classification on USA dataset are shown in the Figure 3a. Hence, the best $\lambda = 1, \epsilon = 100$ in our experiment.



(a) Node classification results. **(b) Link prediction results.**

Figure 2: Hyperparameter sensitivity analysis of codebook size.

Statistical Analysis. To mitigate the influence of randomness and ensure the statistical reliability of the results, we first systematically evaluate the performance of RGT compared to RiemannGFM across multiple datasets and random seeds. Five independent experiments are conducted on the Pubmed dataset, and Table 6 reports the mean and standard deviation of Acc and F1. Paired t-test results show that RGT significantly outperforms RiemannGFM at the 99% confidence level. Furthermore, Figure 3b presents the radar plot



(a) Heatmap about λ and ϵ. (b) P-value on four datasets.

Figure 3: Heatmap about λ and ϵ and radar plot of p-value.

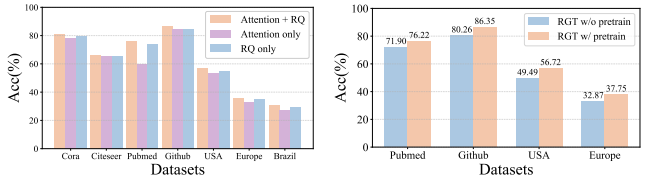
of p-value under the five-shot setting across four datasets. The results $p < 0.05$ demonstrate that RGT significantly outperforms RiemannGFM on certain datasets at the 95% confidence level.

Table 6: Cross-domain transfer learning performance of node classification on Pubmed (Bold: best).

Seed	RiemannGFM (Acc)	RGT (Acc)	RiemannGFM (F1)	RGT (F1)
Seed 0	73.94±5.66	**76.22±0.80**	74.25±4.25	**76.14±0.77**
Seed 1	71.52±5.44	**75.94±0.88**	72.11±4.01	**75.89±0.73**
Seed 2	72.03±5.70	**76.08±0.79**	73.09±4.37	**76.07±0.80**
Seed 3	74.11±5.61	**76.35±0.85**	73.64±4.40	**76.25±0.82**
Seed 4	69.77±5.58	**75.71±0.81**	72.87±4.16	**75.83±0.75**
Mean	72.67±5.60	**76.06±0.83**	73.19±4.24	**76.04±0.77**
p-value	–	**0.0056**	–	**0.0008**

Ablation Study. The core components of RGT are the cross-geometry attention and the Riemannian quantizer, which are responsible for encoding local coordinates and globally quantizing structural features, respectively. Therefore, it is essential to conduct ablation studies on these two components. The link prediction results across various datasets are presented in Figure 4a. The results show that applying Riemannian quantizer leads to better performance compared to the non-quantized variant. In addition, the cross-geometry attention mechanism also contributes positively to the results. Overall, Riemannian quantizer plays a more critical role in enhancing the model's transferability across domains.

To further demonstrate the effectiveness of cross-domain structural knowledge transfer, we compare the performance of RGT with and without pretraining. The node classification results are presented in Figure 4b, where the variant without pretraining is labeled as "w/o pretrain." The results clearly show that pretraining significantly enhances RGT's capability in transferring structural knowledge across domains.



(a) Acc of ablation models (b) Ablation on RGT Pretraining.

Figure 4: Ablation study on key components.

Impact of Pretraining Datasets(RQ4). This part investigates the impact of different training datasets on RGT. The are pretrained on three graph datasets (Flickr, AComp and WikiCS) respectively and the results of link prediction are reported in Table 7. The results indicate that the domain and number of datasets used for pretraining can influence model performance. RGT and RiemannGFM maintain stable performance under cross-domain settings. In contrast, while OpenGraph performs well when trained and tested on WikiCS (a citation network), its performance on the Citeseer drops significantly when trained on Flickr or AComp. Moreover, OpenGraph is unable to handle non-attributed graphs, which further limits its transfer ability across domains.

Table 7: Cross-domain link prediction performance on different pretraining datasets.

