Mixed-Curvature Transformers for Graph Representation Learning

Sungjun Cho [1] **Seunghyuk Cho** [1] **Sungwoo Park** [1] **Hankook Lee** [1] **Honglak Lee** [1 2] **Moontae Lee** [1 3]

Abstract

Real-world graphs naturally exhibit hierarchical or cyclical structures that are unfit for the typical Euclidean space. While there exist graph neural networks that leverage non-Euclidean spaces to embed such structures more accurately, these methods are confined under the message-passing paradigm, making the models vulnerable against side-effects such as oversmoothing. More recent work have proposed attention-based graph Transformers that can 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. Our model can learn the curvature appropriate for the input graph in an end-to-end fashion, without the need of additional tuning on different curvature initializations. We also provide a kernelized approach to non-Euclidean attention, which enables our model to run in 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 our approach.

1. Introduction

Graph-structured data often comprise of complex structures such as hierarchical trees and cycles, yet using the typical Euclidean space requires a large number of dimensions to accurately embed such structures (Sala et al., 2018). In response, the graph learning community has developed generalizations of graph convolutional neural networks (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; Bachmann et al., 2020).

[1]LG AI Research [2]University of Michigan [3]University of Illinois Chicago. Correspondence to: Moontae Lee <moontae.lee@lgresearch.ai>.

Presented at the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

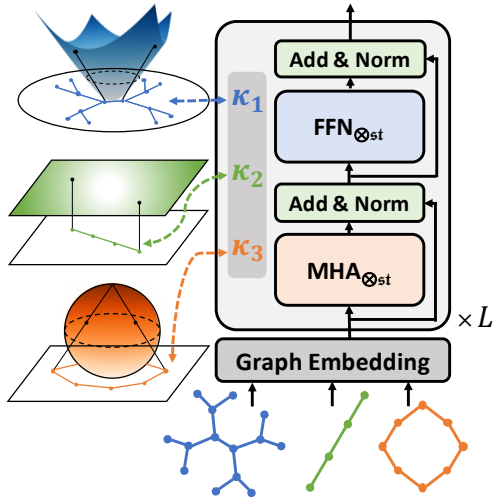

Figure 1. Illustration of our proposed FPS-T architecture. Well-known constant curvature spaces can be projected to the stereographic model, with a common chart map isomorphic to the Euclidean space. Each space can efficiently embed different types of graphs (*e.g.*, trees in hyperbolic space, lines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

However, existing non-Euclidean GCNs aggregate features via *message-passing* and are thus susceptible to oversmoothing, making it difficult to stack deep layers and learn long-range interactions in the graph (Yang et al., 2022). While there exist Transformer-based graph encoders that can easily exchange information across larger distances through global attention (Kim et al., 2022), these approaches are still confined within the Euclidean regime, and their extension towards mixed-curavture spaces has not yet been studied.

In this paper, we propose **Fully Product-Stereographic Transformer (FPS-T)**, a generalization of the Transformer architecture (Vaswani et al., 2017) towards mixed-curvature non-Euclidean geometry (see Figure 1). Specifically, we endow each attention head a stereographic model with curvature κ that can universally represent Euclidean ($\kappa = 0$), hyperbolic ($\kappa < 0$), and spherical spaces ($\kappa > 0$) (Bachmann et al., 2020). We then generalize each operation in the Transformer framework to work on the product-stereographic model. Each operation is end-to-end differentiable with respect to κ, and thus our approach can jointly learn embeddings as well as curvatures that best fit the input graph.

Related work. Hyperbolic GCNs are known to outperform Euclidean GCNs in various tasks on hierarchical graphs (Chami et al., 2019). To take advantage of both hyperbolic and spherical spaces, Bachmann et al. (2020) proposed a GCN that operates on the stereographic model. In this work, we develop a graph Transformer that can capture long-range interactions while operating on the steregraphic model with learnable curvatures.

2. Fully Product-Stereographic Transformer

Here, we describe the details of FPS-T. We first discuss the stereographic model that can universally represent spherical, Euclidean, and hyperbolic spaces. and generalize each operation in Transformer to the product-stereographic model.

