CURVE YOUR ATTENTION: MIXED-CURVATURE TRANSFORMERS FOR GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world graphs naturally exhibit hierarchical trees and cyclic structures that are unfit for the typical Euclidean space. While there exist graph neural networks that utilize hyperbolic or spherical spaces towards embedding such structures more accurately, these methods are confined under the message-passing paradigm, making them vulnerable against side-effects such as oversmoothing and oversquashing. More recent work have proposed global attention-based graph Transformers that can alleviate such drawbacks and easily model long-range interactions, but their extensions towards non-Euclidean geometry are yet unexplored. To bridge this gap, we propose Fully Product-Stereographic Transformer, a generalization of Transformers towards operating entirely on the product of constant curvature spaces. When combined with tokenized graph Transformers, our model can learn the curvature appropriate for the input graph in an end-to-end fashion, without any additional tuning on different curvature initializations. We also provide a kernelized approach to non-Euclidean attention, which enables our model to run with computational cost linear to the number of nodes and edges while respecting the underlying geometry. Experiments on graph reconstruction and node classification demonstrate the benefits of generalizing Transformers to the non-Euclidean domain.

1 INTRODUCTION

Learning from graph-structured data is a challenging task in machine learning, with various downstream applications that involve modeling individual entities and relational interactions among them (Sen et al., 2008; Watts & Strogatz, 1998; Gleich et al., 2004). A dominant line of work consists of graph convolutional networks (GCNs) that aggregate features across graph neighbors through *message-passing* (Gilmer et al., 2017; Kipf & Welling, 2016; Veličković et al., 2017; Wu et al., 2019; Hamilton et al., 2017). While most GCNs learn features that lie on the typical Euclidean space with zero curvature, real-world graphs often comprise of complex structures such as hierarchical trees and cycles that Euclidean space requires excessive dimensions to accurately embed (Sala et al., 2018). In response, the graph learning community has developed generalizations of GCNs to spaces with non-zero curvature such as hyperbolic, spherical, or mixed-curvature spaces with both negative and positive curvatures (Chami et al., 2019; Liu et al., 2019; Bachmann et al., 2020; Xiong et al., 2022).

Unfortunately, non-Euclidean GCNs are not immune to harmful side-effects of message-passing such as oversmoothing (Oono & Suzuki, 2019; Cai & Wang, 2020; Yang et al., 2022) and oversquashing (Topping et al., 2021; Alon & Yahav, 2020). These drawbacks make it difficult to stack GCN layers towards large depths, limiting its expressive power (Feng et al., 2022; Maron et al., 2019) as well as predictive performance on tasks that require long-range interactions to solve (Dwivedi et al., 2022; Liu et al., 2021). To cope with such limitations, recent work have instead proposed Transformer-based graph encoders that can easily exchange information across long-range distances through global self-attention (Kim et al., 2022; Ying et al., 2021; Dwivedi & Bresson, 2020; Kreuzer et al., 2021). However, existing graph Transformers are still confined within the Euclidean regime, and their extensions towards non-Euclidean geometry has not yet been studied.

In this paper, we bridge this gap by generalizing the Transformer architecture (Vaswani et al., 2017) towards non-Euclidean spaces with learnable curvatures. Specifically, we endow each attention head a stereographic model (Bachmann et al., 2020) that can universally represent Euclidean, hyperbolic, and

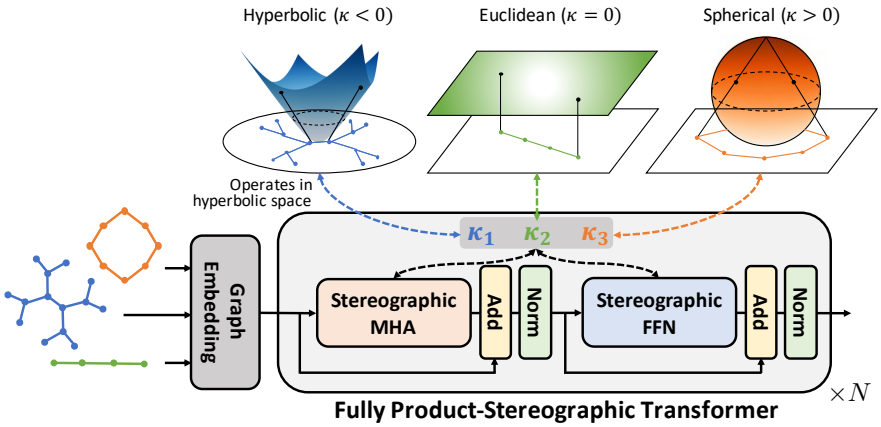


Figure 1: Illustration of our proposed FPS-T architecture. Well-known constant curvature spaces can be projected to the stereographic model, with a common chart map isomorphic to the d-dimensional Euclidean space. Each space can efficiently embed different types of graphs (*e.g.*, trees in hyperbolic space, lines in Euclidean space, and cycles in spherical space). In FPS-T, each layer chooses a set of curvatures that fits the input graph by changing the sign of the curvature κ in a differentiable manner.

spherical spaces (Figure 1). We generalize each operation of the Transformer architecture to inputs on the product-stereographic model, all of which are end-to-end differentiable with respect to the curvatures, thereby allowing the model to jointly train embeddings as well as the underlying curvature. The resulting model, which we name as **Fully Product-Stereographic Transformer (FPS-T)**, takes advantage of both non-Euclidean geometry and long-range interactions. We empirically show that the learnable sectional curvature of FPS-T successfully converges to the geometry of the input graph, leading to better predictive performance and parameter efficiency in graph reconstruction and node classification compared to its Euclidean counterpart. To the best of our knowledge, our work is the first to propose a natural generalization of Transformers to mixed-curvature spaces. We summarize our core contributions as follows:

- We propose FPS-T, a generalization of Transformer towards operating entirely on the product-stereographic model with curvatures that are learnable in an end-to-end fashion.
- For graph representation learning, we integrate FPS-T with Tokenized Graph Transformer (Kim et al., 2022), and develop a kernelized approximation of non-Euclidean attention to reduce the computational cost to linear in number of nodes and edges.
- Graph reconstruction and node classification experiments on synthetic as well as real-world graphs demonstrate the benefit of generalizing Transformers to the mixed-curvature domain.

2 Related Work

Non-Euclidean graph representations. Non-Euclidean spaces are known to well-preserve specific types of graph structure where Euclidean space fails. Especially, non-Euclidean spaces with constant sectional curvature, *e.g.*, hyperbolic and spherical spaces, are widely used in graph representation learning due to its tractable operations. Hyperbolic spaces are capable of efficiently embedding complex hierarchical structures in graphs (Nickel & Kiela, 2018; 2017; Ganea et al., 2018; Krioukov et al., 2010; Sala et al., 2018). Graphs with cyclic structures are well-suited for spherical spaces (Wilson et al., 2014; Grattarola et al., 2019). Riemannian manifolds with varying curvature and constant sign are also proposed for graph encoding (Cruceru et al., 2021). However, Riemannian manifolds where the sign of the curvature is fixed are not a good choice for more complex graphs that exhibit both hierarchy and cycles. Instead, the product of constant-curvature spaces (Gu et al., 2019), heterogeneous manifolds (Giovanni et al., 2022), and pseudo-Riemannian manifolds (Law & Stam, 2020) are found to be well-suited for learning representations of such complex graphs.

Message passing GCNs also benefit from considering a non-Euclidean representation space. Hyperbolic GCNs are known to outperform Euclidean counterparts in various tasks on hierarchical graphs such as citation networks (Chami et al., 2019; Zhang et al., 2021; Pei et al., 2020) and

molecules (Chami et al., 2019; Liu et al., 2019). Deepsphere (Defferrard et al., 2020) also adopted the spherical space to GCNs with applications such as 3D object and earth climate modeling. To take the advantage of multiple spaces, (Zhu et al., 2020b) proposed a hybrid architecture that fuses Euclidean and hyperbolic graph representations together. (Deng et al., 2023) similarly proposed modeling interactions between three constant-curvature spaces (*i.e.*, Euclidean, hyperbolic, and spherical). To allow smooth connections between the three constant-curvature spaces, (Bachmann et al., 2020) proposed a model of constant-curvature space called the stereographic model, on which geometric operations such as distances and inner products are differentiable at all curvature values including zero. Incorporating pseudo-Riemannian manifolds with the GCN architecture also showed promising results (Xiong et al., 2022), but its performance is sensitive to the time dimension of the manifold, which requires extensive hyperparameter tuning.

Overall, GCNs achieve great predictive performance in homophilic graphs where connected nodes share the same features, but they tend to fail in hetereophilic graphs, as stacking up GCN layers to capture message passing between distant nodes induces oversmoothing (Oono & Suzuki, 2019; Cai & Wang, 2020) and oversquashing (Topping et al., 2021). To relieve this architectural limitation while utilizing non-Euclidean geometrical priors, we instead develop a Transformer-based graph encoder that operates on the steregraphic model to learn graph representations.

Graph Transformers. Inspired by huge success of Transformers in NLP and CV (Devlin et al., 2018; Brown et al., 2020; Dosovitskiy et al., 2020), there exist various work that extend Transformers for encoding graphs with edge connectivities that are neither sequential nor grid-like. Graph Transformer (Dwivedi & Bresson, 2020) and Spectral Attention Network (Kreuzer et al., 2021) were the first pioneers to explore this direction by replacing sinusoidal positional encodings widely used in NLP with Laplacian eigenvectors of the input graph. Graphormer (Ying et al., 2021) then proposed utilizing edge connectivities by using shortest-path distances as an attention-bias, showing state-of-the-art performance on molecular property prediction. TokenGT (Kim et al., 2022) proposed a tokenization technique that views each graph as a sequence of nodes and edges. Unlike other methods, TokenGT allows straightforward integration of engineering techniques of pure Transformers such as linearized attention (Katharopoulos et al., 2020), while enjoying theoretical expressivity that surpasses that of message-passing GCNs.