Pretraining	Method	Citeseer	Cora	USA	Brazil
Flickr	OpenGraph	73.17±0.23	77.46±0.23	53.16±0.98	61.16±0.19
	RiemannGFM	98.27±1.38	96.84±0.24	93.40±0.84	84.03±0.26
	RGT(Ours)	**99.11±0.01**	**97.81±0.32**	**93.41±0.02**	**84.92±0.50**
AComp	OpenGraph	72.95±0.28	78.47±0.25	79.79±0.81	68.31±0.14
	RiemannGFM	98.50±0.01	96.59±0.00	92.85±0.13	84.32±0.13
	RGT(Ours)	**98.87±0.03**	**97.91±0.17**	**92.90±0.02**	**84.89±0.11**
WikiCS	OpenGraph	87.01±1.26	78.07±0.12	80.72±1.45	63.57±0.74
	RiemannGFM	98.44±0.81	96.62±0.02	92.63±0.05	85.10±0.08
	RGT(Ours)	**98.81±0.02**	**97.56±0.08**	**92.63±0.05**	**85.91±0.54**

Scale to Web-scale Graph(RQ5). We evaluate the performance of the pretrained RGT model on the ogbn-products dataset. The cross-domain node classification results are illustrated in Figure 5. During inference, the primary computational overhead originates from eigenvalue computation and subgraph sampling. Nevertheless, RGT remains capable of efficiently handling datasets of this scale. These results demonstrate that RGT exhibits strong scalability and competitive performance on web-scale graph datasets.

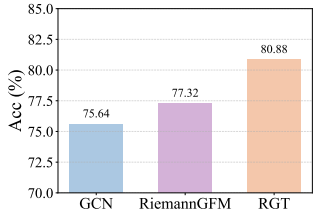


Figure 5: Cross-domain Node Classification Performance on Web-Scale Graphs.

5 Conclusion

This work primarily contributes a novel Riemannian graph tokenizer, designed to quantise and transfer structural knowledge across graph domains. Specifically, we first encode the coordinates of geometric vocabulary. Subsequently, the proposed Riemannian quantizer facilitates the learning of structural knowledge tranfer. During the quantization stage, we propose a Riemannian Straight-Through Estimator (RSTE) to maintain the intrinsic structure. The proposed model is evaluated on both attributed and non-attributed graph datasets. The empirical results demonstrate the superiority of RGT. Future work may explore jointly encoding structural semantics and textual features.

6 Acknowledgments

This work is supported in part by the National Natural Science Foundation of China(No.62550138,62192784,62572064,62472329,62202164).

References

[1] Ethem Alpaydin. 2021. *Machine learning*. MIT press.

[2] Gregor Bachmann, Gary Bécigneul, and Octavian Ganea. 2020. Constant curvature graph convolutional networks. In *ICLR*. PMLR, 486–496.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[4] Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720* (2020).

[5] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology* 15, 3 (2024), 1–45.

[6] Ching-Yao Chuang and Stefanie Jegelka. 2022. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. *NeurIPS* 35 (2022), 2944–2957.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[8] Christopher Fifty, Ronald G Junkins, Dennis Duan, Aniketh Iyengar, Jerry W Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. 2024. Restructuring vector quantization with the rotation trick. *arXiv preprint arXiv:2410.06424* (2024).

[9] Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. 2021. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics* 22, 6 (2021), bbab159.

[10] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2018. Learning mixed-curvature representations in product spaces. In *International conference on learning representations*.

[11] Zihao Guo, Qingyun Sun, Haonan Yuan, Xingcheng Fu, Min Zhou, Yisen Gao, and Jianxin Li. 2025. GraphMoRE: Mitigating Topological Heterogeneity via Mixture of Riemannian Experts. In *AAAI*, Vol. 39. 11754–11762.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS* 30 (2017).

[13] Neil He, Jiahong Liu, Buze Zhang, Ngoc Bui, Ali Maatouk, Menglin Yang, Irwin King, Melanie Weber, and Rex Ying. 2025. Position: Beyond Euclidean–Foundation Models Should Embrace Non-Euclidean Geometries. *arXiv preprint arXiv:2504.08896* (2025).

[14] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. 2023. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*. 737–746.

[15] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in the 33rd NeurIPS*, Vol. 33. 22118–22133.

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

[17] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. 2023. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *ICML*.

[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[19] Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226* (2018).

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

[21] Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024. ZeroG: Investigating Cross-dataset Zero-shot Transferability in Graphs. In *KDD*. ACM, 1725–1735.

[22] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024. One For All: Towards Training One Graph Model For All Classification Tasks. In *ICLR*. OpenReview.net.

[23] Jiawei Liu, Zhiyan Liu, Xun He, Jianwang Zhai, Zhengyuan Shi, Qiang Xu, Bei Yu, and Chuan Shi. 2025. WideGate: Beyond Directed Acyclic Graph Learning in Subcircuit Boundary Prediction. (2025).

[24] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. 2025. Graph Foundation Models: Concepts, Opportunities and Challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[25] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. 2020. Differentiating through the fréchet mean. In *ICLR*. PMLR, 6393–6403.

