

Technical Appendices and Supplementary Material

A Notation

Table 3 summarizes all notations mainly used in this work and their corresponding explanations.

Table 3: Main Notations

Notations	Descriptions
$\mathcal{Q}_\beta^{p,q}$	A pseudo-hyperboloid with $q + 1$ time dimensions, p space dimensions, and space curvature β
$\hat{\mathcal{Q}}_\beta^{p,q}$	The zero-first submanifold of $\mathcal{Q}_\beta^{p,q}$
$\langle \cdot, \cdot \rangle$	The inner product in Euclidean space
$\langle \cdot, \cdot \rangle_q$	The inner product in pseudo-Euclidean space
$\ \cdot\ $	The vector norm in Euclidean space
$\ \cdot\ _q$	The vector norm in pseudo-Euclidean space
$G = (V_G, E_G)$	A static graph G with the node set V_G and the edge set E_G
x, y, z, u, v, u', v'	A single vector or point
X	A set of vectors x_i
N	The number of vectors in a vector set X
$\mathcal{T}_x \mathcal{M}$	The tangent space at point x in a manifold \mathcal{M}
\exp, \log	The exponential and logarithmic maps
$\mathcal{P}_{x \rightarrow y}^\gamma$	The parallel transport from $\mathcal{T}_x \mathcal{Q}_\beta^{p,q}$ to $\mathcal{T}_y \mathcal{Q}_\beta^{p,q}$ along the geodesic γ
$D_\gamma(x, y), d_\gamma(x, y)$	The distance between two points x, y
d	The embedding dimension
\mathbb{R}	The Euclidean space
\mathbb{N}	The natural number set
\mathbb{S}	The spherical space
\mathbb{L}	The hyperbolic space or the Lorentz model
Φ	The extrinsic mapping
Ψ	The diffeomorphic function
$\mathcal{N}(\mu, \sigma^2)$	The Gaussian distribution with the mean μ and the variance σ^2
f	The probability density function of the Gaussian distribution
Γ	The bijective function to Gaussian distributions

B Related works

Graph embedding in Euclidean space. Euclidean space has long been the foundation of deep learning because it supports fundamental linear algebraic operations such as vector addition and matrix multiplication. As a result, many early graph representation learning methods were developed within Euclidean geometry and achieved notable successes. One of the pioneering approaches is the Graph Convolutional Network (GCN) introduced by Kipf and Welling (2016), which enabled deep learning-based graph embeddings. To address GCN’s limitations in depth and its restricted capacity to capture long-range dependencies, Transformer architecture have been adapted (Vaswani et al., 2017) for graph data. For example, GraphGPS (Rampášek et al., 2022) integrates local message passing with global attention in a hybrid architecture. Despite their superior performance over graph neural networks in small-scale graph applications, Transformer-based methods often struggle with scalability. (Wu et al., 2022, 2023). Standard attention computes pairwise interactions between all node pairs through a softmax function, resulting in quadratic computational complexity. An alternative approach is utilizing linear attention (Katharopoulos et al., 2020), which offers linear computational complexity, for large-scale graph learning. For instance, NodeFormer (Wu et al., 2022) introduces a novel kernelized Gumbel-Softmax message-passing mechanism to approximate the standard attention with linear complexity. SGFormer (Wu et al., 2023) demonstrates the surprising effectiveness of a *single-layer, single-head* linear attention module in node-level tasks. Leveraging

the advantages of linear attention, we propose a pseudo-Riemannian graph model \mathcal{Q} -GT built upon the linear Transformer architecture to ensure scalability and efficiency.

Graph embedding in non-Euclidean space. As data becomes increasingly complex and diverse, the limitations of Euclidean geometry in graph representation learning are becoming more evident. For instance, [Krioukov et al. \(2010\)](#) proved that hyperbolic geometry is better suited for embedding scale-free and hierarchical graphs due to its exponential expansion property. Consequently, [Chami et al. \(2019\)](#) extended the classic GCN to hyperbolic GCN by shifting the graph operation to the tangent space for executing vector operations. [Yang et al. \(2024\)](#) developed a scalable and robust hyperbolic Transformer called Hypformer, which employs the linear attention mechanism fully on the hyperbolic manifold. For modeling graphs of mixed topologies, κ -GCN ([Bachmann et al. 2019](#)) generalized GCN to the Cartesian product of constant curvature manifolds. However, computing explicit distances in the product of manifolds can introduce bias toward one component over the other, thereby restricting the model’s representational capacity ([Law, 2021](#)). Alternatively, pseudo-Riemannian geometry, which generalizes spherical and hyperbolic geometries, can address the weakness of product spaces with its intrinsic distance ([Law, 2021](#); [Xiong et al., 2021](#)). [Law and Stam \(2020\)](#) proposed the first nonparametric method to learn graph representations with a novel optimization tool on pseudo-Riemannian manifolds. Then [Law \(2021\)](#) designed a graph neural network on the quotient manifold, which is a submanifold of the pseudo-Riemannian manifold to avoid geodesically disconnection. On the other hand, \mathcal{Q} -GCN ([Xiong et al., 2021](#)) introduced a diffeomorphism that decomposes pseudo-Riemannian manifolds into the product of spherical and Euclidean spaces, enabling a lightweight extension of GCNs to pseudo-Riemannian geometry. Nevertheless, existing methods suffer from three notable limitations: (1) the diffeomorphism to the product of spherical and Euclidean spaces discards hierarchical information inherent in the graph structure; (2) the GNN architecture is prone to the oversquashing problem as the number of layers increases; and (3) previous methods rely on a brute-force algorithm to select the appropriate pseudo-hyperboloid, which is computationally expensive. To address these limitations and attain better outcomes, we propose \mathcal{Q} -GT, which is the first Transformer-based graph method in pseudo-Riemannian geometry.

C More details about space searching algorithm

C.1 Gaussian sectional curvature

Sectional curvature is a fundamental concept in Riemannian geometry that describes how a manifold curves in different two-dimensional directions ([Lee, 2019](#)). Given a point x on a manifold \mathcal{M} and two linear independent tangent vectors $u, v \in \mathcal{T}_x\mathcal{M}$ spanning a two-dimensional subspace U , the sectional curvature measures the Gaussian curvature of the surface $\text{Exp}(U) \subseteq \mathcal{M}$ formed by exponential mapping tangent vectors within U . Intuitively, it provides insight into the local geometric properties around a point x ([Gu et al., 2018](#)), where positive sectional curvature indicates a locally spherical structure, zero curvature corresponds to a Euclidean-like space, and negative curvature signifies a hyperbolic structure. In the context of graphs, [Gu et al. \(2018\)](#) proposed a discrete curvature analog κ to estimate the section curvature in an undirected graph G . Given a node $m \in G$, for any two neighbor nodes b, c and a reference node $a \neq m$, the curvature analog $\kappa(m; b, c)$ is defined as:

$$\kappa(m; b, c) = \frac{1}{|V_G| - 1} \sum_{a \in G, a \neq m} \left(\frac{d_G(a, m)}{2} + \frac{d_G^2(b, c)}{8d_G(a, m)} - \frac{d_G^2(a, b) + d_G^2(a, c)}{4d_G(a, m)} \right), \quad (16)$$

where $|V_G|$ is the number of nodes in the graph G , $d_G(a, b)$ is the length of the shortest path between nodes a and b . The sectional curvature $\kappa(m)$ assigned to point m is determined by averaging curvature analogs $\kappa(m; b, c)$ over different neighbors (b, c) . In practice, sampling on the neighbors b, c and reference point a is applied to reduce computational overhead ([Gu et al., 2018](#); [Bachmann et al., 2019](#)).