However, existing graph Transformer architectures are yet confined within the Euclidean domain, making them unable to precisely embed graphs onto the feature space similarly to geometric GCNs. While Hyperbolic Attention Network (Gulcehre et al., 2018) proposed an attention mechanism that operates on hyperbolic space, its distance-based approach imposes a computational cost quadratic to the graph size and its geometry is fixed to hyperbolic. Instead, we generalize the representation space of Transformer to stereographic model, which allows us to cover more various types of graphs. We also linearize the attention mechanism on the stereographic model similar to Katharopoulos et al. (2020), which results in a model that runs in cost linear to the number of nodes and edges.

3 Preliminaries

In this section, we introduce concepts related to our main geometrical tool, the product-stereographic model (Bachmann et al., 2020). We also discuss multi-head attention, the driving force of the Transformer architecture (Vaswani et al., 2017).

3.1 Product-Stereographic Model

Riemannian manifolds. A Riemannian manifold is consisted of a smooth manifold $\mathcal{M}$ and a metric tensor g. Each point $\boldsymbol{x}$ on the manifold $\mathcal{M}$ defines a tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, which is a collection of all vectors that are tangent to $\boldsymbol{x}$, also called the tangent vector. The metric tensor $g : \mathcal{M} \to \mathbb{R}^{n\times n}$ assigns a positive-definite matrix to each point $\boldsymbol{x}$, which defines its inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$ as $\boldsymbol{v}_1^T g(\boldsymbol{x})\boldsymbol{v}_2$ where $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ are the tangent vectors of $\boldsymbol{x}$. The metric tensor also defines geometrical properties and operations on the Riemannian manifold. Geodesic γ is the shortest curve between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$ and its distance can be computed as $d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y}) = \int_0^1 \langle\dot{\gamma}(t), \dot{\gamma}(t)\rangle_{\gamma(t)} dt$, where $\gamma : [0, 1] \to \mathcal{M}$ is a unit-speed curve satisfying $\gamma(0) = \boldsymbol{x}$ and $\gamma(1) = \boldsymbol{y}$. We can move the point $\boldsymbol{x} \in \mathcal{M}$ along a tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ using exponential map $\exp_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ which is defined as $\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \gamma(1)$ where γ is a geodesic and $\gamma(0) = \boldsymbol{x}, \dot{\gamma}(0) = \boldsymbol{v}$. The logarithmic map $\log_{\boldsymbol{x}} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ is the inverse of $\exp_{\boldsymbol{x}}$. A tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ can be transferred along a geodesic from $\boldsymbol{x}$ to $\boldsymbol{y}$ using parallel transport $\mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{T}_{\boldsymbol{y}}\mathcal{M}$.

Note that the product of Riemannian manifolds is also a Riemannian manifold. A point on the product Riemannian manifold $\boldsymbol{x} \in \otimes_{i=1}^{n} \mathcal{M}_i$ is consisted of points from each component manifold $\mathcal{M}_i$ as $\boldsymbol{x} = \|_{i=1}^{n} \boldsymbol{x}_i$, where $\boldsymbol{x}_i \in \mathcal{M}_i$ and $\|$ denotes concatenation. The distance between $\boldsymbol{x}, \boldsymbol{y} \in \otimes_{i=1}^{n} \mathcal{M}_i$ is calculated as $\sqrt{\sum_{i=1}^{n} d^2_{\mathcal{M}_i}(\boldsymbol{x}_i, \boldsymbol{y}_i)}$. Exponential/logarithmic maps and parallel transports are applied in a manifold-wise fashion (*e.g.*, $\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \|_{i=1}^{n} \exp_{\boldsymbol{x}_i}(\boldsymbol{v}_i)$ with $\boldsymbol{v} = \|_{i=1}^{n} \boldsymbol{v}_i$ and $\boldsymbol{v}_i \in \mathcal{T}_{\boldsymbol{x}_i} \mathcal{M}_i$).

Constant-curvature spaces. Curvature is an important geometrical property used to characterize Riemannian manifolds. One widely-used curvature to explain Riemannian manifolds is the sectional curvature: given two linearly independent tangent vector fields $U, V \in \mathfrak{X}(\mathcal{M})$, the sectional curvature $K(U, V)$ is computed as $K(U, V) = \frac{\langle R(U,V)V, U \rangle}{\langle U,U \rangle \langle V,V \rangle - \langle U,V \rangle^2}$, where $R(\cdot, \cdot) : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$ is a Riemannian curvature tensor. The sectional curvature measures the divergence between geodesics starting with the tangent vector fields U, V for each point of the manifold. With positive or negative sectional curvatures, geodesics become closer or farther than with zero curvature. Throughout this paper, we refer to a space of a constant sectional curvature as a *constant-curvature space*. For example, the Euclidean space $\mathbb{E}$ is the special case of the constant-curvature space with zero curvature. When positive or negative, we call the corresponding spaces to be hyperbolic $\mathbb{H}$ or spherical $\mathbb{S}$.

Stereographic models. A d-dimensional stereographic model $\mathfrak{st}^d_\kappa$ is a constant-curvature space with curvature $\kappa \in \mathbb{R}$. One attractive property of the stereographic model is that the operations such as distance, exp/log-map, and parallel transport are differentiable at any curvature value κ, including $\kappa = 0$. This enables the stereographic model to learn the curvature value κ without any constraint.

The manifold of the stereographic model $\mathfrak{st}^d_\kappa$ is $\{\boldsymbol{x} \in \mathbb{R}^d | -\kappa \|\boldsymbol{x}\|^2 < 1\}$. The metric tensor is defined as $g^\kappa(\boldsymbol{x}) = \frac{4}{1+\kappa\|\boldsymbol{x}\|^2} \boldsymbol{I} =: (\lambda^\kappa_{\boldsymbol{x}})^2 \boldsymbol{I}$, where $\lambda^\kappa_{\boldsymbol{x}}$ is known as the conformal factor. The mobius addition between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathfrak{st}^d_\kappa$ is computed as $\boldsymbol{x} \oplus_\kappa \boldsymbol{y} = \frac{(1-2\kappa \boldsymbol{x}^T \boldsymbol{y} - \kappa\|\boldsymbol{y}\|^2)\boldsymbol{x} + (1+\kappa\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1-2\kappa \boldsymbol{x}^T \boldsymbol{y} + \kappa^2 \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2}$. Based on mobius addition, we can derive other geometric operations as Table 3 in Appendix A. The table also shows that when κ converges to zero, the operations become equivalent to Euclidean space operations, so the stereographic model essentially recovers Euclidean geometry.

3.2 Multi-Head Attention

In vanilla Transformer (Vaswani et al., 2017), each block consists of multiple attention heads, each taking a sequence of token embeddings $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ with sequence length n and feature dimension d as input. Three linear layers $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{V}^V \in \mathbb{R}^{d \times d'}$ first map each token embedding into queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$ with head-dimension d', respectively. Then, the attention score matrix is computed by scaled Euclidean dot-product between $\boldsymbol{Q}$ and $\boldsymbol{K}$, followed by row-wise softmax activation $\sigma(\cdot)$. The attention score matrix is then multiplied to value $\boldsymbol{V}$, returning contextualized token embeddings. The overall procedure can be written as

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V, \quad \text{Attn}(\boldsymbol{X}) = \sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d'}}\right)\boldsymbol{V}. \tag{1}$$

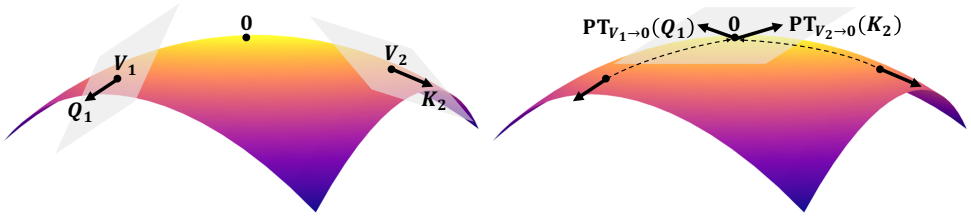
The output from multiple attention heads are concatenated together, then processed through a feed-forward layer before proceeding to the next Transformer block.

4 Fully Product-Stereographic Transformer

Here, we describe the inner wirings of our proposed method. We generalize each operation in Transformer to the product-stereographic model, together forming a geometric Transformer architecture that operates entirely within the stereographic model.

4.1 Stereographic Neural Networks

We first introduce the stereographic analogies of the Euclidean neural networks such as the linear layer, activation, layer normalization, and logit functions. We denote the product-stereographic model $\otimes_{i=1}^{H} \mathfrak{st}^d_{\kappa_i}$ as $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}}$, where $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_H)$ is the ordered set of curvatures of d-dimensional component spaces within a Transformer block with H attention heads. We also use the superscript $\otimes\boldsymbol{\kappa}$ to denote Riemannian operations on product-stereographic model that decompose representations into equal parts, apply the operation, then concatenate back to the product space (*e.g.*, if $\boldsymbol{v} = [v_1, \ldots, v_H]$, then $\exp_{\boldsymbol{0}}^{\otimes\boldsymbol{\kappa}}(\boldsymbol{v}) \coloneqq \|_{i=1}^{H} \exp_{\boldsymbol{0}}^{\kappa_i}(v_i)$).



(a) QKV mapping (b) Parallel transport and inner product

Figure 2: Illustration of our attention mechanism on the non-Euclidean space. FPS-T considers each value-vector as a point that resides on the stereographic model, and query/key-vectors as tangent vectors on the corresponding tangent spaces. All query/key-vectors are parallel-transported to the origin prior to dot-product attention, thereby taking the given geometry into account.

Stereographic linear layer, activation, and layer normalization. Given a Euclidean neural network f, we can define its stereographic counterpart as $\exp_{\mathbf{0}}^{\otimes_{\boldsymbol{\kappa}}}\left(f\left(\log_{\mathbf{0}}^{\otimes_{\boldsymbol{\kappa}}}(\boldsymbol{X})\right)\right)$. The stereographic linear layer $\mathrm{Linear}_{\otimes_{\boldsymbol{\kappa}}}(\boldsymbol{X};\boldsymbol{W})$ is thus defined by setting f as the Euclidean linear layer $f(\boldsymbol{X};\boldsymbol{W}) = \boldsymbol{X}\boldsymbol{W}$. The same approach can be used for any Euclidean activation function f_{act} (*e.g.*, ReLU, Tanh, ELU, and Sigmoid), from which we obtain stereographic activation functions. Stereographic layer normalization $\mathrm{LN}_{\otimes_{\boldsymbol{\kappa}}}$ is defined in the same manner.