[26] Yuanfu Lu, Ruobing Xie, Chuan Shi, Yuan Fang, Wei Wang, Xu Zhang, and Leyu Lin. 2021. Social influence attentive neural network for friend-enhanced recommendation. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part IV*. Springer, 3–18.

[27] Yihong Ma, Ning Yan, Jiayu Li, Masood Mortazavi, and Nitesh V Chawla. 2024. Hetgpt: Harnessing the power of prompt tuning in pre-trained heterogeneous graph neural networks. In *Proceedings of the ACM Web Conference 2024*. 1015–1023.

[28] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. 2024. Position: Graph Foundation Models Are Already Here. In *ICML*. OpenReview.net.

[29] OpenAI. 2024. GPT-4 Technical Report.

[30] Peter Petersen. 2006. *Riemannian geometry*. Vol. 171. Springer.

[31] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. *struc2vec*: Learning Node Representations from Structural Identity. In *KDD*. ACM, 385–394.

[32] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-Scale attributed node embedding. *J. Complex Networks* 9, 2 (2021).

[33] Rik Sarkar. 2011. Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane. In *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 7034)*. Springer, 355–366.

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

[35] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *CoRR* (2018).

[36] Li Sun, Zhenhao Huang, Hao Peng, Yujie Wang, Chunyang Liu, and Philip S Yu. 2024. Lsenet: Lorentz structural entropy neural network for deep graph clustering. *arXiv preprint arXiv:2405.11801* (2024).

[37] Li Sun, Zhenhao Huang, Qiqi Wan, Hao Peng, and Philip S Yu. 2024. Spiking graph neural network on riemannian manifolds. *Advances in Neural Information Processing Systems* 37 (2024), 34025–34055.

[38] Li Sun, Zhenhao Huang, Hua Wu, Junda Ye, Hao Peng, Zhengtao Yu, and Philip S Yu. 2023. Deepricci: Self-supervised graph structure-feature co-refinement for alleviating over-squashing. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 558–567.

[39] Li Sun, Zhenhao Huang, Ming Zhang, and Philip S Yu. 2025. Deeper with Riemannian Geometry: Overcoming Oversmoothing and Oversquashing for Graph Foundation Models. *arXiv preprint arXiv:2510.17457* (2025).

[40] Li Sun, Zhenhao Huang, Suyang Zhou, Qiqi Wan, Hao Peng, and Philip S. Yu. 2025. RiemannGFM: Learning a Graph Foundation Model from Riemannian Geometry. In *WWW*. ACM, 1154–1165.

[41] Li Sun, Zhongbao Zhang, Junda Ye, Hao Peng, Jiawei Zhang, Sen Su, and Philip S Yu. 2022. A self-supervised mixed-curvature graph neural network. In *AAAI*, Vol. 36. 4146–4155.

[42] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2024. All in One: Multi-task Prompting for Graph Neural Networks (Extended Abstract). In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*. ijcai.org, 8460–8465.

[43] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. 2021. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522* (2021).

[44] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *NeurIPS* 30 (2017).

[45] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*. OpenReview.net.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903

[47] Limei Wang, Kaveh Hassani, Si Zhang, Dongqi Fu, Baichuan Yuan, Weilin Cong, Zhigang Hua, Hao Wu, Ning Yao, and Bo Long. 2025. Learning Graph Quantized Tokenizers. In *ICLR*.

[48] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. 2023. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *NeurIPS* 36 (2023), 61501–61513.

[49] Zehong Wang, Zheyuan Zhang, Nitesh V. Chawla, Chuxu Zhang, and Yanfang Ye. 2024. GFT: Graph Foundation Model with Transferable Tree Vocabulary. In *NeurIPS*.

[50] Bin Wu, Yihang Wang, Yuanhao Zeng, Jiawei Liu, Jiashu Zhao, Cheng Yang, Yawen Li, Long Xia, Dawei Yin, and Chuan Shi. 2025. Graph Foundation Models for Recommendation: A Comprehensive Survey. arXiv:2502.08346

[51] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[52] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. 2021. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence* 2, 2 (2021), 109–127.

[53] Lianghao Xia, Ben Kao, and Chao Huang. 2024. OpenGraph: Towards Open Graph Foundation Models. In *EMNLP*. Association for Computational Linguistics, 2365–2379.