To assess the topological structure of the graph G , we compute the Gaussian Sectional Curvature (GSC) distribution $\mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$ ([Bachmann et al., 2019](#)). In particular, the mean $\bar{\mu}$ and the variance $\bar{\sigma}^2$ of $\mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$ are calculated as:

$$\bar{\mu} = \frac{1}{|V_G|} \sum_{m \in G} \kappa(m), \bar{\sigma}^2 = \frac{1}{|V_G|} \sum_{m \in G} (\kappa(m) - \bar{\mu})^2. \quad (17)$$

810 GSC distribution $\mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$ provides a comprehensive view of how curvature is distributed across
811 the entire graph, reflecting the geometric complexity of the graph. The mean curvature $\bar{\mu}$ captures
812 the fundamental structural properties of the graph (i.e., whether it is predominantly hierarchical or
813 cyclical), while the variance $\bar{\sigma}^2$ measures the degree of topological heterogeneity. A negative mean
814 $\bar{\mu} < 0$ suggests a more hierarchical structure, favoring embedding in a hyperbolic manifold, whereas
815 a positive mean $\bar{\mu} > 0$ indicates a more cyclical structure, which is suited for a spherical manifold.
816 The variance $\bar{\sigma}^2$ further refines this interpretation: a small variance implies a more homogeneous
817 graph topology, closely aligned with either a hyperbolic or spherical space, while a large variance
818 signifies a more diverse topological structure comprising both cyclical and hierarchical parts.

819 C.2 Complexity

820 To ensure the efficiency of our strategy, we begin by uniformly sampling a subset of nodes \mathcal{A} such
821 that $|\mathcal{A}| \ll |V_G|$. For each node $a \in \mathcal{A}$, we precompute the graph distance $d_G(a, b)$ for all node
822 $b \in G$. Consequently, the curvature analog in Eq. (16) is computed as an average over \mathcal{A} when
823 $m \notin \mathcal{A}$ and over $\mathcal{A} \setminus \{m\}$ when $m \in \mathcal{A}$. The time complexity of our algorithm is $O(|\mathcal{A}|(|V_G| +$
824 $|E_G|) + n_s|\mathcal{A}||V_G| + d)$, where n_s is the number of times we sample two neighbor nodes, $|E_G|$ is the
825 number of edges in graph G , and d is the dimension of embedding space. In contrast, a brute-force
826 approach to selecting hyperparameters p and q has a complexity of $O(n_{\text{iter}}d\Delta)$ where n_{iter} denotes the
827 number of training iterations, and Δ represents the model complexity. For instance, in a simple GCN
828 (Kipf and Welling [2016]), the model complexity is $O(L(|V_G|d^2 + |E_G|d))$, where L is the number
829 of layers, while in a standard linear Transformer (Katharopoulos et al. [2020]), it is $O(|V_G|d^2)$. Since
830 our space searching algorithm scales linearly with d , it remains efficient even for high-dimensional
831 embeddings. Meanwhile, the brute-force approach becomes computationally expensive when both d
832 and n_{iter} are large.

833 C.3 Analysis of bijective function Γ

834 Constructing the bijective function Γ that reflects the relationship between variations in the ideal
835 GSC distribution and the structural characteristics of the pseudo-hyperboloid is a non-trivial task. To
836 facilitate this construction, we analyze the function Γ under three different geometric cases, which
837 are balanced, spherical, and hyperbolic. In the balanced case, the mean curvature μ_i is set as 0
838 when the input graph exhibits the most uniform structure, containing equal proportions of cyclical
839 and hierarchical topologies. This structure is best embedded in the balanced manifold $\mathcal{Q}_\beta^{p_i, q_i}$ where
840 $p_i = q_i = (d - 1)/2$. The standard variance σ_i can take any arbitrary value, and we assign it to 1
841 for simplicity. In the spherical case, when the manifold $\mathcal{Q}_\beta^{p_i, q_i}$ becomes more spherical by setting
842 $p_i < (d - 1)/2 < q_i$, it best corresponds to a graph with a positive mean curvature. Therefore, we set
843 $\mu_i = |\bar{\mu}|$, where $\bar{\mu}$ is the estimated mean curvature of input graph. The variance σ_i^2 , defined as in Eq.
844 (15), increases monotonically with increasing p_i . This is because as p_i grows, the manifold $\mathcal{Q}_\beta^{p_i, q_i}$
845 shifts toward a more balanced structure, which is suited for a uniform topological graph with large
846 variance in its GSC distribution. Notably, the standard deviation σ_i reaches its minimum value σ_{\min}
847 when $p_i = 0$ (a fully spherical manifold), and its maximum value σ_{\max} when $p_i = (d - 1)/2 - 1$.
848 We determine σ_{\max} such that the probability density function at zero curvature, $f(0)$, is nearly equal
849 to $f(\mu_i)$ for the highest variance, while σ_{\min} ensures that $f(0) \ll f(\mu_i)$ for the lowest variance.
850 In practice, we assign $\sigma_{\min} = 0.5|\bar{\mu}|$ and $\sigma_{\max} = 5|\bar{\mu}|$. Similarly, we define the GSC distribution
851 $\mathcal{N}(\mu_i, \sigma_i^2)$ for the hyperbolic case in Eq. (15) where $p_i > (d - 1)/2 > q_i$. The key difference is that,
852 unlike the spherical case, the mean curvature μ_i is negative and the standard deviation σ_i decreases
853 monotonically as p_i increases.

854 D Geodesic tools of pseudo-hyperboloid

855 D.1 Parallel transport

856 Given two points x and y that are connected by a geodesic γ on the manifold $\mathcal{Q}_\beta^{p, q}$, ξ is a vector in
857 the tangent space $\mathcal{T}_x \mathcal{Q}_\beta^{p, q}$, the parallel transport is formulated by Law [2021]:

$$P_{x \rightarrow y}^\gamma(\xi) = \xi - \frac{\langle \xi, y \rangle_q}{\langle x, y \rangle_q - |\beta|} (y + x). \quad (18)$$

858 D.2 Distance

859 When $x, y \in \mathcal{Q}_\beta^{p,q}$ are geodesically connected, the induced distance between x and y in pseudo-
 860 hyperboloid is given by $d_\gamma(x, y) = \sqrt{\|\log_x(y)\|_q^2}$. However, in cases where $\log_x(y)$ is not defined
 861 due to geodesic disconnection, [Xiong et al. \(2021\)](#) proposed an approximation using midpoints.
 862 Specifically, the antipodal point $-x$ serves as an intermediary for the broken geodesic between x and
 863 y , leading to the distance approximation:

$$D_\gamma(x, y) = \begin{cases} d_\gamma(x, y), & \text{if } \langle x, y \rangle_q < |\beta| \\ d_\gamma(x, -x) + d_\gamma(-x, y) = \pi\sqrt{|\beta|} + d_\gamma(-x, y), & \text{if } \langle x, y \rangle_q \geq |\beta| \end{cases} \quad (19)$$

864 D.3 Exponential and logarithmic maps

865 The closed form expressions of two tangent maps $\exp_x^\beta(\xi)$ and $\log_x^\beta(y)$ proposed by [Law and Stam](#)
 866 [\(2020\)](#):

$$\exp_x^\beta(\xi) = \begin{cases} \cosh\left(\frac{\sqrt{|\langle \xi, \xi \rangle_q|}}{\sqrt{|\beta|}}\right) x + \frac{\sqrt{|\beta|}}{\sqrt{|\langle \xi, \xi \rangle_q|}} \sinh\left(\frac{\sqrt{|\langle \xi, \xi \rangle_q|}}{\sqrt{|\beta|}}\right) \xi, & \text{if } \langle \xi, \xi \rangle_q > 0 \\ x + \xi, & \text{if } \langle \xi, \xi \rangle_q = 0 \\ \cos\left(\frac{\sqrt{|\langle \xi, \xi \rangle_q|}}{\sqrt{|\beta|}}\right) x + \frac{\sqrt{|\beta|}}{\sqrt{|\langle \xi, \xi \rangle_q|}} \sin\left(\frac{\sqrt{|\langle \xi, \xi \rangle_q|}}{\sqrt{|\beta|}}\right) \xi, & \text{if } \langle \xi, \xi \rangle_q < 0 \end{cases} \quad (20)$$

$$\log_x^\beta(y) = \begin{cases} \frac{\cosh^{-1}\left(\frac{\langle x, y \rangle_q}{\beta}\right)}{\sqrt{\left(\frac{\langle x, y \rangle_q}{\beta}\right)^2 - 1}} \left(y - \frac{\langle x, y \rangle_q}{\beta} x\right), & \text{if } \frac{\langle x, y \rangle_q}{|\beta|} < -1 \\ y - x, & \text{if } \frac{\langle x, y \rangle_q}{|\beta|} = -1 \\ \frac{\cos^{-1}\left(\frac{\langle x, y \rangle_q}{\beta}\right)}{\sqrt{1 - \left(\frac{\langle x, y \rangle_q}{\beta}\right)^2}} \left(y - \frac{\langle x, y \rangle_q}{\beta} x\right), & \text{if } \frac{\langle x, y \rangle_q}{|\beta|} \in (-1, 1) \end{cases} \quad (21)$$

867 where $\xi \in \mathcal{T}_x \mathcal{Q}_\beta^{p,q}$ is a tangent vector and y is a point on the manifold $\mathcal{Q}_\beta^{p,q}$.