Stereographic logits. Suppose that $\boldsymbol{x} \in \mathfrak{st}_{\boldsymbol{\kappa}}^d$ is a stereographic embedding retrieved from the last transformer layer. For prediction tasks such as node classification, we need to compute the probability that the node with embedding $\boldsymbol{x}$ belongs to class c. Inspired by logistic regression in Euclidean spaces, Bachmann et al. (2020) proposes its stereographic variant as

$$p(y = c \mid \boldsymbol{x}) \propto \exp\left(\mathrm{sign}(\langle -\boldsymbol{p}_c \oplus_{\boldsymbol{\kappa}} \boldsymbol{x}, \boldsymbol{a}_c\rangle)\|\boldsymbol{a}_c\|_{\boldsymbol{p}_c} d_{\boldsymbol{\kappa}}(\boldsymbol{x}, H_{\boldsymbol{a}_c,\boldsymbol{p}_c})\right), \tag{2}$$

where $H_{\boldsymbol{a}_c,\boldsymbol{p}_c} = \{\boldsymbol{x} \in \mathfrak{st}_{\boldsymbol{\kappa}}^d \mid \langle -\boldsymbol{p}_c \oplus_{\boldsymbol{\kappa}} \boldsymbol{x}, \boldsymbol{a}_c\rangle = 0\}$ is a hyperplane formed by $\boldsymbol{a}_c \in \mathcal{T}_{\boldsymbol{p}_c}\mathfrak{st}_{\boldsymbol{\kappa}}^d$ and $\boldsymbol{p}_c \in \mathfrak{st}_{\boldsymbol{\kappa}}^d$. For stereographic model $\mathfrak{st}_\kappa^d$, the distance between $\boldsymbol{x} \in \mathfrak{st}_\kappa^d$ and hyperplane $H_{\boldsymbol{a},\boldsymbol{p}}$ equals

$$d_\kappa(\boldsymbol{x}, H_{\boldsymbol{a},\boldsymbol{p}}) = \sin^{-1}_{\kappa*|\kappa|}\left(\frac{2|\langle -\boldsymbol{p} \oplus_\kappa \boldsymbol{x}, \boldsymbol{a}\rangle|}{(1+\kappa\|\langle -\boldsymbol{p} \oplus_\kappa \boldsymbol{x}, \boldsymbol{a}\rangle\|^2)\|\boldsymbol{a}\|}\right). \tag{3}$$

This distance function can be easily extended to the product-stereographic model as mentioned in Section 3.1, and parameters $\boldsymbol{a}, \boldsymbol{p}$ that define the hyperplane are learnable during training.

4.2 Stereographic Multi-Head Attention

Using the stereographic operations above, we propose a multi-head attention mechanism under product-stereographic models. The key intuition is that each h-th attention head operates on the κ_h-stereographic space. Given a sequence of n product-stereographic embeddings $\boldsymbol{X} \in \mathfrak{st}_\kappa^{n\times d}$, the attention head with curvature κ first obtains values using the stereographic linear layer. For queries and keys, it maps each stereographic embedding to the tangent space of the values as:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q \in \mathcal{T}_{\boldsymbol{V}}\mathfrak{st}_\kappa^{n\times d'}, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K \in \mathcal{T}_{\boldsymbol{V}}\mathfrak{st}_\kappa^{n\times d'}, \quad \boldsymbol{V} = \mathrm{Linear}_\kappa(\boldsymbol{X};\boldsymbol{W}^V) \in \mathfrak{st}_\kappa^{n\times d'}, \tag{4}$$

where $\boldsymbol{W}^Q, \boldsymbol{W}^K \in \mathbb{R}^{d\times d'}$ are the query/key weight matrices, and $\boldsymbol{W}^V \in \mathbb{R}^{d\times d'}$ is the weight matrix for values. Then, the attention score between the i-th query $\boldsymbol{Q}_i$ and j-th key $\boldsymbol{K}_j$ is computed by parallel-transporting the vectors to the origin, and taking the inner product at the origin as

$$\alpha_{ij} = \langle \mathrm{PT}_{\boldsymbol{V}_i \to \mathbf{0}}(\boldsymbol{Q}_i), \mathrm{PT}_{\boldsymbol{V}_j \to \mathbf{0}}(\boldsymbol{K}_j)\rangle_{\mathbf{0}}. \tag{5}$$

Figure 2 illustrates our geometric attention mechanism. Because the metric tensor of the origin of the stereographic model is simply $4\boldsymbol{I}$ with identity matrix $\boldsymbol{I}$, the Riemannian inner product becomes equivalent to the Euclidean inner product at the origin. Finally, we aggregate values based on the attention scores using the Einstein midpoint:

$$\mathrm{Aggregate}_\kappa(\boldsymbol{V}, \boldsymbol{\alpha})_i \coloneqq \frac{1}{2} \otimes_\kappa \left(\sum_{j=1}^n \frac{\alpha_{ij}\lambda_{\boldsymbol{V}_j}^\kappa}{\sum_{k=1}^n \alpha_{ik}(\lambda_{\boldsymbol{V}_k}^\kappa - 1)} \boldsymbol{V}_j\right), \tag{6}$$

with conformal factor $\lambda_{\boldsymbol{V}_i}^\kappa$ at point $\boldsymbol{V}_i \in \mathfrak{st}_\kappa^{d'}$. By concatenating the aggregated results from each attention head, the final outcome of product-stereographic multi-head attention is

$$\mathrm{MHA}_{\otimes_{\boldsymbol{\kappa}}}(\boldsymbol{X}) = \|_{h=1}^H \mathrm{Aggregate}_{\kappa_h}(\boldsymbol{V}^h, \boldsymbol{\alpha}^h) \in \otimes_{h=1}^H \mathfrak{st}_{\kappa_h}^{n\times d}, \tag{7}$$

where κ_h denotes the curvature of the h-th attention head.

4.3 WRAP-UP

For completeness, we fill in the gap on how intermediate steps such as skip-connection are generalized towards non-zero curvatures, and how representations are processed between Transformer layers with distinct curvatures. First, recall that vanilla Transformer utilizes residual connections and Layer normalization to mitigate vanishing gradients and induce better convergence (Vaswani et al., 2017). To apply these operations on representations in the product-stereographic space, we switch to

$$\boldsymbol{X}_l = \mathrm{MHA}_{\otimes\boldsymbol{\kappa}}(\mathrm{LN}_{\otimes\boldsymbol{\kappa}}(\boldsymbol{X}_l^{\text{in}})) \oplus_\kappa \boldsymbol{X}_l^{\text{in}}, \quad \boldsymbol{X}_l^{\text{out}} = \mathrm{FFN}_{\otimes\boldsymbol{\kappa}}(\mathrm{LN}_{\otimes\boldsymbol{\kappa}}(\boldsymbol{X}_l)) \oplus_\kappa \boldsymbol{X}_l. \tag{8}$$

Note that while each attention head in stereographic multi-head attention operates on each stereographic model independently, the product-stereographic feed-forward network $\mathrm{FFN}_{\otimes\boldsymbol{\kappa}}$, for which we use two stereograhpic linear layers with an activation in between, fuses representations from distinct geometries and performs interactions between different steregraphic models similarly to previous work (Zhu et al., 2020b; Deng et al., 2023).

Furthermore, note that each l-th Transformer layer operates on a distinct product-stereographic space $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^l}$ where $\boldsymbol{\kappa}^l = (\kappa_1^l, \ldots, \kappa_H^l)$ together forms the geometric signature of the layer. For consistency, we assume that the input embeddings are on the product-stereographic model of the first layer (*i.e.*, $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^1}$). In case of classification tasks where logits are computed, the product-stereographic logit layer operates on the last set of curvatures (*i.e.*, $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^L}$ where L denotes the number of Transformer layers). In between layers, representations are translated from $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^l}$ to $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^{l+1}}$ by assuming a shared tangent space at the origin (*i.e.*, $\boldsymbol{X}_{l+1}^{\text{in}} = (\exp_{\mathbf{0}}^{\otimes\boldsymbol{\kappa}_{l+1}} \circ \log_{\mathbf{0}}^{\otimes\boldsymbol{\kappa}_l})(\boldsymbol{X}_l^{\text{out}})$). Altogether, it is straightforward to find that **FPS-T becomes equivalent to the original Transformer as all κ approaches 0**, yet it possesses the capability to deviate itself away from Euclidean geometry given it leads to better optimization. For all experiments, we initialize all curvatures as zero to demonstrate the practicality of our method by not requiring additional hyperparameter tuning over different curvature combinations.

4.4 EXTENSION TO GRAPH TRANSFORMER

To learn graph-structured data with FPS-T, we borrow the tokenization technique used by TOKENGT (Kim et al., 2022). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with N nodes in node-set $\mathcal{V}$, M edges in edge-set $\mathcal{E}$, and respective features $\boldsymbol{X}^{\mathcal{V}} \in \mathbb{R}^{N\times d}$, $\boldsymbol{X}^{\mathcal{E}} \in \mathbb{R}^{M\times d}$. Then, we tokenize $\mathcal{G}$ into a sequence $\boldsymbol{X} = [\boldsymbol{X}^{\mathcal{V}}, \boldsymbol{X}^{\mathcal{E}}] \in \mathbb{R}^{(N+M)\times d}$ by treating each node and edge as an independent token, and augment the tokens with 1) node identifiers that serve as positional encoding and 2) type identifiers that allow the model to distinguish between node- and edge-tokens. TOKENGT feeds this sequence into vanilla Transformer, an approach proven to pass the 2-dimensional Weisfeiler-Lehman (2-WL) graph isomorphism test and surpass the theoretical expressivity of message-passing GNNs (Kim et al., 2022; Maron et al., 2019). More details on the tokenization procedure can be found in Appendix B.