[54] Bo Yan, Yang Cao, Haoyu Wang, Wenchuan Yang, Junping Du, and Chuan Shi. 2024. Federated heterogeneous graph neural network for privacy-preserving recommendation. In *Proceedings of the ACM Web Conference 2024*. 3919–3929.
[55] Ling Yang, Ye Tian, Minkai Xu, Zhongyi Liu, Shenda Hong, Wei Qu, Wentao Zhang, Bin Cui, Muhan Zhang, and Jure Leskovec. 2024. VQGraph: Rethinking Graph Representation Space for Bridging GNNs and MLPs. In *ICLR*. OpenReview.net.
[56] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *TASLP* (2021).
[57] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021. Lorentzian Graph Convolutional Networks. In *WWW*. ACM / IW3C2, 1249–1261.
[58] Zhongjian Zhang, Xiao Wang, Huichi Zhou, Yue Yu, Mengmei Zhang, Cheng Yang, and Chuan Shi. 2025. Can large language models improve the adversarial robustness of graph neural networks?. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 2008–2019.
[59] Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. 2024. All in One and One for All: A Simple yet Effective Method towards Cross-domain Graph Pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*. ACM, 4443–4454.
[60] Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael M Bronstein, and Jian Tang. 2025. Fully-inductive Node Classification on Arbitrary Graphs. In *ICLR*.
[61] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.
[62] Yongxin Zhu, Bocheng Li, Yifei Xin, and Linli Xu. 2024. Addressing representation collapse in vector quantized models with one linear layer. *arXiv preprint arXiv:2411.02038* (2024).

A Theoretical Analysis

A.1 The Completeness of Geometric Vocabulary

Definition 1 formalizes a complete geometric vocabulary that supports the structural patterns of arbitrary graph.

From the perspective of graph theory, any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be fully described as a combination of tree and cycle structures. A spanning tree (composed by stitching together multiple subtrees) provides the connected backbone of the graph, while the non-tree edges form cycles. Therefore, the combination of **trees** and **cycles** is sufficient to reconstruct the graph topology in terms of connectivity.

However, from the geometric modeling and topological structure perspective, it is necessary to additionally introduce the **sequence structure** as a linear substructure. A sequence represents a segment of the graph with zero curvature, i.e., a locally flat subgraph that serves as a first-order Euclidean approximation. The geometric vocabulary possesses the following geometric interpretations:

- Tree structures are mapped to hyperbolic space ($\kappa < 0$), capturing hierarchical and branching patterns,
- Cycle structures are mapped to spherical space ($\kappa > 0$), encoding closed-loop topologies,
- Sequence structures are mapped to Euclidean space ($\kappa = 0$), modeling locally linear or path-like substructures.

Therefore, the trees, cycles and sequences constitute a complete geometric vocabulary for representing graph structures. Each structural type is naturally associated with a constant curvature space—hyperbolic (negative), spherical (positive), or Euclidean (zero). They provide a unified framework for modeling arbitrary graphs across diverse structural patterns, topologies and geometric characteristics. Figure 6 illustrates the cross-domain transferability of RGT based on geometric vocabulary.

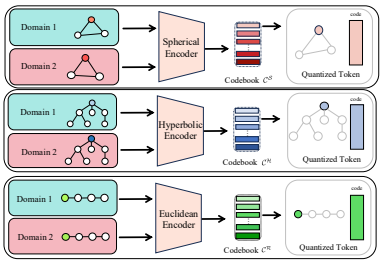


Figure 6: Cross-Domain Transfer of Geometric Vocabulary.

A.2 Proof of The Theorem 1

Let $c_i^{\mathcal{M}} = RQ(z_i^{\mathcal{M}}), c_j^{\mathcal{M}} = RQ(z_j^{\mathcal{M}})$. By the triangle inequality, we have $d(c_i^{\mathcal{M}}, c_j^{\mathcal{M}}) \le d(c_i^{\mathcal{M}}, z_i^{\mathcal{M}}) + d(z_i^{\mathcal{M}}, z_j^{\mathcal{M}}) + d(z_j^{\mathcal{M}}, c_j^{\mathcal{M}})$. Since $c_i^{\mathcal{M}}, c_j^{\mathcal{M}}$ are the nearest codewords of $z_i^{\mathcal{M}}, z_j^{\mathcal{M}}$ and $d(z_i^{\mathcal{M}}, z_j^{\mathcal{M}}) \le \epsilon$. Then we obtain $d(c_i^{\mathcal{M}}, z_i^{\mathcal{M}}) \le \epsilon, \quad d(c_j^{\mathcal{M}}, z_j^{\mathcal{M}}) \le \epsilon$. Hence, $d(c_i^{\mathcal{M}}, c_j^{\mathcal{M}}) \le 3\epsilon$, the theorem is proved.