868 E Proof of Theorem 1

869 **Theorem 1.** For any point $x \in \hat{\mathcal{Q}}_\beta^{p,q+1}$, there exists a diffeomorphism $\Psi : x \in \hat{\mathcal{Q}}_\beta^{p,q+1} \rightarrow \mathbb{S}_{|\beta|}^q \times \mathbb{L}_\beta^p$
 870 that maps x onto the product manifolds of a sphere and a hyperboloid. The mapping and its inverse
 871 are given by:

$$\Psi(x) = \begin{pmatrix} \sqrt{|\beta|} \frac{u}{\|u\|} \\ v \end{pmatrix}, \quad \Psi^{-1}(z) = \begin{pmatrix} 0 \\ \frac{v'_0}{\sqrt{|\beta|}} u' \\ v'_{[1:]}\end{pmatrix}, \quad (22)$$

872 where $x = \begin{pmatrix} 0 \\ u \\ v \end{pmatrix} \in \hat{\mathcal{Q}}_\beta^{p,q+1}$ with $u \in \mathbb{R}^{q+1} \setminus \{0\}$, $v \in \mathbb{R}^p$, and $z = \begin{pmatrix} u' \\ v' \end{pmatrix}$ with $u' \in \mathbb{S}_{|\beta|}^q$, $v' \in \mathbb{L}_\beta^p$.

873 *Proof.* It is easy to show that Ψ and Ψ^{-1} are smooth since their elemental functions
 874 $\{\sqrt{|\beta|} \frac{u}{\|u\|}, \|u\|, v, \frac{v'_0}{\sqrt{|\beta|}} u', v'_{[1:]}\}$ are smooth with $u \in \mathbb{R}^{q+1} \setminus \{0\}$, $v \in \mathbb{R}^p$, $u' \in \mathbb{S}_{|\beta|}^q$, $v' \in \mathbb{L}_\beta^p$. We
 875 only need to prove that $\Psi(\Psi^{-1}(z)) = z$ and $\Psi^{-1}(\Psi(x)) = x$.

876 In the first term $\Psi(\Psi^{-1}(z))$, we have:

$$\Psi(\Psi^{-1}(z)) = \Psi\left(\begin{pmatrix} 0 \\ \frac{v'_0}{\sqrt{|\beta|}} u' \\ v'_{[1:]}\end{pmatrix}\right) = \begin{pmatrix} \sqrt{|\beta|} \frac{\frac{v'_0}{\sqrt{|\beta|}} u'}{\left\|\frac{v'_0}{\sqrt{|\beta|}} u'\right\|} \\ \left\|\frac{v'_0}{\sqrt{|\beta|}} u'\right\| \\ v'_{[1:]}\end{pmatrix} = \begin{pmatrix} \sqrt{|\beta|} \frac{u'}{\|u'\|} \\ \frac{v'_0}{\sqrt{|\beta|}} \|u'\| \\ v'_{[1:]}\end{pmatrix}. \quad (23)$$

877 Since $u' \in \mathbb{S}_{|\beta|}^q$, we have $\|u'\| = \sqrt{|\beta|}$. Eq. (23) becomes:

$$\Psi(\Psi^{-1}(z)) = \begin{pmatrix} u' \\ v'_0 \\ v'_{[1:]}\end{pmatrix} = \begin{pmatrix} u' \\ v' \end{pmatrix} = z. \quad (24)$$

878 For the second term $\Psi^{-1}(\Psi(x))$, we have:

$$\Psi^{-1}(\Psi(x)) = \Psi^{-1}\left(\begin{pmatrix} \sqrt{|\beta|} \frac{u}{\|u\|} \\ v \end{pmatrix}\right) = \begin{pmatrix} 0 \\ \frac{\|u\|}{\sqrt{|\beta|}} \sqrt{|\beta|} \frac{u}{\|u\|} \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ u \\ v \end{pmatrix} = x \quad (25)$$

879 □

880 F Proofs of Propositions 1 and 2

881 **Proposition 1.** *The extrinsic mapping function Φ and its inverse Φ^{-1} are smooth, bijective, and*
882 *isometric.*

883 *Proof.* Since both the extrinsic function Φ and its inverse Φ^{-1} are linear, they are inherently *smooth*
884 *mappings.*

885 We also have $\Phi^{-1}(\Phi(x)) = x$ and $\Phi(\Phi^{-1}(y)) = y$, which proves the *bijective* properties of two
886 functions.

887 To prove the isometric property, let x, y are two points on pseudo-hyperboloid $\mathcal{Q}_{\beta}^{p,q}$, we have:

$$\langle x, y \rangle_q = \langle \Phi(x), \Phi(y) \rangle_{q+1} \text{ and } \left\| \left(y - \frac{\langle x, y \rangle_q}{\beta} x \right) \right\|_q = \left\| \left(\Phi(y) - \frac{\langle \Phi(x), \Phi(y) \rangle_{q+1}}{\beta} \Phi(x) \right) \right\|_{q+1}.$$

888 Hence, we have $\|\log_x(y)\|_q = \|\log_{\Phi(x)}(\Phi(y))\|_{q+1}$ or $d_{\gamma}(x, y) = d_{\gamma}(\Phi(x), \Phi(y))$ when x and
889 y are geodesically connected. In the case of geodesically disconnection, it is easy to show that
890 $d_{\gamma}(-x, y) = d_{\gamma}(-\Phi(x), \Phi(y))$, leading to the $D_{\gamma}(x, y) = D_{\gamma}(\Phi(x), \Phi(y))$ where D_{γ} is the ap-
891 proximated distance in Eq. (19). Therefore, Φ is *isometric*. The isometric property proof for the
892 inverse function Φ^{-1} is similar. □

893 **Proposition 2.** *Given two vectors $x, y \in \mathcal{Q}_{\beta}^{p,q}$, the closed form of pseudo-hyperboloid vector addition*
894 *is expressed as:*

$$z = y \oplus_{\beta} x = \Phi(x) + \frac{\langle \Phi(x), \Phi(y) \rangle_{q+1}}{|\beta|} (\bar{\mathbf{o}} + \Phi(y)).$$

895 *Proof.* Beginning from the Definition 2 we have the formula of vector addition:

$$z = \exp_{\Phi(y)}^{\beta}(P_{\bar{\mathbf{o}} \rightarrow \Phi(y)}^{\gamma}(\log_{\bar{\mathbf{o}}}^{\beta}(\Phi(x)))).$$

896 Since $\langle \Phi(x), \bar{\mathbf{o}} \rangle_{q+1} = 0$, following the definition of logarithmic map in Eq. (21), we have:

$$\log_{\bar{\mathbf{o}}}^{\beta}(\Phi(x)) = \frac{\pi}{2} \Phi(x).$$

897 Applying the parallel transport $P_{\bar{\mathbf{o}} \rightarrow \Phi(y)}^{\gamma}$ in Eq. (18), we have:

$$P_{\bar{\mathbf{o}} \rightarrow \Phi(y)}^{\gamma}\left(\frac{\pi}{2} \Phi(x)\right) = \frac{\pi}{2} \Phi(x) + \frac{\pi}{2} \frac{\langle \Phi(x), \Phi(y) \rangle_{q+1}}{|\beta|} (\bar{\mathbf{o}} + \Phi(y)) = \frac{\pi}{2} \xi.$$