In our work, we encode the input sequence through FPS-T instead, such that nodes and edges exchange information globally on the product-stereographic space. As augmented feature vectors $\boldsymbol{X}$ are initially Euclidean, we assume each token lies within the tangent space at the origin of the product-stereographic model of the first layer $\mathcal{T}_{\mathbf{0}}\mathfrak{st}^{d'}_{\otimes\boldsymbol{\kappa}^1} \cong \mathbb{R}^{H\times d'}$, where $|\boldsymbol{\kappa}^1| = H$ and $Hd' = d$. Therefore, we apply exponential mapping on the tokens to place them on the product-stereographic model via $\exp_{\mathbf{0}}^{\otimes\boldsymbol{\kappa}^1}(\boldsymbol{X})$, the output of which is forwarded through FPS-T.

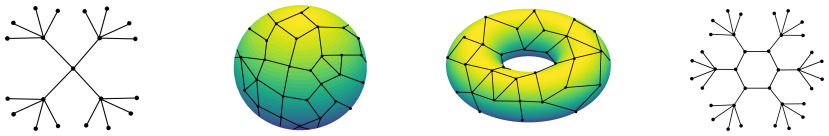
4.5 COST LINEARIZATION OF STEREOGRAPHIC ATTENTION

One drawback of the graph tokenization method above is that its computational cost becomes intractable when encoding large graphs. As computing the attention score matrix takes time and memory quadratic to the sequence length, a graph with N nodes and M edges incurs an asymptotic cost of $\mathcal{O}((N+M)^2)$, which can be $\mathcal{O}(N^4)$ for dense graphs. Fortunately, there exist various advancements used to make Transformers more efficient (Tay et al., 2022; Kitaev et al., 2020; Choromanski et al., 2020; Wang et al., 2020; Xiong et al., 2021; Cho et al., 2022).

In linearized attention (Katharopoulos et al., 2020), it is shown that the Euclidean attention score $\langle \boldsymbol{Q}_i, \boldsymbol{K}_j \rangle$ can be approximated with the product of kernel function $\phi(\boldsymbol{Q}_i)\phi(\boldsymbol{K}_j)^T$, where $\phi(\boldsymbol{X}) = \mathrm{ELU}(\boldsymbol{X}) + 1$. For stereographic attention (Equation 5), computing dot-products on the tangent space of the origin allows us to extend this kernelization to FPS-T. Let $\tilde{\boldsymbol{Q}}_i = \mathrm{PT}_{\boldsymbol{V}_i\to\mathbf{0}}(\boldsymbol{Q}_i)$ and $\tilde{\boldsymbol{K}}_j = \mathrm{PT}_{\boldsymbol{V}_j\to\mathbf{0}}(\boldsymbol{K}_j)$ be the tangent vectors on the origin prior to taking the dot-product. By applying

Table 1: Synthetic graph reconstruction results in average distortion (lower is better). The best FPS-T configuration and its learned curvatures are well-aligned to the geometry of the input graph.

Model	Space	TREE	SPHERE	TORUS	RING OF TREES
TOKENGT	$\mathbb{E}^{10}$	0.04363	0.04023	0.07172	0.05553
	$\mathbb{E}^5 \times \mathbb{E}^5$	0.04357	0.04139	0.07167	0.05546
FPS-T (ours)	$\mathfrak{st}^{10}_{\kappa_1}$	**0.00072**	**0.02176**	0.06415	0.03393
	$\mathfrak{st}^5_{\kappa_1} \times \mathfrak{st}^5_{\kappa_2}$	0.00105	0.02206	**0.06135**	**0.01630**
Best FPS-T curvatures		(-1.219)	$(+0.0629)$	$(+1.308, +0.2153)$	$(+0.3241, -3.314)$



(a) TREE (b) SPHERE (c) TORUS (d) RING OF TREES

Figure 3: Illustration of geometric graphs used in our synthetic graph reconstruction experiment.

kernelization to stereographic attention, we can rewrite the stereographic aggregation (Equation 6) as

$$\frac{1}{2} \otimes_\kappa \left(\sum_{j=1}^{n} \frac{\langle \tilde{\boldsymbol{Q}}_i, \tilde{\boldsymbol{K}}_j \rangle_{\mathbf{0}} \lambda^\kappa_{\boldsymbol{V}_j}}{\sum_{k=1}^{n} \langle \tilde{\boldsymbol{Q}}_i, \tilde{\boldsymbol{K}}_k \rangle_{\mathbf{0}} (\lambda^\kappa_{\boldsymbol{V}_k} - 1)} \boldsymbol{V}_j \right) \approx \frac{1}{2} \otimes_\kappa \left[\phi(\tilde{\boldsymbol{Q}}) \left(\phi'(\tilde{\boldsymbol{K}})^T \tilde{\boldsymbol{V}} \right) \right]_i \tag{9}$$

where $\phi'(\boldsymbol{K})_i = \phi(\boldsymbol{K})_i (\lambda^\kappa_{\boldsymbol{V}_i} - 1)$ and $\tilde{\boldsymbol{V}}_i = \frac{\lambda^\kappa_{\boldsymbol{V}_i}}{\lambda^\kappa_{\boldsymbol{V}_i} - 1} \boldsymbol{V}_i$.

This approximation enables FPS-T to encode graphs with $\mathcal{O}(N + M)$ cost, matching the complexity of message-passing GCNs (Wu et al., 2020) while taking the non-Euclidean geometry into account. In Appendix C, we empirically verify this asymptotic cost and also find that the additional cost of Riemannian operations in FPS-T are mostly dominated by pre-existing Transformer operations when encoding large networks. In the upcoming experiments, we use the kernelized approach for FPS-T and find that the approximation performs well in practice.

5 EXPERIMENTS

We first evaluate FPS-T on synthetic geometric graph reconstruction (e.g. tree or spherical graph) to verify whether our approach learns curvatures that best fit the input graph. We also benchmark existing graph reconstruction and node classification datasets to empirically demonstrate the benefit of capturing long-range interactions under mixed-curvature spaces in real-world settings.

5.1 GRAPH RECONSTRUCTION

Datasets. For synthetic graph reconstruction, we generate four types of graphs where the suitable geometry is known a priori — TREES ($\mathbb{H}$), SPHERE ($\mathbb{S}$), TORUS ($\mathbb{S} \times \mathbb{S}$), and RING OF TREES ($\mathbb{S} \times \mathbb{H}$). An example illustration of the synthetic graphs can be found in Figure 3. We then evaluate FPS-T on four real-world networks: WEB-EDU (Gleich et al., 2004) is a web-page network under the *.edu* domain connected with hyperlinks; POWER (Watts & Strogatz, 1998) is a network that models the electrical power grid in western US; BIO-WORM (Cho et al., 2014) is a genetics network of the *C. elegans* worm; FACEBOOK (Leskovec & Mcauley, 2012) is a social network. Further details on the datasets such as sectional curvature statistics of the networks can be found in Appendix D.

Training. The goal of graph reconstruction is to learn continuous node representations of the given graph that preserve the edge connectivity structure through distances in the feature space. Let $\boldsymbol{h}_u$ denote the encoded representation of node $u \in \mathcal{V}$ given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For synthetic graph reconstruction, we train FPS-T and TOKENGT by minimizing the graph distortion (Gu et al., 2019):

$$\mathcal{L}(\boldsymbol{h}, \mathcal{G}) = \sum_{\substack{(u,v) \in \mathcal{V} \times \mathcal{V} \\ u \neq v}} \left| \left(\frac{d(\boldsymbol{h}_u, \boldsymbol{h}_v)}{d_{\mathcal{G}}(u, v)} \right)^2 - 1 \right|$$

where $d(\boldsymbol{h}_u, \boldsymbol{h}_v)$ denote the distance between $\boldsymbol{h}_u$ and $\boldsymbol{h}_v$ on the representation space, and $d_{\mathcal{G}}(u, v)$ equals the shortest path distance between nodes u and v on graph $\mathcal{G}$. Both methods use a single layer with 1 or 2 attention heads with a combined latent dimension of 10.

Dataset Avg. Curvature	WEB-EDU -0.63	POWER -0.28	FACEBOOK -0.08	BIO-WORM -0.03
MLP	83.24±1.32	83.89±4.02	50.64±15.12	73.34±20.85
GCN	79.95±0.23	98.25±0.02	78.99±0.29	93.32±1.06
GAT	88.86±0.36	99.03±0.01	82.81±0.25	97.76±0.03
SAGE	86.34±0.31	97.58±0.14	81.01±0.26	96.86±0.06
SGC	78.78±0.12	97.69±0.05	74.69±0.36	89.73±0.59
TOKENGT	89.45±0.06	99.10±0.00	84.71±0.02	97.82±0.02
HGCN	80.13±0.31	96.82±0.08	74.35±5.39	86.96±0.30
HGNN	83.64±0.26	97.85±0.05	78.74±0.58	90.97±1.06
HAT	90.21±0.36	93.86±0.34	80.09±0.20	93.58±0.42
κ-GCN	55.34±35.88	98.23±0.09	20.80±20.69	84.16±13.67
$\mathcal{Q}$-GCN	80.34±0.07	97.87±0.01	76.33±0.01	96.15±0.01
FPS-T	**99.10**±0.01	**99.32**±0.01	**86.16**±0.10	**98.19**±0.03

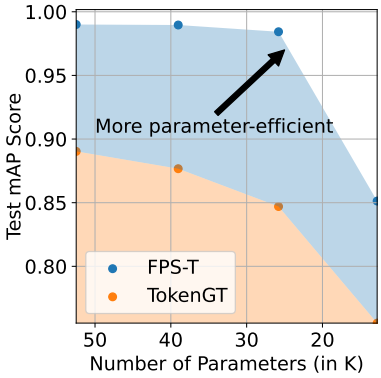


Figure 4: **Left:** Real-world graph reconstruction results. We run each method under 5 random seeds and report the average mAP with 95% confidence intervals. **Right:** Test mAP (Y-axis) of FPS-T and TOKENGT on WEB-EDU with decreasing model size (X-axis; by decreasing the latent dimension). Using mixed-curvarture spaces can be more parameter efficient in preserving graph structures.