A.3 Proof of The Theorem 2

Let $(\mathcal{M}, g)$ be a smooth Riemannian manifold equipped with a geodesic distance function $d_{\mathcal{M}}(\cdot,\cdot)$. Consider a set of geometric vocabulary $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$ and a codebook $C = \{\boldsymbol{c}_j^{\mathcal{M}}\}_{j=1}^{m} \subset \mathcal{M}^d$. The quantizer $RQ : \{\mathcal{S}_i^{(j)}\}_{j=1}^{M} \to C$ assigns each geometry vocabulary $\boldsymbol{z}$ to a codeword according to:

$$\boldsymbol{c}_j^{\mathcal{M}} = \arg\min_{\boldsymbol{c} \in \mathcal{M}^d} \mathbb{E}\left[d_{\mathcal{M}}(\boldsymbol{z}, \boldsymbol{c}) \mid \boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M} \right], \tag{17}$$

Then, the quantizer RQ minimizes the expected geodesic distortion over the entire space:

$$\mathbb{E}\left[d_{\mathcal{M}}(\boldsymbol{z}, RQ(\boldsymbol{z})) \right]. \tag{18}$$

Let $P_j = \mathbb{P}(\boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M})$ denote the probability that a geometry vocabulary falls into region $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$. Suppose RQ' is another quantizer defined on the same set $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$, but with a different codebook $C' = \{\boldsymbol{c}'_j\}_{j=1}^{m}$. The expected distortion of RQ' is given by:

$$\begin{aligned}
\mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, RQ'(\boldsymbol{z}))] &= \sum_{j=1}^{m} P_j \cdot \mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, RQ'(\boldsymbol{z})) \mid \boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M}] \\
&= \sum_{j=1}^{m} P_j \cdot \mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, \boldsymbol{c}'_j) \mid \boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M}] \\
&\ge \sum_{j=1}^{m} P_j \cdot \min_{\boldsymbol{c} \in C} \mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, \boldsymbol{c}) \mid \boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M}] \\
&= \sum_{j=1}^{m} P_j \cdot \mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, \boldsymbol{c}_j^{\mathcal{M}}) \mid \boldsymbol{z} \in \{\mathcal{S}_i^{(j)}\}_{j=1}^{M}] \\
&= \mathbb{E}[d_{\mathcal{M}}(\boldsymbol{z}, RQ(\boldsymbol{z}))].
\end{aligned} \tag{19}$$

Conclusion. Therefore, choosing each codeword $\boldsymbol{c}_j^{\mathcal{M}}$ as the global centroid of $\{\mathcal{S}_i^{(j)}\}_{j=1}^{M}$ across multiple domains minimizes the total expected geodesic distortion. This confirms that the quantizer RQ defined in Eq.(6) is the optimal structure-level Riemannian quantization strategy.

B Riemannian Straight-Through Estimation and Codebook Collapse

B.1 Riemannian Straight-Through Estimator

Let $z_i^{\mathcal{M}}$ denote the encoder output in manifold $\mathcal{M}$ and $\boldsymbol{q}_i^{\mathcal{M}}$ denote the corresponding nearest code. Since the nearest-neighbor quantization operation is non-differentiable, we adopt the following Riemannian Straight-Through Estimation (RSTE) formulation:

$$\hat{\boldsymbol{z}}_i{}^{\mathcal{M}} = \exp_{\boldsymbol{z}_i^{\mathcal{M}}}\left(\mathrm{sg}\left[\log_{\boldsymbol{z}_i^{\mathcal{M}}}(\boldsymbol{q}_i^{\mathcal{M}})\right]\right), \quad \boldsymbol{q}_i^{\mathcal{M}} \in \mathcal{M}, \tag{20}$$

where $\log_{\boldsymbol{z}_i^{\mathcal{M}}}(\boldsymbol{q}_i^{\mathcal{M}})$ maps the code $\boldsymbol{q}_i^{\mathcal{M}}$ onto the tangent space at $\boldsymbol{z}_i^{\mathcal{M}}$. $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, which ensures that gradients are only propagated through $\boldsymbol{z}_i^{\mathcal{M}}$ and not through $\boldsymbol{q}_i^{\mathcal{M}}$.