Now we calculate the squared norm of ξ :

$$\begin{aligned}
\langle \xi, \xi \rangle_{q+1} &= \langle \Phi(x) + \frac{\langle \Phi(x), \Phi(y) \rangle_{q+1}}{|\beta|} (\bar{o} + \Phi(y)), \Phi(x) + \frac{\langle \Phi(x), \Phi(y) \rangle_{q+1}}{|\beta|} (\bar{o} + \Phi(y)) \rangle_{q+1} \\
&= -\frac{(\langle \Phi(x), \Phi(y) \rangle_{q+1})^2}{|\beta|} + \langle x + \frac{\langle x, y \rangle_q}{|\beta|} y, x + \frac{\langle x, y \rangle_q}{|\beta|} y \rangle_q \\
&= -\frac{(\langle x, y \rangle_q)^2}{|\beta|} + \langle x, x \rangle_q + \left(\frac{\langle x, y \rangle_q}{|\beta|} \right)^2 \langle y, y \rangle_q + 2 \frac{\langle x, y \rangle_q}{|\beta|} \langle x, y \rangle_q \\
&= -\frac{(\langle x, y \rangle_q)^2}{|\beta|} + \beta + \left(\frac{\langle x, y \rangle_q}{|\beta|} \right)^2 \beta + 2 \frac{\langle x, y \rangle_q}{|\beta|} \langle x, y \rangle_q \\
&= -\frac{(\langle x, y \rangle_q)^2}{|\beta|} + \beta - \frac{(\langle x, y \rangle_q)^2}{|\beta|} + 2 \frac{(\langle x, y \rangle_q)^2}{|\beta|} = \beta < 0.
\end{aligned}$$

Since the squared norm $\langle \frac{\pi}{2}\xi, \frac{\pi}{2}\xi \rangle_{q+1} = \left(\frac{\pi}{2}\right)^2 \beta < 0$, following the definition of exponential map in Eq. (20), we have:

$$z = \exp_{\Phi(y)}^{\beta} \left(\frac{\pi}{2} \xi \right) = \xi.$$

This finishes the proof. \square

G Pseudo-Riemannian graph convolutional network

This section describes the pseudo-Riemannian graph convolutional network, \mathcal{Q} -GCN2, which builds upon the diffeomorphic framework introduced in Section 3. \mathcal{Q} -GCN2 comprises three fundamental operations: linear transformation, neighborhood aggregation, and nonlinear activation. As the linear transformation and nonlinear activation have been detailed in Sections 4.1 and 4.2, respectively, we focus here on the formulation of neighborhood aggregation in pseudo-Riemannian geometry. We then present the complete structure of \mathcal{Q} -GCN2.

G.1 Pseudo-Riemannian neighborhood aggregation

Neighborhood aggregation in GCNs (Kipf and Welling, 2016; Chami et al., 2019) relies on scalar multiplication, an operation that has not been defined on the pseudo-hyperboloid $\mathcal{Q}_{\beta}^{p,q}$. To address this, we instead formulate the neighborhood aggregation on the product of spherical and hyperbolic manifolds by using our diffeomorphic framework. Given the neighborhood $\mathcal{N}(i)$ of node i and their features in the product space $\{x_j^{\mathbb{S}_{|\beta|}^q \times \mathbb{L}_{\beta}^p}\}_{j \in \mathcal{N}(i)}$ where each node vector $x_j^{\mathbb{S}_{|\beta|}^q \times \mathbb{L}_{\beta}^p} = (u_j \in \mathbb{S}_{|\beta|}^q, v_j \in \mathbb{L}_{\beta}^p)^T$, and a set of positive weights $\{w_{i,j}\}_{j \in \mathcal{N}(i)}$, we propose the neighborhood aggregation at node i as follows:

$$\tilde{x}_i^{\mathbb{S}_{|\beta|}^q \times \mathbb{L}_{\beta}^p} = \text{Agg}(\{x_j^{\mathbb{S}_{|\beta|}^q \times \mathbb{L}_{\beta}^p}\}, \{w_{i,j}\}) = \sqrt{|\beta|} \left(\frac{\sum_{j \in \mathcal{N}(i)} w_{i,j} u_j}{\|\sum_{j \in \mathcal{N}(i)} w_{i,j} u_j\|}, \frac{\sum_{j \in \mathcal{N}(i)} w_{i,j} v_j}{\sqrt{\|\sum_{j \in \mathcal{N}(i)} w_{i,j} v_j\|_{\mathbb{L}}^2}} \right)^T, \quad (26)$$

where $\|x\|_{\mathbb{L}}^2 = \langle x, x \rangle_{\mathbb{L}}$ is the Lorentz squared norm. The aggregating function in Eq. (26) is the extension of a hyperbolic centroid formulation proposed by Law et al. (2019). Since the squared distance in the product space can be decomposed to the sum of squared distances in the component space (Gu et al., 2018), i.e. $\|\tilde{x}_i - x_j\|_{\mathbb{S} \times \mathbb{L}}^2 = \|\tilde{x}_i - x_j\|_{\mathbb{S}}^2 + \|\tilde{x}_i - x_j\|_{\mathbb{L}}^2$, we can find the centroid that minimizes the weighted sum of squared distances in the product space by concatenating two centroids in spherical and hyperbolic spaces. In other words, our neighborhood aggregation is a centroid with respect to the squared distance in the product space. We first calculate the spherical centroid from the feature set $\{u_j\}_{j \in \mathcal{N}(i)}$, then combine with the hyperbolic centroid of a set $\{v_j\}_{j \in \mathcal{N}(i)}$.

G.2 Overall structure

The development of the \mathcal{Q} -GCN2 model is a direct consequence of our proposed diffeomorphic framework and the Transformer model \mathcal{Q} -GT. With all key operations of \mathcal{Q} -GCN2 introduced, we now

summarize the structure of a single \mathcal{Q} -GCN2 layer. Given the node features $X^{\mathcal{Q}_{\beta}^{p,q}} = \{x_i^{\mathcal{Q}_{\beta}^{p,q}}\}_{i=1}^N$ of input graph G , we follow the procedure in \mathcal{Q} -GT to project them onto the product space, resulting in $X^{\mathbb{S}_{|\beta|}^q \times \mathbb{L}_{\beta}^p} = \Psi(\Phi(X^{\mathcal{Q}_{\beta}^{p,q}}))$. Next, the representations $X^{\mathbb{S}_{|\beta|}^q \times \mathbb{L}_{\beta}^p}$ is sent to the message-passing structure in a \mathcal{Q} -GCN2 layer:

$$\hat{x}_i^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}} = \left(\hat{u}_i = \left(\sqrt{|\beta|} W_1 u_i / \|W_1 u_i\| \right), \hat{v}_i = \left(\sqrt{\|W_2 v_i\|^2 - \beta}, W_2 v_i \right) \right)^T, \quad (27)$$

$$\tilde{x}_i^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}} = \text{Agg}(\{\hat{x}_j^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}}\}, \{w_{i,j}\})_{j \in \mathcal{N}(i)}, \quad (28)$$

$$\bar{x}_i^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}} = \sigma_{\text{Activation}}(\tilde{x}_i^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}}), \quad (29)$$

where $W_1 \in \mathbb{R}^{(q'+1) \times (q+1)}$, $W_2 \in \mathbb{R}^{(p') \times (p+1)}$ are two transformation matrices, $w_{i,j} = ((\deg(j) + 1)(\deg(i) + 1))^{-1/2}$ is the normalized weight from GCN (Kipf and Welling, 2016), and $\deg(i)$ is the degree of node i . Finally, the output $\bar{X}^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}} = \{\bar{x}_i^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}}\}_{i=1}^N$ are projected back to the pseudo-Riemannian manifold by using our diffeomorphic framework, i.e. $\bar{X}^{\mathcal{Q}_{\beta}^{p',q'}} = \Phi^{-1}(\Psi^{-1}(\bar{X}^{\mathbb{S}_{|\beta|}^{q'} \times \mathbb{L}_{\beta}^{p'}}))$.