For real-world graph reconstruction, we instead minimize a loss function that aims for preserving the local connections as computing the all-pairwise shortest path distances becomes computationally intractable with large networks:

$$\mathcal{L}(\boldsymbol{h}, \mathcal{G}) = \sum_{(u,v)\in\mathcal{E}} \log \frac{e^{-d(\boldsymbol{h}_u, \boldsymbol{h}_v)}}{\sum_{v'\in\bar{\mathcal{E}}(u)} e^{-d(\boldsymbol{h}_u, \boldsymbol{h}_{v'})}}$$

Here, $\bar{\mathcal{E}}(u)$ denotes the set of non-neighbors of node u. In addition to TOKENGT, we also compare FPS-T against baselines including Euclidean (GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), SAGE (Hamilton et al., 2017), SGC (Wu et al., 2019)), hyperbolic (HGCN (Chami et al., 2019), HGNN (Liu et al., 2019), HAT (Zhang et al., 2021)), and mixed-curvature (κ-GCN (Bachmann et al., 2020), $\mathcal{Q}$-GCN (Xiong et al., 2022)) message passing-based GCNs. For fair comparison, we set the number of layers to one and latent dimension to 16 for all models. We train all models for 10k epochs using an Adam optimizer with learning rate $1e{-}2$. The node features are given as one-hot encodings with additional random noise following Xiong et al. (2022). We defer details on the choice of hyperparameters of baseline methods to Appendix E.

Results. Table 1 reports the synthetic graph reconstruction results in average graph distortion as well as curvatures learned by FPS-T. As expected, FPS-T consistently outperforms its Euclidean counterpart on all four networks due to the networks exhibiting highly non-Euclidean structures. Despite being initialized at zero, the learnable curvatures in FPS-T converge towards curvatures that intuitively match with the input graph: for RING OF TREES, FPS-T with two attention heads converge towards one positive and one negative curvature, outperforming the single-head variant.

Next, the left table in Figure 4 shows the average sectional curvature of each real-world network and corresponding graph reconstruction results in mean average-precision (mAP) which measures the average ratio of nearest points that are actual neighbors of each node. We find that FPS-T shows significant performance gains on all four networks when compared to all baselines including Euclidean TOKENGT. Specifically, FPS-T shows a 10.5% gain in mAP against TOKENGT on WEB-EDU with an average sectional curvature of -0.63, showing that performing attention on the non-Euclidean product-stereographic space is especially effective when encoding graphs containing of many non-zero sectional curvatures.

Note that non-Euclidean spaces are theoretically known to well-embed complex structures in low dimensions, while Euclidean spaces require a large number of dimensions to attain reasonable precision (Sala et al., 2018). Based on this observation, we test whether FPS-T enjoys better parameter efficiency compared to TOKENGT by training two models with decreasing latent dimensions in $\{16, 12, 8, 4\}$. In the right plot of Figure 4, we report the mAP score of TOKENGT and FPS-T on the WEB-EDU network after training with decreasing number of parameters. We observe that our approach of incorporating mixed-curvature spaces consistently obtains low distortion embeddings in a more parameter-efficient manner, outperforming TOKENGT with $d = 16$ using half its model size.

Table 2: Node classification results. We run each method under 10 different random seeds and report the average F1 scores with 95% confidence intervals and average rankings across all datasets.

Dataset $\mathcal{H}(\mathcal{G})$	TEXAS 0.11	CORNELL 0.13	WISCONSIN 0.20	ACTOR 0.22	AIRPORT 0.72	CITESEER 0.74	PUBMED 0.80	CORA 0.81	Avg. Rank
MLP	70.54±3.00	58.38±4.04	81.20±1.87	33.62±0.55	54.05±1.78	52.58±1.97	67.17±0.91	52.44±1.08	8.25
GCN	57.84±1.62	47.84±1.77	45.40±2.62	27.09±0.36	92.00±0.63	71.38±0.43	78.37±0.26	80.40±0.53	7.38
GAT	59.46±1.12	55.14±1.80	46.20±2.30	27.43±0.23	92.35±0.36	71.70±0.28	78.14±0.31	82.29±0.46	6.13
SAGE	68.38±3.54	70.54±2.01	78.40±0.52	36.87±0.50	93.21±0.57	70.58±0.42	77.31±0.59	78.88±0.87	5.13
SGC	57.57±2.96	52.97±2.87	46.40±2.01	27.14±0.46	90.48±1.01	**72.11±0.38**	75.11±1.27	79.68±0.65	8.25
TOKENGT	88.65±2.06	71.62±2.13	83.00±0.65	36.59±0.89	95.90±0.59	71.23±0.51	**78.93±0.27**	81.42±0.79	2.50
HGCN	54.59±3.93	55.68±1.80	55.60±2.53	28.89±0.16	92.47±0.63	69.92±0.61	75.67±0.99	80.00±0.85	7.00
HGNN	50.81±3.60	52.70±1.42	54.60±2.68	29.09±0.19	90.55±0.71	69.82±0.53	76.72±0.86	79.30±0.51	8.75
HAT	82.16±2.52	70.54±1.67	81.80±1.36	38.34±0.26	92.88±0.57	68.14±0.53	77.50±0.42	79.81±0.58	4.38
κ-GCN	56.22±4.38	55.68±5.59	46.60±2.41	26.39±0.60	82.58±3.70	54.06±4.45	68.61±3.05	73.70±0.69	10.3
$\mathcal{Q}$-GCN	51.35±3.44	55.95±2.85	52.80±2.20	28.18±0.55	91.39±1.05	66.15±0.45	77.13±0.59	79.63±0.57	8.25
FPS-T	**89.19±2.37**	**72.16±2.96**	**83.60±1.14**	**39.61±0.54**	**96.01±0.55**	70.03±0.71	78.52±0.58	**82.32±0.70**	**1.75**

5.2 NODE CLASSIFICATION

Datasets. For node classification we experiment on eight different networks: three WebKB networks (TEXAS, CORNELL, WISCONSIN) that connect web-pages via hyperlinks (Craven et al., 1998), a co-occurrence network from Wikipedia pages related to English films (ACTOR) (Tang et al., 2009), three citation networks (CITESEER, PUBMED, CORA) (Sen et al., 2008), and an airline network (AIRPORT) (Chami et al., 2019). These networks are chosen to test our approach under a wide spectrum of graph homophily $\mathcal{H}(\mathcal{G})$, which measures the ratio of edges that connect nodes that share the same label (Zhu et al., 2020a). In other words, a hetereophilic graph with small graph homophily requires capturing long-range interactions for proper labeling, which is naturally difficult for message passing-based approaches with small receptive fields. More detailed statistics on the networks can be found in Appendix D.

Training. For all methods, we fix the embedding dimension to 16 and train each model to minimize the cross-entropy loss using an Adam optimizer with a learning rate of $1e{-}2$. For models that use learnable curvatures (*i.e.*, HGCN, κ-GCN and FPS-T), we use a learning rate of $1e{-}4$ for the curvatures. The optimal number of layers, activation function, dropout rate, and weight decay of each method are chosen via grid search on each dataset. Details on the hyperparameter search-space and dataset splits can be found in Appendix E.2.

Results. Table 2 shows the results from node classification. Overall, our method attains best accuracy on 6 out of 8 datasets, showing that FPS-T is effective across networks with various graph homophily. In case of hetereophilic networks, we find that the small receptive fields of message-passing GCNs are extremely inadequate, often being outperformed by a simple MLP that completely ignores the graph connectivity. On the other hand, FPS-T consistently outperforms MLP as well as GCN baselines, due to its ability to exchange information across long distances via global-attention. It also significantly outperforms TOKENGT by 8.3% on Actor, showing that adjusting the geometry towards non-Euclidean can further enhance predictive performance. In homophilic networks where message-passing is more well-suited, FPS-T shows competitive performance against GCN baselines. This is expected as FPS-T enjoys the same capacity as TOKENGT to mimic any order-2 equivariant bases (Kim et al., 2022), which includes local message-passing, through attention score computation.

6 CONCLUSION

We propose FPS-T, a natural generalization of the Transformer architecture towards mixed-curvature spaces with learnable curvatures. When combined with the graph tokenization technique of Kim et al. (2022), our model can embed graphs with less distortion and higher parameter-efficiency than its Euclidean counterpart by operating on the product-stereographic model. We also show that our model outperforms existing hyperbolic and mixed-curvature message-passing GCN baselines on node classification via global-attention that can capture long-range interactions. By linearizing the cost of self-attention through kernelized approximation, FPS-T runs in cost linear to the number of nodes and edges, allowing practical use on large-scale networks. For future work, we plan to extend towards heterogeneous manifolds (Giovanni et al., 2022) with input-dependent sectional curvatures and also optimize Riemannian operations towards better stability and efficiency under finite precision. As we propose a foundational generalization of the Transformer architecture, investigating what geometry suits best for various tasks in the NLP and CV domain would also be an interesting direction.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Gregor Bachmann, Gary Bécigneul, and Octavian Ganea. Constant curvature graph convolutional networks. In *International Conference on Machine Learning*, pp. 486–496. PMLR, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

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

Ara Cho, Junha Shin, Sohyun Hwang, Chanyoung Kim, Hongseok Shim, Hyojin Kim, Hanhae Kim, and Insuk Lee. WormNet v3: a network-assisted hypothesis-generating server for Caenorhabditis elegans. *Nucleic Acids Research*, 42(W1):W76–W82, 05 2014. ISSN 0305-1048. doi: 10.1093/nar/gku367. URL https://doi.org/10.1093/nar/gku367.

Sungjun Cho, Seonwoo Min, Jinwoo Kim, Moontae Lee, Honglak Lee, and Seunghoon Hong. Transformers meet stochastic block models: Attention with data-adaptive sparsity and cost. *arXiv preprint arXiv:2210.15541*, 2022.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

Calin Cruceru, Gary Bécigneul, and Octavian-Eugen Ganea. Computationally tractable riemannian manifolds for graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7133–7141, 2021.

Michaël Defferrard, Martino Milani, Frédérick Gusset, and Nathanaël Perraudin. Deepsphere: a graph-based spherical cnn. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1e3OlStPB.