B.2 RSTE Alleviates Codebook Collapse

Codebook collapse is a major challenge in vector quantization (VQ) methods, and numerous studies have sought to address this issue (e.g., SimVQ, RVQ and other related works). Codebook collapse commonly occurs in the following scenarios. When the encoded vector $\boldsymbol{z}$ is significantly distant from its assigned discrete code $\boldsymbol{q}$, applying the Euclidean interpolation $\boldsymbol{z} + \mathrm{sg}[\boldsymbol{q} - \boldsymbol{z}]$ tends to pull multiple $\boldsymbol{z}$ vectors into a shared local Euclidean neighborhood. During training, gradients are propagated purely in Euclidean space, causing the optimization to overfit to a small subset of codebook entries, while leaving other codes unused or inactive. In contrast, RVQ leverages the Riemannian structure by computing the local geometric offset of the code q relative to $\boldsymbol{z}$ via the logarithmic map $\log_{\boldsymbol{z}}(\boldsymbol{q})$. During the exponential map $\exp_{\boldsymbol{z}}(\cdot)$ that projects back to the manifold, each codeword maintains its manifold-aware structure. Gradients are propagated within the tangent space $\mathcal{T}_{\boldsymbol{z}}\mathcal{M}$, which is consistent with the underlying geometry of the space. Alternatively, other techniques such as expiring stale codes [56] or affine re-parameterization [17] can also be employed to mitigate this problem. As a result, different structural samples are encouraged to select codes centered around different geometric regions of the manifold. This promotes diversified codeword utilization and effectively mitigates the codebook collapse issue.

C Algorithm

The overall training process of RGT is presented in Algorithm 1.

Algorithm 1: Training Algorithm of RGT

Input: multi-domain graphs, Hyperparameters of RGT.
Output: Model parameters of RGT

```
Initialize node features and node coordinates on CCSs;
  Sample substructures according to the geometry vocabulary;
while model not converged do
    for each substructure in each geometry do
        Conduct the cross-geometry attention in Eq.(2) and update node coordinates by Eq.(3)
        Conduct the Riemannian Quantizer in Eq.(6) to tokenise graph structures
        Conduct the geometry-aligned decoder in Eq.(10)
    Compute the loss with Eq.(11)
    Update model parameters via gradient descent via Eq.(15) and Eq.(16).
```

D Experiment Details

D.1 Experimental Setup

During the experimental process, we implement each layer of the model as described in the section 3. Two aspects of the setup require special attention. **Hyperparameter settings:** The number of layers for tree-based sampling is set to 2, cycles are constructed as triangles (3-node), and the sequence length is set to L. The embedding dimension d for structural embedding is 32. The curvatures of the constant curvature spaces κ are set to 1 (spherical), -1 (hyperbolic), and 0 (Euclidean space). The feature dimension of each code is 32, the number of code in codebook $m = 128$. All hyperparameters are selected via grid search. For baselines, the hyperparameters strictly follow those reported in the original papers. **Pretraining and Evaluation:** The pretraining datasets used for our model is the same as in RiemannGFM. Other baselines use their original pretraining datasets. During evaluation, the pretrained codebooks are kept frozen while the GNN classifier is fine-tuned on the target test datasets. Other models are evaluated on the same test datesets. The results are averaged over multiple runs to reduce the impact of randomness.

Pretraining of RGT. We pretrain RGT using the Adam optimizer with a learning rate of 0.001, a batch size of 32. The pretraining stage runs for 3 epochs and 3 iterations. For RQ, the codebook size to 128, the code dimension to 32. Dropout is set to 0.1.

Fine-tuning of RGT. We fine-tune RGT for node classification and link prediction. For node classification, we use a learning rate of 0.01 and train for 120 epochs. The task hidden dimension is set to 128. For link prediction, we train for 120 epochs with a hidden dimension of 256. Drop edge and drop feature rates are set to 0.2 and 0.3, respectively.

Table 8: Hyperparameters for RGT in pretraining and fine-tuning

Hyperparameter	Pretraining	Fine-tuning
Learning Rate	0.001	0.01
Epochs	3	120
Batch Size	32	64
Dropout	0.1	0.1
Drop Edge Rate	–	0.2
Drop Feature Rate	–	0.3
Embedding Dimension	32	32
Hidden Dimension	256	256
Task Hidden Dimension	–	128
Codebook Size	128	–
Code Dimension	32	–
Patience	3	10

Comparison with GFT and VQGraph. RGT, GFT [49] and VQGraph [55] all leverage vector quantization techniques to extract discrete structural semantics from graphs. However, they differ significantly in terms of core design principles and application scenarios. The key distinctions between RGT and the other two models are summarized as follows:

Table 9: Cross-domain transfer learning performance of node classification on three datasets.