G.3 Integrating into Graph Transformer

Incorporating GCN to Graph Transformer is well-known for graph representation learning (Wu et al., 2023; Yang et al., 2024), which enhances the local structural information. Let p^* and q^* denote the optimal space and time dimensions, estimated by our searching algorithm in Section 5. Given the outputs of \mathcal{Q} -GCN2 and \mathcal{Q} -GT as $\mathbf{X}_1, \mathbf{X}_2 \in \mathcal{Q}_{\beta}^{p^*,q^*}$, we utilize this approach and combine two outputs with a balance hyperparameter α as follows:

$$\begin{aligned} \bar{\mathbf{X}}_1 &= \log_{\mathbf{O}}^{\beta}(\Phi(\mathbf{X}_1)), \bar{\mathbf{X}}_2 = \log_{\mathbf{O}}^{\beta}(\Phi(\mathbf{X}_2)) \\ \bar{Z} &= \alpha \bar{\mathbf{X}}_1 + (1 - \alpha) \bar{\mathbf{X}}_2 \\ Z &= \exp_{\mathbf{O}}^{\beta}(\bar{Z}) \in \hat{\mathcal{Q}}_{\beta}^{p^*,q^*+1} \end{aligned} \quad (30)$$

where Φ is the extrinsic mapping function in Eq. (3), $\hat{\mathcal{Q}}_{\beta}^{p^*,q^*+1}$ is the zero-first submanifold, $\exp_{\mathbf{O}}^{\beta}$ and $\log_{\mathbf{O}}^{\beta}$ are the exponential and logarithmic maps in Eq. (20) and Eq. (21). In practice, the value of the balance weight α is determined by the grid search strategy.

H More ablation studies

H.1 Effectiveness of space searching algorithm

We further assess the effectiveness of the proposed space searching algorithm on Cora, Citeseer and Pubmed datasets, as visualized in Fig. 4. It is clearly seen that varying space dimensions significantly impact the performance of pseudo-Riemannian models, especially in node classification task. This highlights the importance of selecting an appropriate pseudo-hyperboloid for effective graph representation learning. Our algorithm consistently identifies an optimal or near-optimal space dimension p^* in both tasks. Meanwhile, the performance of \mathcal{Q} -GCN does not align with the space searching result or the underlying topology of the input graph. While \mathcal{Q} -GCN performs reasonably well when the product space becomes Euclidean space through increasing space dimension, Euclidean space lacks the capacity to capture the cyclical and hierarchical structures in many real-world graphs. This limitation leads to inferior performance compared to our proposed \mathcal{Q} -GCN2 and \mathcal{Q} -GT.

H.2 Runtime comparison

For a better complexity comparison, we report the training time of our proposed \mathcal{Q} -GCN2 and \mathcal{Q} -GT against the baseline \mathcal{Q} -GCN in Table 4. All models are trained for 1000 epochs using their corresponding optimal configurations on a 16-dimensional manifold. \mathcal{Q} -GCN suffers higher runtime than \mathcal{Q} -GCN2 due to its reliance on tangent space mappings from the product manifold, which introduces additional computational overhead. This finding underscores the advance of our diffeomorphic framework, which allows operations to be conducted directly on the product

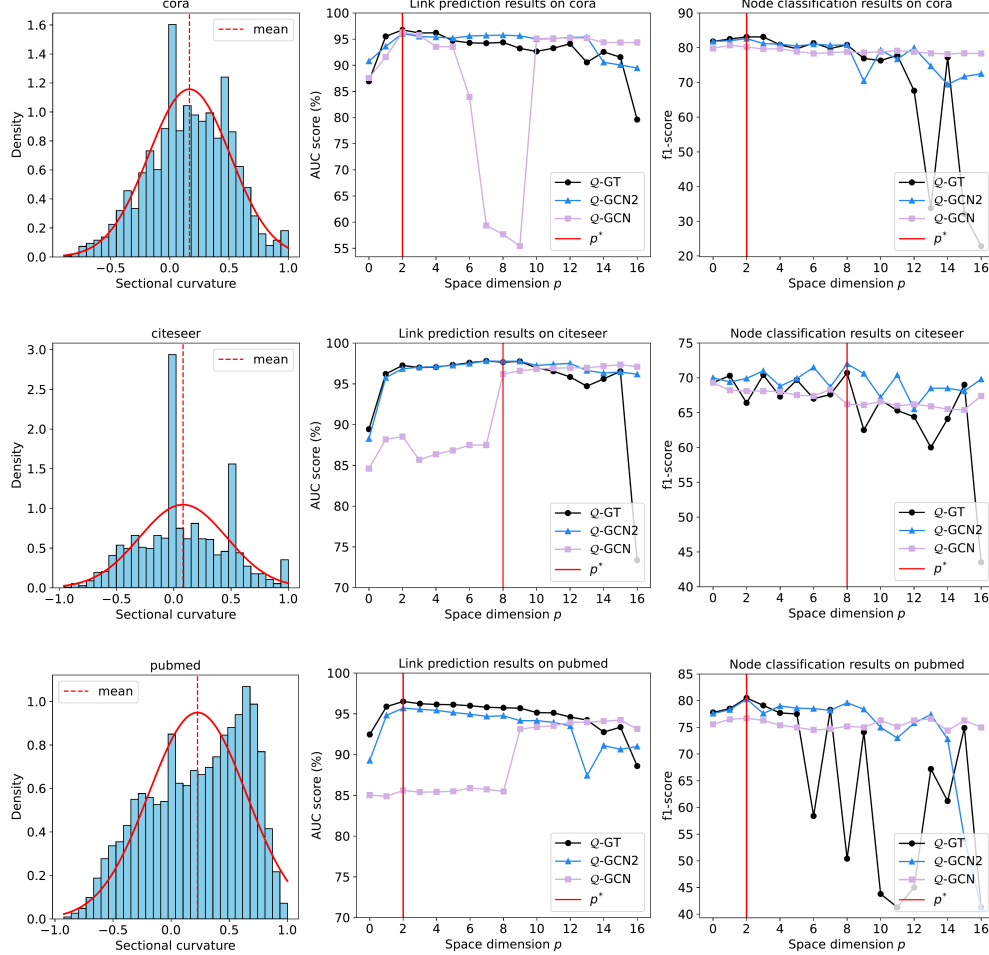


Figure 4: Evaluating space searching algorithm on Cora, Citeseer and Pubmed datasets. The red line denotes space dimension p^* estimated by the searching algorithm.

Table 4: Comparison of training time and the runtime of the space searching algorithm (in second).

Datasets	Q-GCN	Q-GCN2	Q-GT	Space searching	Ratio to Q-GT
Cora	136.6	82.44	193.7	12.70	0.07x
Citeseer	141.8	76.12	188.2	6.90	0.04x
Airport	211.4	66.59	249.5	15.30	0.06x
Pubmed	278.0	83.52	386.0	71.60	0.19x
Arxiv	347.2	145.9	353.5	1922.1	5.43x
Penn94	299.7	121.4	258.5	361.1	1.40x
Twitch Gamers	377.2	181.2	386.3	1611.7	4.18x

manifold rather than in the tangent space Despite that, Q -GT incurs higher training time compared to GNN-based models because it is built upon the Transformer architecture.

In addition, Table 4 presents a comparison between the runtime of the space searching algorithm and the training time of Q -GT across seven datasets. On Cora, Citeseer, Pubmed, and Airport, the computing time searching algorithm is less than 20% training time of Q -GT. However, in the two large datasets, Arxiv and Twitch Gamers, the space searching requires more computation, around 4-5 times higher than the training of Q -GT. Compared to a brute-force approach which requires training the model 16 times (once per each dimension), our algorithm is more efficient in both time and computation.