Cheng Deng, Fan Xu, Jiaxing Ding, Luoyi Fu, Weinan Zhang, and Xinbing Wang. Fmgnn: Fused manifold graph neural network. *arXiv preprint arXiv:2304.01081*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are k-hop message passing graph neural networks. *arXiv preprint arXiv:2205.13328*, 2022.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pp. 1646–1655. PMLR, 2018.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Francesco Di Giovanni, Giulia Luise, and Michael M. Bronstein. Heterogeneous manifolds for curvature-aware graph embedding. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=rtUxsN-kaxc.

David Gleich, Leonid Zhukov, and Pavel Berkhin. Fast parallel pagerank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038, available via http://research. yahoo. com/publication/YRL-2004-038. pdf*, 13:22, 2004.

Daniele Grattarola, Daniele Zambon, Lorenzo Livi, and Cesare Alippi. Change detection in graph streams by learning graph embeddings on constant-curvature manifolds. *IEEE Transactions on neural networks and learning systems*, 31(6):1856–1869, 2019.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJxeWnCcF7.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Marc Law and Jos Stam. Ultrahyperbolic representation learning. *Advances in neural information processing systems*, 33:1668–1678, 2020.

Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 44(12):10270–10276, 2021.

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

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/59dfa2df42d9e3d41f5b02bfc32229dd-Paper.pdf.

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

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4460–4469. PMLR, 10–15 Jul 2018.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Richard C. Wilson, Edwin R. Hancock, Elżbieta Pekalska, and Robert P.W. Duin. Spherical and hyperbolic embeddings of data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2255–2269, 2014. doi: 10.1109/TPAMI.2014.2316836.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

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

Bo Xiong, Shichao Zhu, Nico Potyka, Shirui Pan, Chuan Zhou, and Steffen Staab. Pseudo-riemannian graph convolutional networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=KeIuNChob1H.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138–14148, 2021.

Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. Hrcf: Enhancing collaborative filtering via hyperbolic geometric regularization. In *Proceedings of the ACM Web Conference 2022*, pp. 2462–2471, 2022.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Ye. Hyperbolic graph attention network. *IEEE Transactions on Big Data*, 8(6):1690–1701, 2021.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020a.

Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. Graph geometry interaction learning. *Advances in Neural Information Processing Systems*, 33:7548–7558, 2020b.

ANSWERS TO POTENTIAL QUESTIONS

To provide a better understanding of our draft, we start the supplementary material by providing answers to potential questions on the overall motivation of our work, our proposed methodology, and presented empirical results. We hope most questions during review can be answered in this section, and would be happy to clarify any further questions during the author response period as well. Further supplementary material can be found in the following sections.

Q1: Why should we consider mixed-curvature spaces for graph representation learning?
A1: Previous works have shown that graphs with both hyperbolic (*e.g.* hierarchical trees) and spherical (*e.g.* cycles) structures require both curvature spaces to be embedded accurately (Gu et al., 2019; Bachmann et al., 2020). Based on our sectional curvature estimations in Figure 9, we find that this is often the case in real-world networks, showing sectional curvatures with both negative and positive signs. As the stereographic model can universally model both spherical and hyperbolic spaces, we consider it to be a good fit for learning such networks. The benefit of mixed-curvature spaces is also evident in our results on graph reconstruction, as FPS-T outperforms hyperbolic baselines (*i.e.* HGCN (Chami et al., 2019), HGNN (Liu et al., 2019), HAT (Zhang et al., 2021)).

Q2: What advantages does the product-steregraphic model have over the pseudo-Riemannian manifold used in $\mathcal{Q}$-GCN (Xiong et al., 2022)?
A2: While the pseudo-Riemannian manifold with an indefinite metric can model both spherical and hyperbolic spaces by containing both as submanifolds, the $\mathcal{Q}$-GCN architecture requires setting the time-dimension of the manifold as hyperparameter (Xiong et al., 2022) (a pseudo-hyperboloid with larger time-dimension makes it more similar to a spherical manifold). According to the results in (Xiong et al., 2022), we find that the downstream performance is sensitive to the time-dimension, which implies extensive hyperparameter tuning for quality predictions. To make things worse, the number of possible time-dimensions increases linearly to the embedding dimension of choice, which can make the tuning process intractable when scaling up towards large dimensions.

On the other hand, the product-stereographic model used in FPS-T does not require any such hyperparameter tuning. While the initial curvatures may be of concern during model initialization, we find that FPS-T performs well under initially setting all curvatures to zero, thereby starting from a Euclidean space and gradually tuning the curvatures to fit the input graph.

Q3: How is FPS-T different from Hyperbolic Attention Network (Gulcehre et al., 2018)?
A3: Hyperbolic Attention Network (HAtt) (Gulcehre et al., 2018) performs global-attention by computing the attention scores based on the hyperbolic distances between query- and key-vectors. Our FPS-T instead extends the dot-product attention from vanilla Transformer (Vaswani et al., 2017), thereby inheriting the same theoretical expressiveness on universal approximability (Yun et al., 2019) through a natural generalization to mixed-curvature spaces. Furthermore, our method can tune the curvatures based on the data and task and can represent both positive and negative curvatures, whereas HAtt uses the hyperboloid and the Klein model and thus its representation space is limited within the hyperbolic domain.

Q4: How is the running cost FPS-T compared vs. TokenGT (Kim et al., 2022) and other baselines (Xiong et al., 2022; Bachmann et al., 2020)?
A4: For comparing the computational cost of FPS-T against our baselines, we measure the runtime and peak memory use during inference on the four networks used in our graph reconstruction datasets. Note the network statistics are presented in Table 4.

Figure 5 shows the computational cost measurements. Compared to TokenGT, FPS-T essentially does not use more memory. Furthermore, both time and memory cost of FPS-T are far below those of $\mathcal{Q}$-GCN, which demonstrates better utility of closed-form operations on the κ-stereographic model compared to those on the pseudo-Riemannian manifold.

Q5: What are the features used in the graph reconstruction and node classification experiments?
A5: For graph reconstruction, we use a one-hot encoding as node representations, and thus each node embedding is learned independently. Specifically for TokenGT and FPS-T where the sequence includes edge tokens as well, we do not provide any edge features, and thus the edge tokens only contain the positional encoding of the graph and its type identifier (more details can be found in Appendix B).

Dataset	Web-Edu	Power	Facebook	Bio-Worm
GCN	0.71	2.54	2.71	1.45
GAT	7.90	7.35	11.24	7.86
SAGE	10.90	9.13	22.20	7.66
SGC	1.11	2.35	3.12	2.02
TokenGT	10.10	8.93	57.09	50.28
HGCN	8.55	10.64	9.50	9.06
HGNN	5.90	6.51	9.57	9.59
HAT	17.50	20.11	20.11	16.15
κ-GCN	7.91	10.68	8.84	8.03
$\mathcal{Q}$-GCN	72.09	75.36	74.62	72.42
FPS-T	22.72	21.64	70.94	66.44

Dataset	Web-Edu	Power	Facebook	Bio-Worm
GCN	37.72	96.32	74.63	30.26
GAT	37.99	96.49	80.46	35.52
SAGE	145.05	377.39	257.01	86.96
SGC	37.89	96.60	74.86	30.39
TokenGT	73.64	190.38	437.80	352.60
HGCN	144.51	376.67	252.64	83.18
HGNN	144.51	376.67	252.64	83.18
HAT	181.01	413.18	289.14	119.69
κ-GCN	74.80	133.70	113.11	68.32
$\mathcal{Q}$-GCN	361.03	883.21	600.42	219.70
FPS-T	73.64	190.38	444.97	358.60

Figure 5: Average runtime (left, in ms) and peak memory (right, in MB) estimation during inference. Each table shows average results over 20 different runs.

For node classification, we use the node features provided from each dataset. The WebKB networks (Cornell, Texas, Wisconsin) use bag-of-words representations of each web-page as the input node features. Actor uses a set of keywords from the Wikipedia page pertaining to each actor-node. Airport uses node features that contain geographic location of each airport as well as GDP of the country in which the airport is located, following (Chami et al., 2019). Citation networks (Citeseer, Cora, Pubmed) use bag-of-words representations of each paper. Same as in graph reconstruction, we do not endow additional link features for TokenGT and FPS-T other than the positional and token-type information.

Q6: Baselines such as GIL (Zhu et al., 2020b) and FMGNN (Deng et al., 2023) are missing in the experiments.
A6: We have tested GIL in our experiments, but found that the published code shows inconsistent performance across random seeds when compared with the original paper (Zhu et al., 2020b). We were not able to reproduce results at the time of writing, and hence leave the comparison vs. GIL as future work. For FMGNN (Deng et al., 2023), we have not yet found published official code possibly due to the work being published recently, and thus were not able to include the method as our baseline. Nonetheless, we believe that these methods are still limited by their small receptive fields as they lie within the message-passing paradigm, and expect FPS-T to outperform on heterophilic graphs as in our node classification experiments.

Q7: κ-GCN (Bachmann et al., 2020) and $\mathcal{Q}$-GCN (Xiong et al., 2022) are outperformed by Euclidean baselines in graph reconstruction. Why is this so?
A7: We found that while the two methods are able to leverage non-Euclidean geometry, their performances are sensitive to different architectural choices. For instance, we found that κ-GCN with all curvatures initialized at zero does not perform well on real-world graphs, potentially due to issues in optimization. Similarly, $\mathcal{Q}$-GCN performs poorly when using a larger embedding dimension of 16 rather than 10 (as per their original paper), and is also very sensitive to the time-dimension parameter. On the other hand, FPS-T performs well under a simple initialization of zero curvature without any additional hyperparameter. While we could perform extensive tuning on the curvature initializations and time-dimensions for κ-GCN and $\mathcal{Q}$-GCN, respectively, we consider this to be outside the scope of our paper.