Method	Node Classification Results					
	Cora		Citeseer		Brazil	
	ACC	F1	ACC	F1	ACC	F1
DGI	80.62±1.63	80.67±1.97	69.67±0.42	62.92±1.81	31.78±1.62	29.15±4.16
GraphMAE	79.77±0.34	79.82±0.23	50.20±7.07	44.97±8.20	32.14±5.83	21.14±7.72
OFA	56.92±3.09	54.21±2.02	49.68±1.36	50.22±2.13	-	-
ZeroG	59.20±1.07	55.32±1.34	52.34±0.89	55.67±1.22	-	-
Graphany	79.98±1.57	77.56±1.13	63.73±1.35	62.94±1.32	29.24±1.24	20.24±2.48
OpenGraph	75.03±0.97	74.37±0.80	62.34±1.06	60.29±1.34	20.42±2.33	15.63±1.31
GFT	73.38±1.03	73.83±1.08	-	-	-	-
RiemannGFM	78.66±0.47	79.01±0.34	64.78±2.01	63.95±1.69	30.77±4.21	22.52±4.60
RGT(Ours)	**80.72±0.97**	**80.79±0.92**	**66.40±1.53**	**67.81±0.94**	**33.85±1.26**	**29.27±0.00**

Table 10: Cross-domain transfer learning performance of link prediction on three datasets.

Method	Link Prediction Results					
	Cora		Citeseer		Brazil	
	AUC	AP	AUC	AP	AUC	AP
DGI	66.06±1.66	65.64±1.91	82.81±1.12	84.72±1.15	64.09±5.91	59.09±7.17
GraphMAE	54.86±0.20	57.87±1.58	79.18±1.02	75.51±0.94	74.00±2.81	71.86±4.04
OFA	70.84±2.01	71.34±1.89	75.61±2.34	76.13±1.26	-	-
OpenGraph	74.03±0.97	73.21±0.84	74.45±0.46	76.32±0.76	65.31±1.54	64.39±1.65
RiemannGFM	96.75±0.14	96.07±0.15	98.85±0.21	99.04±0.08	84.18±0.20	82.38±0.11
RGT(Ours)	**96.82±0.01**	**96.26±0.01**	**99.48±0.00**	**99.39±0.01**	**84.89±0.11**	**83.61±0.10**

Model objective: GFT focuses on building a general-purpose reasoning model for diverse graph tasks, emphasizing adaptability across domains and tasks. VQGraph, on the other hand, is designed as a lightweight structure-aware MLP model for efficient node classification. In contrast, RGT aims to facilitate structural knowledge transfer across domains, with an emphasis on geometric consistency and representation generalization.

Pretraining strategy: GFT performs large-scale pretraining across various graph tasks and text-attributed graphs to capture domain-invariant patterns. VQGraph trains on a single dataset to extract local structural information. RGT adopts a cross-domain pretraining strategy to encode structural semantics and geometric mappings over multi-domain graphs.

Token representation: GFT directly uses discrete tokens as transferable patterns and inputs them into a downstream classifier. VQGraph treats tokens as auxiliary structural knowledge to enhance MLP training. In contrast, RGT encodes structural tokens in multiple geometric spaces and performs unified fusion in a shared tangent space, enabling consistent geometric alignment and cross-space adaptability.

Downstream applicability: GFT is applicable to a broad range of tasks such as node classification and few-shot learning. VQGraph is primarily designed for conventional node classification with a basic pretraining-finetuning pipeline. RGT supports various graph tasks including node classification and link prediction, and demonstrates strong robustness and transferability, particularly on non-attributed graphs and cross-domain settings.

D.2 Additional Results

To further assess the generalization capability of the proposed RGT model, we conduct supplementary cross-domain node classification experiments on three additional datasets. As shown in Table 9, RGT consistently achieves performance that is competitive with or superior to that of existing baselines across all datasets. These results align with the main experimental findings presented in the paper, thereby reinforcing the robustness and adaptability of RGT across diverse domains and graph structures. Collectively, the supplementary experiments provide additional empirical support for the effectiveness of RGT in cross-domain transfer learning tasks.

We further evaluate the generalization capability of RGT through link prediction tasks on the same datasets. As shown in Table 10, RGT maintains consistently strong performance across domains, further reinforcing its robustness and adaptability in diverse graph learning scenarios.

Table 11: One-shot learning performance of node classification on three datasets.