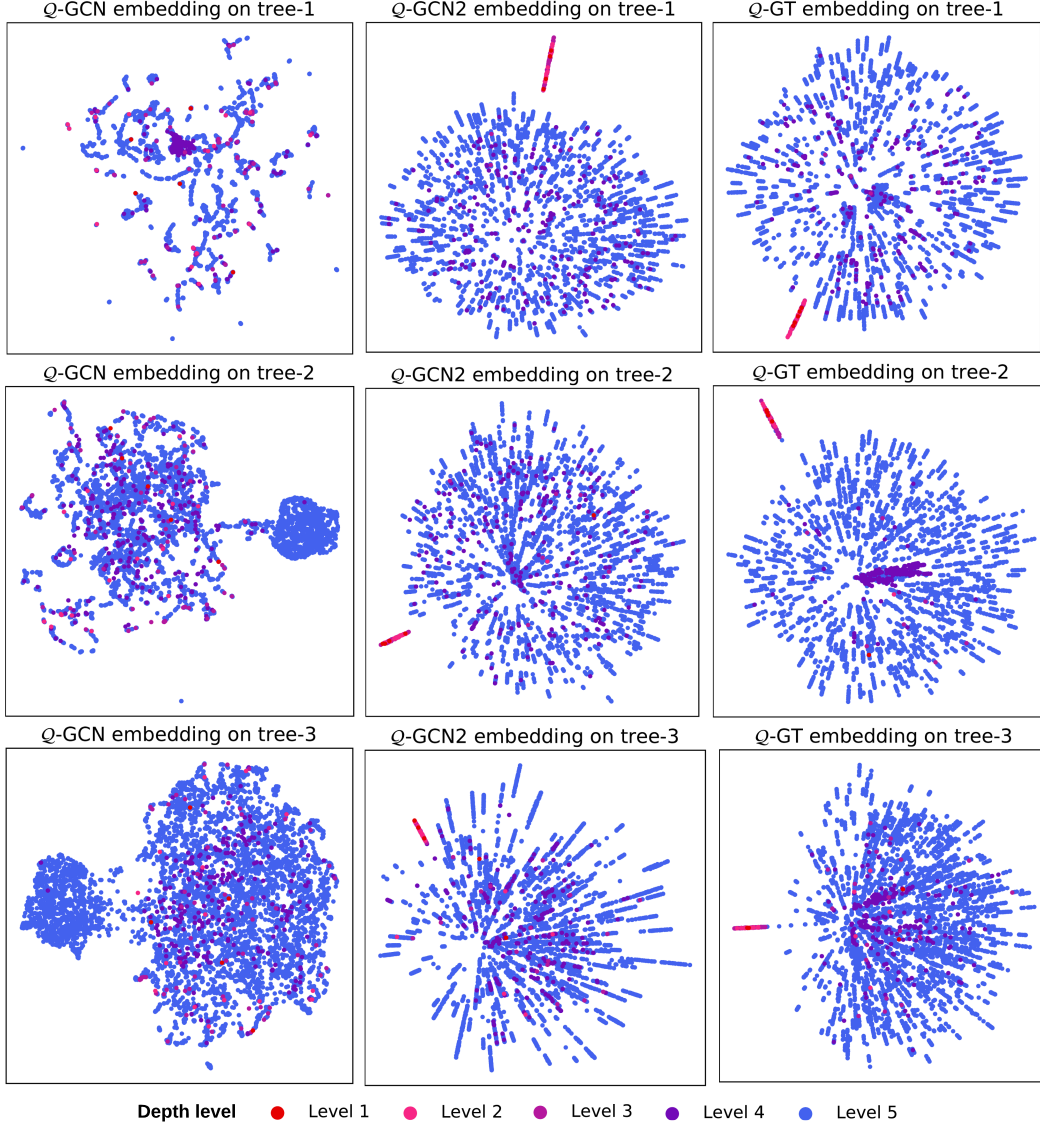


Figure 5: Visualization of the learned embeddings for the link prediction task on Tree-1, Tree-2, Tree-3 datasets. The colors denote depth levels. Nodes at Levels 1 and 2 are classified as low-level, which are closer to the root, while the other are considered high-level.

H.3 Advantage of diffeomorphic framework

To investigate the advantage of the proposed diffeomorphic framework, we create three datasets Tree-1, Tree-2 and Tree-3 that best reflect the hyperbolic geometry. Detailed descriptions of these datasets are provided in Appendix I.1. Table 5 presents the link prediction results of Q-GCN, Q-GCN2 and Q-GT on these graphs. Q-GCN2 and Q-GT significantly outperform Q-GCN across all datasets. For instance, on Tree-2, Q-GT and Q-GCN2 achieve AUC improvements of 9.54 % and 9.05 %, respectively, over Q-GCN. To shed light on the superior performances of our models, we further visualize the learned embeddings on the Tree-1,

Table 5: The link prediction results in ROC AUC (%) on tree datasets.

Datasets	Tree-1	Tree-2	Tree-3
Space dim p^*	16	15	13
Time dim q^*	0	1	3
Q-GCN	91.44 ± 0.52	85.83 ± 0.51	84.86 ± 0.35
Q-GCN2	95.34 ± 0.31	93.60 ± 0.32	89.25 ± 0.34
Q-GT	96.05 ± 0.10	94.02 ± 0.06	92.03 ± 0.11

Table 6: Ablation study on the impact of each component in \mathcal{Q} -GT.

Methods	Datasets			
	Airport	Pubmed	Tree-2	Tree-3
\mathcal{Q} -GT	96.56 ± 0.07	96.40 ± 0.10	94.02 ± 0.06	92.03 ± 0.11
w/o residual	95.22 ± 0.12	96.24 ± 0.15	89.66 ± 2.14	90.70 ± 0.38
<i>Gain(%)</i>	- 1.39	- 0.17	- 4.64	- 1.45
w/o refining	95.73 ± 0.07	95.33 ± 0.28	81.43 ± 0.63	89.98 ± 0.27
<i>Gain(%)</i>	- 0.86	- 1.11	- 13.4	- 2.23
w/o GCN	94.43 ± 0.35	90.47 ± 0.18	86.78 ± 0.49	83.53 ± 1.86
<i>Gain(%)</i>	- 2.21	- 6.15	- 7.70	- 9.24

Table 7: Link prediction results with different embedding dimensions.

Methods	Datasets			
	Airport	Pubmed	Tree-2	Tree-3
Original \mathcal{Q} -GT (16)	96.56 ± 0.07	96.40 ± 0.10	94.02 ± 0.06	92.03 ± 0.11
\mathcal{Q} -GT (8)	95.36 ± 0.28	95.88 ± 0.20	87.83 ± 1.35	85.50 ± 3.64
<i>Gain(%)</i>	- 1.24	- 0.54	- 6.58	- 7.10
\mathcal{Q} -GT (32)	96.09 ± 0.15	96.48 ± 0.10	94.36 ± 0.36	90.29 ± 2.08
<i>Gain(%)</i>	- 0.49	+ 0.08	+ 0.36	- 1.89

Tree-2, and Tree-3 datasets in Fig. 5. Following the experimental setup in Section 6.3, the embeddings are projected to the product spaces, excluding the spherical component for clearer interpretation. In \mathcal{Q} -GCN, low-level nodes are scattered randomly among high-level nodes. \mathcal{Q} -GCN2 and \mathcal{Q} -GT, by contrast, exhibit a clear clustering of low-level nodes, indicating that these models effectively distinguish between low-level and high-level nodes. These observations emphasize the advantage of our diffeomorphic framework for preserving hierarchical structures of graphs.