A Riemannian Operations on the Stereographic Model

In this section, we introduce closed-form equations of Riemannian operations on the stereographic model. We first define the $\tan_\kappa$ and $\sin_\kappa$ as:

$$\tan_\kappa(x) = \begin{cases} \frac{1}{\sqrt{\kappa}} \tan(\sqrt{\kappa}x), & \kappa > 0 \\ x, & \kappa = 0 \\ \frac{1}{\sqrt{-\kappa}} \tanh(\sqrt{-\kappa}x), & \kappa < 0. \end{cases}, \qquad \sin_\kappa(x) = \begin{cases} \frac{1}{\sqrt{\kappa}} \sin(\sqrt{\kappa}x), & \kappa > 0 \\ x, & \kappa = 0 \\ \frac{1}{\sqrt{-\kappa}} \sinh(\sqrt{-\kappa}x), & \kappa < 0. \end{cases}$$

Based on the mobius addition and $\tan_\kappa$, we can define the Riemannian operations of the stereographic model as shown in Table 3.

Table 3: Closed-forms of the Riemannian operations of the stereographic model. As the curvature κ converges to zero, the Riemannian operations recover the Euclidean operations.

Operations	$\kappa \in \mathbb{R}$	$\kappa \to 0$
Distance	$d_\kappa(\boldsymbol{x}, \boldsymbol{y}) = 2 \tan_\kappa^{-1}(\| -\boldsymbol{x} \oplus_\kappa \boldsymbol{y}\|)$	$2\|\boldsymbol{y} - \boldsymbol{x}\|$
Exponential map	$\exp_{\boldsymbol{x}}^\kappa(\boldsymbol{v}) = \boldsymbol{x} \oplus_\kappa \left(\tan_\kappa \left(\frac{\sqrt{\|\kappa\|}\lambda_{\boldsymbol{x}}^\kappa \|\boldsymbol{v}\|}{2} \right) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} \right)$	$\boldsymbol{x} + \boldsymbol{v}$
Log map	$\log_{\boldsymbol{x}}^\kappa(\boldsymbol{y}) = \frac{2}{\sqrt{\|\kappa\|}\lambda_{\boldsymbol{x}}^\kappa} \tan_\kappa^{-1}(\| -\boldsymbol{x} \oplus_\kappa \boldsymbol{y}\|) \frac{-\boldsymbol{x} \oplus_\kappa \boldsymbol{y}}{\| -\boldsymbol{x} \oplus_\kappa \boldsymbol{y}\|}$	$\boldsymbol{y} - \boldsymbol{x}$
Parallel transport	$\mathrm{PT}_{\boldsymbol{x} \to \boldsymbol{y}}^\kappa(\boldsymbol{v}) = (-(\boldsymbol{y} \oplus_\kappa -\boldsymbol{x}) \oplus_\kappa (\boldsymbol{y} \oplus_\kappa (-\boldsymbol{x} \oplus_\kappa \boldsymbol{v}))) \cdot \frac{\lambda_{\boldsymbol{x}}^\kappa}{\lambda_{\boldsymbol{y}}^\kappa}$	$\boldsymbol{v}$

B Details of FPS-T

B.1 Tokenization Procedure of TokenGT

In order to learn graph-structured data with FPS-T, we borrow the tokenization technique proposed by TokenGT (Kim et al., 2022). Let graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an input graph with N nodes in node-set $\mathcal{V}$, M edges in edge-set $\mathcal{E}$, and respective features $\boldsymbol{X}^{\mathcal{V}} \in \mathbb{R}^{N \times d}$, $\boldsymbol{X}^{\mathcal{E}} \in \mathbb{R}^{M \times d}$. Then, we tokenize the graph into a sequence by treating each node and edge as an independent token, and augment each token embedding with 1) node identifiers $\boldsymbol{P} \in \mathbb{R}^{N \times d}$ that serve as positional encoding and 2) type identifiers $\boldsymbol{E} \in \mathbb{R}^{2 \times d}$ that allows the model to distinguish between node- and edge-tokens.

$$\boldsymbol{X}_u \mapsto \boldsymbol{X}_u + \boldsymbol{P}_u + \boldsymbol{P}_u + \boldsymbol{E}_0 \text{ for each node } u \in \mathcal{V}$$
$$\boldsymbol{X}_{(u,v)} \mapsto \boldsymbol{X}_{(u,v)} + \boldsymbol{P}_u + \boldsymbol{P}_v + \boldsymbol{E}_1 \text{ for each edge } (u, v) \in \mathcal{E}.$$

The node identifiers in $\boldsymbol{P}$ are obtained from top-k eigenvectors of the graph Laplacian $\boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ where $\boldsymbol{A}$ and $\boldsymbol{D}$ denote the adjacency and degree matrices, respectively. The type identifiers $\boldsymbol{E}$ are set as trainable parameters.

TokenGT feeds this sequence into a pure Euclidean Transformer, an approach proven to pass the 2-dimensional Weisfeiler-Lehman (2-WL) graph isomorphism test and surpass the theoretical expressivity of message-passing GCNs (Kim et al., 2022; Maron et al., 2019). In our work, we encode the input sequence through FPS-T instead, such that nodes and edges exchange information globally on the product-stereographic space.

B.2 Overall Architecture

For further guidance, we provide a more detailed illustration of our FPS-T architecture in Figure 6.

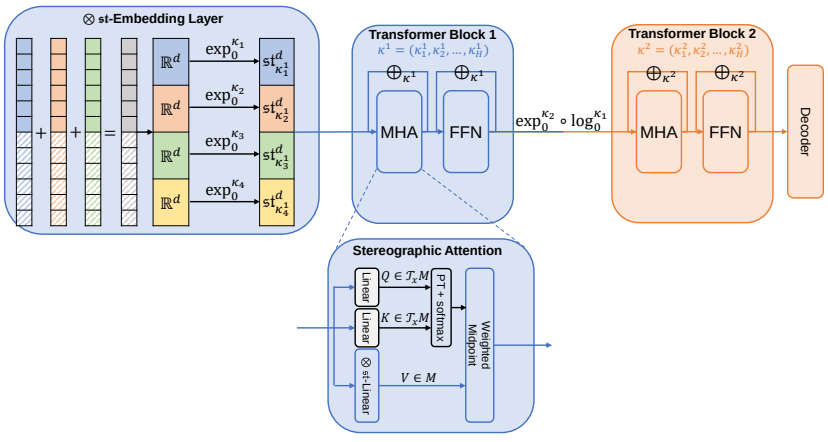


Figure 6: Illustration of the overall FPS-T architecture. Each layer of FPS-T is endowed with a product-streographic space with a curvature assigned to each attention head. The embedding layer applies exp-mapping to the token embeddings using the curvatures of the first layer. The decoder uses the curvatures of the last layer. In-between, we assume the tangent space at the origin is shared between the two product-stereographic spaces, and translate the embeddings via $\exp_{\mathbf{0}}^{\otimes \boldsymbol{\kappa}_{l+1}} \circ \log_{\mathbf{0}}^{\otimes \boldsymbol{\kappa}_l}$.

C LINEARIZED STEREOGRAPHIC ATTENTION

C.1 DETAILED DERIVATION

The equation below shows the derivation of our steregraphic attention with cost linear to the sequence length. The key step is the second step where the Euclidean inner product at the tangent space of the origin is approximated as a dot product of kernel-mappings, as used in (Katharopoulos et al., 2020).

$$\begin{aligned}
\text{LinearizedAggregate}_\kappa(\boldsymbol{V}, \boldsymbol{\alpha})_i &= \frac{1}{2} \otimes_\kappa \left(\sum_{j=1}^{n} \frac{\langle \tilde{\boldsymbol{Q}}_i, \tilde{\boldsymbol{K}}_j \rangle_{\mathbf{0}} \lambda^\kappa_{\boldsymbol{V}_j}}{\sum_{k=1}^{n} \langle \tilde{\boldsymbol{Q}}_i, \tilde{\boldsymbol{K}}_k \rangle_{\mathbf{0}} (\lambda^\kappa_{\boldsymbol{V}_k} - 1)} \boldsymbol{V}_j \right) \\
&\approx \frac{1}{2} \otimes_\kappa \left(\sum_{j=1}^{n} \frac{\phi(\tilde{\boldsymbol{Q}}_i)\phi(\tilde{\boldsymbol{K}}_j)^T (\lambda^\kappa_{\boldsymbol{V}_j} - 1)}{\sum_{k=1}^{n} \phi(\tilde{\boldsymbol{Q}}_i)\phi(\tilde{\boldsymbol{K}}_k)^T (\lambda^\kappa_{\boldsymbol{V}_k} - 1)} \cdot \frac{\lambda^\kappa_{\boldsymbol{V}_j}}{\lambda_{\boldsymbol{V}_j} - 1} \boldsymbol{V}_j \right) \\
&= \frac{1}{2} \otimes_\kappa \left[\phi(\tilde{\boldsymbol{Q}}) \left(\phi'(\tilde{\boldsymbol{K}})^T \tilde{\boldsymbol{V}} \right) \right]_i
\end{aligned}$$

C.2 RUNTIME ESTIMATION

To verify that the kernelization approach above leads to $\mathcal{O}(N + M)$ asymptotic cost for FPS-T, we measure the average runtime of forwarding networks with varying sizes. Specifically, we synthesize Erdös-Rényi graphs with number of nodes in $\{100, 500, 1000, 5000, 1000\}$ and edge-probability in $\{0.01, 0.025, 0.05\}$, and measure the average runtime over 20 forward steps for FPS-T and TOKENGT to compare the cost of two methods. Figure 7 shows the runtime of FPS-T with increasing number of nodes and edges. Here we find that the overall cost is indeed linear to the number of nodes and edges combined. The runtime ratio of FPS-T relative to the runtime of TOKENGT also approaches towards 1.06, which indicates that the additional cost of adding Riemannian operations on top of TokenGT is mostly dominated by preexisting Transformer operations such as MHA and FFN.

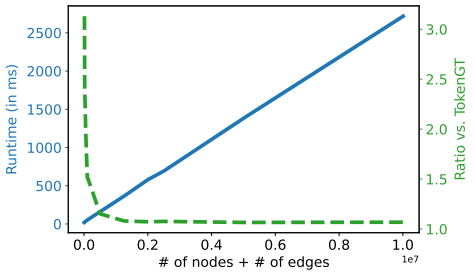


Figure 7: Runtime comparison of FPS-T vs. TOKENGT on graphs with increasing number of nodes and edges. The x-axis shows the total number of nodes and edges (in 10^7). The blue solid line (left y-axis) shows the absolute runtime of FPS-T, and the green dashed line (right y-axis) shows its ratio relative to TOKENGT.