Method	Node Classification Results					
	Cora		Citeseer		Brazil	
	ACC	F1	ACC	F1	ACC	F1
DGI	39.21±1.70	34.33±2.09	34.13±3.08	31.41±1.11	29.92±3.24	21.75±2.54
GraphMAE	51.86±8.81	50.90±8.49	41.24±8.71	36.07±9.75	26.56±3.41	17.53±3.34
OFA	57.35±3.09	55.92±2.85	36.89±3.98	38.48±5.76	-	-
ZeroG	63.59±1.75	63.24±1.22	**60.34±1.21**	**58.24±1.67**	-	-
Graphany	69.39±3.29	68.99±1.11	33.92±0.03	34.04±0.02	24.34±0.72	19.34±3.68
OpenGraph	**74.63±1.30**	**74.12±1.21**	58.65±1.23	56.24±1.94	20.34±2.31	21.71±1.96
GFT	42.66±6.08	35.81±6.89	-	-	-	-
RiemannGFM	57.54±4.24	56.95±5.24	38.02±9.45	32.42±9.87	25.18±2.24	21.97±5.65
RGT(Ours)	46.04±3.91	43.27±6.40	39.40±5.25	34.64±6.33	**30.77±4.87**	**22.72±2.81**

Table 12: Five-shot learning performance of node classification on three datasets.

Method	Node Classification Results					
	Cora		Citeseer		Brazil	
	ACC	F1	ACC	F1	ACC	F1
DGI	64.54±4.19	65.27±4.90	48.03±3.56	44.73±3.02	30.62±3.24	29.29±2.76
GraphMAE	72.85±1.61	71.80±1.74	47.74±4.99	40.88±6.81	28.57±6.47	20.81±5.85
OFA	65.23±3.19	62.72±2.35	43.25±3.78	40.21±4.78	-	-
ZeroG	63.59±1.75	63.20±1.52	55.92±5.21	52.74±4.16	-	-
Graphany	68.50±3.29	67.24±1.96	34.67±7.48	35.02±5.51	28.66±7.65	22.31±3.27
OpenGraph	**74.42±1.38**	**73.63±1.08**	50.34±3.34	51.21±2.78	28.30±3.14	24.39±3.12
GFT	43.77±7.49	39.35±6.69	-	-	-	-
RiemannGFM	61.15±4.34	60.55±5.20	56.35±5.24	55.74±4.62	29.57±4.68	22.48±5.16
RGT(Ours)	65.54±3.38	65.19±3.87	**58.30±4.49**	**57.44±4.95**	**33.60 ±2.46**	**30.47±6.05**

Table 13: Cross-domain node classification performance on different pre-training datasets.

Pre-training	Method	Testing Datasets			
		Citeseer	Cora	USA	Brazil
Flickr	OpenGraph	58.86±0.89	59.43±1.60	25.97±4.98	17.04±4.12
	Riemannian	65.74±2.07	77.62±0.78	51.34±1.61	23.85±9.29
	RGT(Ours)	**66.82±0.75**	**80.34±0.70**	**54.20±1.36**	**28.46±4.62**
AComp	OpenGraph	46.53±1.6	50.68±1.29	25.88±3.84	15.56±3.99
	Riemannian	66.4±1.76	80.02±0.87	48.66±1.49	30.38±3.28
	RGT(Ours)	**67.28±0.99**	**80.32±0.67**	**54.79±1.85**	**30.77±0.00**
WikiCS	OpenGraph	53.01±1.02	68.38±0.54	26.89±5.22	15.56±4.63
	Riemannian	65.68±1.87	78.42±1.05	51.93±1.08	**28.85±1.14**
	RGT(Ours)	**66.40±0.52**	**80.10±0.81**	**53.45±0.90**	28.46±4.62

To further examine the generalization of RGT under limited supervision, we conduct additional one-shot node classification experiments on the same set of datasets. As reported in Table 11, while performance varies across methods and datasets, RGT remains competitive in low-label regimes, demonstrating its potential for transfer learning even in extremely data-scarce settings.

To complete our analysis under low-resource conditions, we include 5-shot node classification results on three additional datasets in Table 12. RGT shows stable and competitive performance, particularly excelling on Citeseer and Brazil, where it surpasses all baselines. These results further confirm RGT's capability to generalize under limited supervision, aligning well with our main experimental findings.

To further substantiate the main findings, we conduct additional analysis on the cross-domain node classification performance of RGT under different pre-training datasets. As illustrated in Table 13, RGT consistently achieves superior or competitive results across all target datasets, irrespective of the source domain. These results highlight the robustness of RGT's pre-training framework and its ability to transfer across diverse domains. While baseline methods may perform adequately under certain configurations, RGT demonstrates more stable and transferable representations overall. This additional evidence further confirms the effectiveness of RGT in cross-domain scenarios, complementing the core experiments presented in the main text.