H.4 Impact of key factors in \mathcal{Q} -GT

To analyze the impact of each key component in our proposed model, we perform an ablation study using three variants derived from \mathcal{Q} -GT:

- **w/o residual:** \mathcal{Q} -GT without residual connections. All residual connections are removed from each layer of the model.
- **w/o refining:** \mathcal{Q} -GT without refinement operations, including layer normalization, dropout, and non-linear activation. This variant consists solely of linear layers.
- **w/o GCN:** A variant of \mathcal{Q} -GT that excludes the graph-based module \mathcal{Q} -GCN2, relying solely on the Transformer module for representation learning.

We use the same experimental setup as \mathcal{Q} -GT to evaluate these variants, and the link prediction results are reported in Table 6. In general, removing any component would lower the model’s performance. **w/o residual** leads to a decline in the model’s performance by 1.91% in average. This suggests that residual connections play an important role in stabilizing training and facilitating gradient flow across layers, which is particularly beneficial in deep architectures like \mathcal{Q} -GT. Similarly, in the case of **w/o refining**, the model’s performance also drops dramatically on Tree-2, which confirms the necessity of refining functions, such as normalization, dropout, and non-linear activation, for effective graph learning in pseudo-Riemannian spaces. When removing graph-based module \mathcal{Q} -GCN2, it causes a significant decrease in the performance of \mathcal{Q} -GT on all datasets. This observation reveals that the Transformer module alone is insufficient for capturing local structural information in the graph.

Additionally, we evaluate the capacity of \mathcal{Q} -GT with different embedding dimensions, 8 and 32, in Table 7. Reducing the dimension to 8 leads to a notable decrease in the link prediction results on hierarchical datasets, whereas increasing to 32 offers only minor improvements. Therefore, we recommend using 16 dimensions to maintain computational efficiency and obtain strong performances.

1016 I Experimental details

Table 8: Summary statistics of the datasets used in this paper.

Datasets	Cora	Citeseer	Airport	Pubmed	Arxiv	Penn94	Twitch Gamers	Tree-1	Tree-2	Tree-3
Total nodes	2708	3327	3188	19717	169343	41554	168114	5461	5461	5461
Total edges	5278	4552	18630	44324	1166234	1362229	6797557	5460	5960	6460
Undirected	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Total classes	7	6	4	3	40	2	2	-	-	-
Features	1433	3703	11	500	128	4814	7	5461	5461	5461
Mean $\bar{\mu}$	0.1658	0.0799	0.3626	0.2264	0.2290	0.2340	0.3238	-0.5983	-0.2924	-0.1417
Std $\bar{\sigma}$	0.2290	0.3716	0.3180	0.4197	0.3030	0.2050	0.2088	0.1082	0.2953	0.2932

1017 I.1 Datasets

1018 Table 8 presents the statistics of all datasets used in this study. The mean $\bar{\mu}$ and standard deviation
1019 $\bar{\sigma}$ denote the Gaussian sectional curvature (GSC) distribution of each input graph. Among the ten
1020 real-world datasets, seven graphs exhibit positive curvature distributions, while other three networks
1021 have negative distributions. In terms of graph size, Arxiv and Twitch Gamers are large-scale networks,
1022 Pubmed and Penn94 are medium-sized, and the remaining datasets are considered small. Detailed
1023 descriptions of each dataset are provided as follows:

- 1024 • **Cora** (McCallum et al., 2000) is a citation network comprising 2708 scientific publications
1025 each node is classified into one of seven classes. The network includes 5278 citation links
1026 among these papers. Each node is represented by a binary word vector, where each entry
1027 indicates the presence or absence of a specific word from a dictionary of 1433 unique terms.
- 1028 • **Citeseer** (Giles et al., 1998) is a citation network consisting of 3327 scientific publications
1029 classified into one of six classes. The citation network consists of 4552 links. Each
1030 publication is described by a binary word vector showing the absence/presence of the
1031 corresponding word from the dictionary of 3703 unique terms.
- 1032 • **Pubmed** (Sen et al., 2008) comprises 19717 scientific articles from the PubMed database
1033 related to diabetes, each categorized into one of three classes. The citation network includes
1034 44324 links between these publications. Each article is represented by a TF-IDF weighted
1035 word vector, constructed from a vocabulary of 500 unique words.
- 1036 • **Airport** (Xiong et al., 2021) is a transportation network containing 3188 airports. Edges
1037 correspond to airline routes connecting the airports. For the node classification task, each
1038 airport is labeled with the population of the country in which it is located.
- 1039 • **Arxiv** (Hu et al., 2020) is a directed citation network between all Computer Science (CS)
1040 arXiv papers. This dataset belongs to the Open Graph Benchmark (OGB) collection.
1041 Each node is an arXiv paper and each directed edge indicates that one paper cites another
1042 one. Each paper comes with a 128-dimensional feature vector obtained by averaging the
1043 embeddings of words in its title and abstract. We apply the space searching algorithm on the
1044 undirected structure of Arxiv network, while training on the original directed graph.
- 1045 • **Penn94** (Lim et al., 2021) is a Facebook "friendship" networks at one hundred American
1046 colleges and universities from 2005. In this graph, nodes correspond to individual students,
1047 each labeled by their reported gender. Node features include academic major, second major
1048 or minor, dormitory or housing assignment, graduation year, and high school background.
- 1049 • **Twitch Gamers** (Lim et al., 2021) is an undirected graph representing connections between
1050 user accounts on the Twitch streaming platform. Nodes correspond to individual Twitch
1051 accounts, with edges indicating mutual follower relationships. Node features include the
1052 number of views, account creation and update timestamps, language, account lifetime, and
1053 a flag indicating whether the account is inactive. The associated binary classification task
1054 involves predicting whether a channel contains explicit content.
- 1055 • **Tree datasets** are created due to the lack of the negative curvature graphs. Tree-1 is
1056 a balanced tree of depth 6 and branching factor 4 consisting of 5461 nodes and 5460
1057 edges. Tree-2 and Tree-3 are derived from Tree-1 by randomly adding 500 and 1000 edge
1058 respectively, introducing different magnitudes of curvature. Each node is initialized with a
1059 one-hot feature vector, and no node labels are provided in these datasets.

1.2 Implementation setup

Objective function. In the node classification task, following Xiong et al. (2021), we project the output of the final layer of both \mathcal{Q} -GT and \mathcal{Q} -GCN2 onto the tangent space and apply Euclidean multinomial logistic regression for classification. For link prediction, non-Euclidean methods utilize the Fermi-Dirac decoder (Krioukov et al., 2010), which computes edge probability scores based on geodesic distances in the embedding space.

Optimization. The trainable parameters of \mathcal{Q} -GT and \mathcal{Q} -GCN2 are defined in Euclidean space via the proposed diffeomorphic framework. As a result, standard Euclidean optimization algorithms, such as Adam (Kingma and Ba, 2014) and SGD (Robbins, 1951), can be directly applied for model training. RiemannianAdam (Béginneul and Ganeva, 2018) is another choice for non-Euclidean methods such as HGCN and Hypformer, which operate under specific manifold constraints.

Setting. We implement our models by Pytorch and Geoopt tool². All the experiments are conducted on a GPU device NVIDIA GeForce RTX 4090 with 24GB memory. For comparison baselines, we follow the implementation of each model introduced in the corresponding paper. The experiment results are reported by the average value with standard deviation on the test sets using three different seeds. Unlike the work of Xiong et al. (2021), we utilize initial features provided in the datasets for node classification instead of pretrained embeddings for evaluating model’s learning capacity.

Table 9: The grid search ranges for hyperparameters.

Hyperparameter	Search range
Learning rate lr	5e-2, 2e-2, 1e-2, 5e-3, 2e-3, 1e-3
Balance weight α	0.5, 0.6, 0.7, 0.8, 0.9
Number of layers	1,2,3,4
Activation	relu, tanh, sigmoid, leakyrelu
Dropout rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7
Weight decay	1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-7

Hyperparameter. Besides the space and time dimensions of pseudo-hyperboloids $\mathcal{Q}_{\beta}^{p,q}$, our models involves other hyperparameters, including learning rate lr , balance weight α , number of layers, activation function, dropout rate, and weight decay. The optimal hyperparameters are obtained by applying grid search strategy, where the ranges summarized in Table 9.