D DATASET DETAILS

D.1 SYNTHETIC GEOMETRIC GRAPH GENERATION

For this experiment, our aim is to reconstruct four synthetic graphs, the suitable geometry for which is known a priori. For TREE, we use a uniform tree with depth 5 and branching factor 4. For SPHERE, we randomly sample 2000 nodes from a unit sphere, and connect two nodes with a edge if they are within a spherical distance of 0.3. Similarly for TORUS, we randomly sample 2000 nodes and connect two nodes if they are within a toroidal distance of 0.02. For RING OF TREES, we attach trees with 4 branches on a cycle with 10 nodes.

D.2 STATISTICS

Here we provide basic statistics on each dataset used in our experiments.

Table 4: Dataset statistics for graph reconstruction.

	Synthetic				Real-World			
Dataset	TREE	SPHERE	TORUS	RING OF TREES	WEB-EDU	POWER	FACEBOOK	BIO-WORM
# nodes	1,365	2,000	2,000	60	3,031	4,941	4,039	2,274
# edges	1,364	66,404	121,795	60	6,474	6,594	78,328	88,234

Table 5: Dataset statistics for node classification.

	Heterophilic				Homophilic			
Dataset	TEXAS	CORNELL	WISCONSIN	ACTOR	AIRPORT	CITESEER	PUBMED	CORA
$\mathcal{H}(G)$	0.11	0.13	0.20	0.22	0.72	0.74	0.80	0.81
# nodes	183	183	251	7,600	3,188	3,327	19,717	2,708
# edges	325	298	515	30,019	18,631	4,732	44,338	5,429
# features	1,703	1,703	1,703	932	4	3,703	500	1,433
# classes	5	5	5	5	4	6	3	7

E EXPERIMENTAL DETAILS

E.1 GRAPH RECONSTRUCTION

For graph reconstruction, we use a single-layer architecture with 16 (10 for FPS-T and TOKENGT in synthetic graph reconstruction) embedding dimensions for all models without hyperparameter tuning for fair comparison. We feed the entire network at each training step with no weight decay or dropout, as the objective is to simply embed nodes on the representation space such that the distances on the feature space are well-aligned to the actual graph topology.

Hyperparameters of baseline methods. For κ-GCN, we use the product of two stereographic models, both of which the curvature is initialized as zero. For $\mathcal{Q}$-GCN, we test different time dimensions in $\{1, 8, 16\}$, and report the best performance among the models.

E.2 NODE CLASSIFICATION

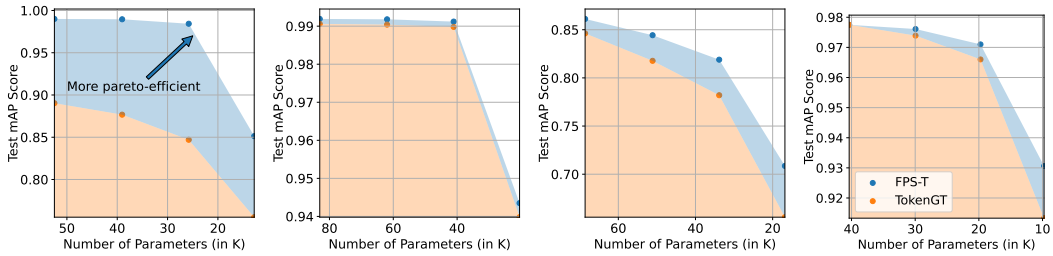
Hyperparameter search space. We share the hyperparameter search space used for node classification in Table 6. The lower two parameters are exclusive for only TOKENGT and FPS-T. For other GCN baselines, we use the same hyperparameterization as used in the graph reconstruction experiments. Note that the number of hops denotes the number of message-passing steps used to mix input node features. This is based on the previous observation that TOKENGT benefits manually injecting a sparse equivariant basis to mix node features, due to TOKENGT not being able to satisfy the orthogonality constraint when the number of Laplacian eigenvectors used for positional encoding is significantly less than the number of nodes in the graph (Kim et al., 2022).

Table 6: Hyperparameter search space used for node classification.

Parameter	Search Space
Number of layers	{1, 2, 3}
Number of heads	{1, 2, 4}
Weight decay	{0, 0.0001, 0.0005, 0.001}
Dropout	{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7}
Activation	{ReLU, ELU, Tanh, Sigmoid}
Number of hops	{0, 10, 20, 30}
Number of Laplacian eigenvectors	{4, 8, 16, 32}

Dataset splits. We use the same data splits used in previous work (Chami et al., 2019; Xiong et al., 2022). The training set for the node classification task of the 3 citation networks is consisted of 20 nodes per class and the validation/test set is created by sampling 500/1000 nodes from the remaining nodes. For the AIRPORT network, we use a 70/15/15 split for training, validating, and testing, respectively. For the remaining four heterophilic networks, we use a 60/20/20 split.

E.3 Additional Results on Parameter Efficiency



(a) Web-Edu (b) Power (c) Facebook (d) Bio-Worm

Figure 8: Model size (x-axis) vs. mAP score (y-axis) plots of FPS-T and TokenGT from each graph reconstruction task. The blue-shade indicates the pareto-efficiency gap between TokenGT and FPS-T in terms of model size. FPS-T is consistently more pareto-efficient than TokenGT across all datasets, with Web-Edu showing the largest gap due to its mixed-geometry deviating the most from Euclidean.

F Graph Sectional Curvature

F.1 Computation

Given an unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we can compute the graph sectional curvature of the node m and its two neighbors b, c as:

$$K_{\mathcal{G}}(m; b, c) = \frac{1}{|V|} \sum_{a \in \mathcal{V}} d_{\mathcal{G}}(a, m)^2 + \frac{d_{\mathcal{G}}(b, c)^2}{4} - \frac{d_{\mathcal{G}}(a, b)^2 + d_{\mathcal{G}}(a, c)^2}{2}, \tag{10}$$

where $d_{\mathcal{G}}(x, y)$ is the shortest path on the graph between node x and y. The graph sectional curvature is known to indicate specific structures such as line, trees, and cycles (Gu et al., 2019).

F.2 Sectional Curvature of the Real-World Graphs

Figure 9 shows the histograms of sectional curvatures computed by Equation 10 in every real-world graph we use in the paper.

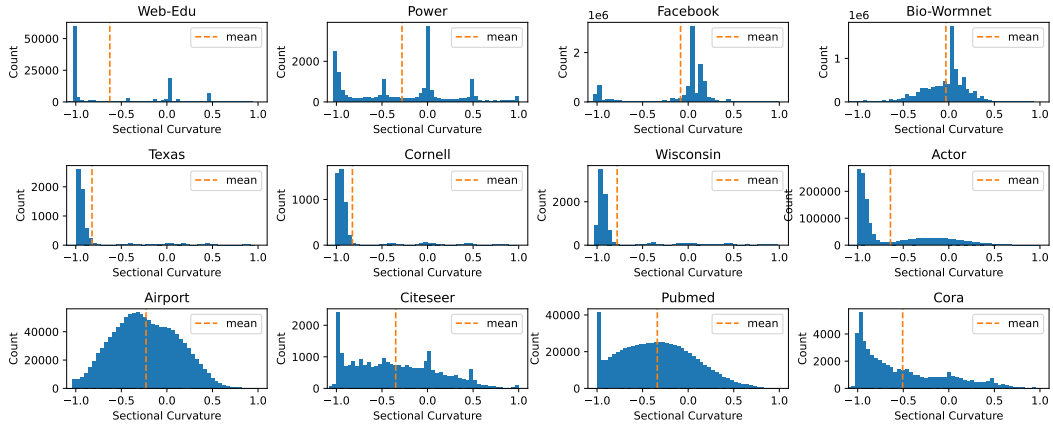


Figure 9: Sectional curvature histograms of all networks used in our experiments.

G VISUALIZATION

Figure 10 shows an example visualization of embeddings trained via graph reconstruction on the WEB-EDU network. We visualize embeddings from TOKENGT and FPS-T by running PCA on the embeddings directly (for TOKENGT), or log-mapping the embeddings to the weighted midpoint first, then applying PCA to reduce the dimension (for FPS-T).

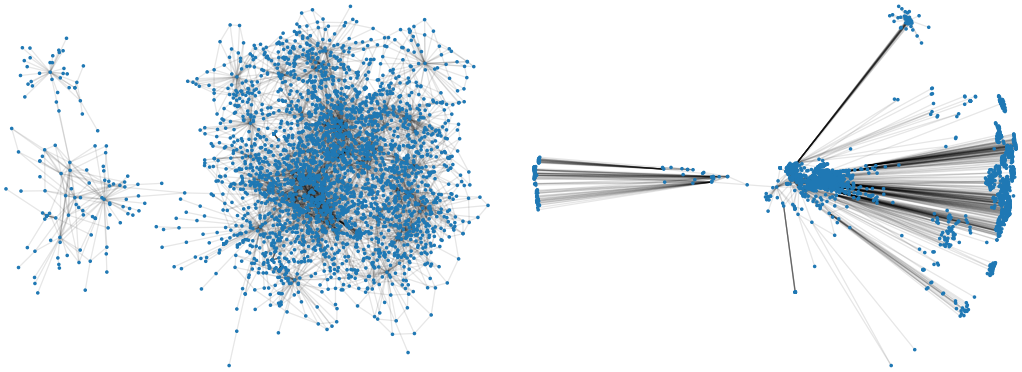


Figure 10: PCA visualizations of WEB-EDU node embeddings learned by TOKENGT (left) and FPS-T (right). Restricting the model to the Euclidean space $\mathbb{E}$ leads to convoluted structures and hence suboptimal graph distortion. FPS-T, on the other hand, adjusts itself towards the hyperbolic product-stereographic space $\mathfrak{st}^8_{-0.5} \times \mathfrak{st}^8_{-3.9}$, and captures the hierarchical structure of the graph more accurately.