1.3 Pseudocode

In this section, we provide the pseudocode of our proposed framework to facilitate implementation for the reader. Algorithm 1 outlines the procedure of the space searching algorithm. In line 1, a subset \mathcal{A} is uniformly sampled from the node set V_G . Lines 2–6 precompute the shortest path distances from each node $a \in \mathcal{A}$ to all other nodes in the graph G . The Gaussian sectional curvature (GSC) distribution of the input graph is computed in lines 7–14. Finally, in lines 15–20, we iteratively calculate the KL divergence between the input GSC distribution $\mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$ and each ideal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ to determine the optimal space dimension p_i .

Algorithm 2 demonstrates the learning process of \mathcal{Q} -GCN2. In line 1, we apply Algorithm 1 to select the embedding manifold $\mathcal{Q}_{\beta}^{p^*, q^*}$. Line 2 projects the initial features to the embedding manifold with a function f which is either a dense layer or a graph convolution layer to ensure the dimensional consistency. The projected representation is passed through L layers of \mathcal{Q} -GCN2 in lines 3-9, where each layer comprises feature transformation, neighborhood aggregation, and nonlinear activation.

The learning process of \mathcal{Q} -GCT is described in Algorithm 3. Similar to \mathcal{Q} -GCN2, lines 1–2 involve selecting the appropriate manifold and transforming node features. The transformed representation is then processed through an L -layer \mathcal{Q} -GT in lines 3-15. To capture local structural information, \mathcal{Q} -GCN2 is incorporated in line 16. The outputs of \mathcal{Q} -GCN2 and \mathcal{Q} -GCT are combined using a balance weight α in lines 17-20. The final node representation Z is obtained for downstream tasks.

²<https://github.com/geoopt/geoopt>

Algorithm 1 The procedure of space searching algorithm

Input: An input graph $G = (V_G, E_G)$ **Output:** The optimal pseudo-hyperboloid $\mathcal{Q}_\beta^{p^*, q^*}$

```
1: Uniformly sampling a subset of nodes  $\mathcal{A} \subset V_G$ 
2: for each node  $a \in \mathcal{A}$  do
3:   for each node  $b \in V_G$  do
4:     Precomputing the shortest path distance  $d_G(a, b)$ 
5:   end for
6: end for
7: for each node  $m \in V_G$  do
8:   for  $i = 1$  to  $n_s$  do
9:     Uniformly sampling two neighbor nodes  $b, c \in \mathcal{N}(m)$ 
10:    Calculating the curvature analog  $\kappa(m; b, c)$  in Eq. (16)
11:   end for
12:   Calculating the sectional curvature  $\kappa(m)$  by averaging curvature analogs  $\kappa(m; b, c)$ 
13: end for
14: Calculating the mean  $\bar{\mu}$  and the variance  $\bar{\sigma}^2$  in Eq. (17)
15: for  $i = 0$  to  $d - 1$  do
16:   Assigning space dimension  $p_i := i$ 
17:   Calculating the ideal GSC distribution  $\mathcal{N}(\mu_i, \sigma_i^2) = \Gamma(p_i)$ 
18:   Calculating the KL divergence  $d_{KL}(\mathcal{N}(\bar{\mu}, \bar{\sigma}^2), \mathcal{N}(\mu_i, \sigma_i^2))$ 
19:   Saving  $p^* = p_i$  that yields the smallest KL divergence
20: end for
21: return the optimal space dimension  $p^*$ 
```

Algorithm 2 L -layer \mathcal{Q} -GCN2 learning process

Input: An input graph $G = (V_G, E_G)$ with the initial features $X^\mathbb{E}$ **Output:** A hidden representation $X_L^{\mathcal{Q}_\beta^{p^*, q^*}}$

```
1: Determining the embedding manifold  $\mathcal{Q}_\beta^{p^*, q^*}$  by using Algorithm 1
2: Projecting initial features  $X_0^{\mathcal{Q}_\beta^{p^*, q^*}} = \exp_\beta^{\beta}([0 \| f(X^\mathbb{E})])$ 
3: for  $l = 1$  to  $L$  do
4:   Decomposing features  $X_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}} = \Psi(\Phi(X_{l-1}^{\mathcal{Q}_\beta^{p^*, q^*}}))$ 
5:    $\hat{X}_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}} = \left( \sqrt{|\beta|} W_1 X_{l-1}^{\mathcal{S}_{|\beta|}^{q^*}} / \|W_1 X_{l-1}^{\mathcal{S}_{|\beta|}^{q^*}}\|, \sqrt{\|W_2 X_{l-1}^{\mathbb{L}_\beta^{p^*}}\|^2 - \beta}, W_2 X_{l-1}^{\mathbb{L}_\beta^{p^*}} \right)^T$ 
6:    $\tilde{X}_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}} = \text{Agg}(\hat{X}_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}}, E_G)$ 
7:    $\bar{X}_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}} = \sigma_{\text{Activation}}(\tilde{X}_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}})$ 
8:   Projecting back to the pseudo-hyperboloid  $X_l^{\mathcal{Q}_\beta^{p^*, q^*}} = \Phi^{-1}(\Psi^{-1}(\bar{X}_{l-1}^{\mathcal{S}_{|\beta|}^{q^*} \times \mathbb{L}_\beta^{p^*}}))$ 
9: end for
10: return the final representation  $X_L^{\mathcal{Q}_\beta^{p^*, q^*}}$ 
```

Algorithm 3 L -layer \mathcal{Q} -GT learning process

Input: An input graph $G = (V_G, E_G)$ with the initial features $X^{\mathbb{E}}$

Output: A hidden representation Z

- 1: Determining the embedding manifold $\mathcal{Q}_{\beta}^{p^*, q^*}$ by using Algorithm 1
 - 2: Projecting initial features $X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}} = \exp_{\mathbf{0}}^{\beta}([0 || f(X^{\mathbb{E}})])$
 - 3: $X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}} \leftarrow \sigma_{\text{LayerNorm}}(X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 4: $X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}} \leftarrow \text{FeatureTransform}(X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 5: $X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}} \leftarrow \sigma_{\text{Activation}}(X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 6: $X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}} \leftarrow \sigma_{\text{Dropout}}(X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 7: **for** $l = 1$ to L **do**
 - 8: $\hat{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}} = \sigma_{\text{LayerNorm}}(X_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 9: $\tilde{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}} = \text{LinearAttention}(\hat{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 10: $\bar{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*+1}} = \tilde{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}} \oplus_{\beta} X_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}}$
 - 11: $\bar{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}} = \text{FeatureTransform}(\bar{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*+1}})$
 - 12: $\check{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}} = \sigma_{\text{Activation}}(\bar{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 13: $X_l^{\mathcal{Q}_{\beta}^{p^*, q^*}} = \sigma_{\text{Dropout}}(\check{X}_{l-1}^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 14: **end for**
 - 15: $X_L^{\mathcal{Q}_{\beta}^{p^*, q^*}} \leftarrow \text{FeatureTransform}(X_L^{\mathcal{Q}_{\beta}^{p^*, q^*}})$
 - 16: $X_{L'}^{\mathcal{Q}_{\beta}^{p^*, q^*}} = L'$ -layer \mathcal{Q} -GCN2($X_0^{\mathcal{Q}_{\beta}^{p^*, q^*}}$)
 - 17: $\mathbf{X}_1 = \log_{\mathbf{0}}^{\beta}(\Phi(X_L^{\mathcal{Q}_{\beta}^{p^*, q^*}}))$
 - 18: $\mathbf{X}_2 = \log_{\mathbf{0}}^{\beta}(\Phi(X_{L'}^{\mathcal{Q}_{\beta}^{p^*, q^*}}))$
 - 19: $\bar{\mathbf{X}} = \alpha \mathbf{X}_1 + (1 - \alpha) \mathbf{X}_2$
 - 20: $Z = \exp_{\mathbf{0}}^{\beta}(\bar{\mathbf{X}})$
 - 21: **return** the final representation Z
-