2.1. Stereographic Model

The d-dimensional stereographic model $\mathfrak{st}_\kappa^d = \{\boldsymbol{x} \in \mathbb{R}^d | -\kappa\|\boldsymbol{x}\|^2 < 1\}$ is a space with constant curvature κ. Its metric tensor at point $\boldsymbol{x} \in \mathfrak{st}_\kappa^d$ is defined as $g^\kappa(\boldsymbol{x}) = \frac{4}{1+\kappa\|\boldsymbol{x}\|^2}\boldsymbol{I} =: (\lambda_{\boldsymbol{x}}^\kappa)^2\boldsymbol{I}$, where $\lambda_{\boldsymbol{x}}^\kappa$ is known as the conformal factor. Usual addition and scalar-multiplication are replaced with Möbius operations: $\boldsymbol{x} \oplus_\kappa \boldsymbol{y} = \frac{(1-2\kappa\boldsymbol{x}^T\boldsymbol{y}-\kappa\|\boldsymbol{y}\|^2)\boldsymbol{x}+(1+\kappa\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1-2\kappa\boldsymbol{x}^T\boldsymbol{y}+\kappa^2\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2}$ and $\alpha \otimes_\kappa \boldsymbol{x} = \tan_\kappa(\alpha \arctan_\kappa(\|\boldsymbol{x}\|_2))\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$ where $\boldsymbol{x}, \boldsymbol{y} \in \mathfrak{st}_\kappa^d$ and $\alpha \in \mathbb{R}$. Each point $\boldsymbol{x}$ is associated with its tangent space $\mathcal{T}_{\boldsymbol{x}}\mathfrak{st}_\kappa^d \cong \mathbb{R}^d$, the set of vectors tangent to point $\boldsymbol{x}$. The log-map $\log_{\boldsymbol{x}}^\kappa$ and exp-map $\exp_{\boldsymbol{x}}^\kappa$ are used to project points on $\mathfrak{st}_\kappa^d$ to a tangent space, and vice versa. Parallel transport $\mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}^\kappa$ moves vectors in $\mathcal{T}_{\boldsymbol{x}}\mathfrak{st}_\kappa^d$ to $\mathcal{T}_{\boldsymbol{y}}\mathfrak{st}_\kappa^d$ through a smooth connection. One attractive property of the stereographic model is that all operations are differentiable w.r.t κ at any point, including $\kappa = 0$, which allows us to learn the curvature κ autonomously based on data. For operations on the product of stereographic models $\otimes_{i=1}^n \mathfrak{st}_{\kappa_i}^d$, we substitute the symbol κ with $\otimes\boldsymbol{\kappa} := (\kappa_1, \ldots, \kappa_n)$, (*e.g.*, $\exp_{\boldsymbol{x}}^{\otimes\boldsymbol{\kappa}}$). More information can be found in (Bachmann et al., 2020).

2.2. Stereographic Neural Networks

Given a Euclidean function f, we can define its stereographic variant as $\exp_{\boldsymbol{0}}^{\otimes\boldsymbol{\kappa}}\left(f\left(\log_{\boldsymbol{0}}^{\otimes\boldsymbol{\kappa}}(\boldsymbol{X})\right)\right)$. 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 activation functions (*e.g.*, ReLU), from which we obtain stereographic activation functions $\mathrm{Act}_{\otimes\boldsymbol{\kappa}}$, and similarly for stereographic layer normalization $\mathrm{LN}_{\otimes\boldsymbol{\kappa}}$. For classification, we attach a stereographic logit layer as proposed by Bachmann et al. (2020).

2.3. Stereographic Multi-Head Attention

Using stereographic neural networks, we propose a multi-head attention mechanism for product-stereographic representations. 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

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

where $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}_V \in \mathbb{R}^{d\times d'}$ are learnable weights.

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 Riemannian inner product at the origin (which is equivalent to Euclidean dot product) as

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

Finally, we aggregate the values based on the attention scores using the Einstein midpoint as

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

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

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

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

2.4. Wrap-up

For completeness, we fill in the gap on 1) how input graph embeddings are formulated, 2) how skip-connection is implemented, and 3) how representations are transferred between Transformer layers with distinct curvatures.

We borrow the embedding technique proposed by TokenGT (Kim et al., 2022). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node features $\boldsymbol{X}^{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and edge features $\boldsymbol{X}^{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times d}$, we tokenize the graph into a sequence $\boldsymbol{X} = [\boldsymbol{X}^{\mathcal{V}}, \boldsymbol{X}^{\mathcal{E}}]$ by treating each node and edge as an independent token. To embed the graph topology, the tokens are augmented with learnable Laplacian positional encoding and type identifiers distinguishes between node- and edge-tokens. We assume each token embedding initially lies on $\mathcal{T}_{\boldsymbol{0}}\mathfrak{st}_{\otimes\boldsymbol{\kappa}^1}^{d'}$, the tangent space at the origin of the product-stereographic model of the first layer, and thus apply $\exp_{\boldsymbol{0}}^{\otimes\boldsymbol{\kappa}^1}(\boldsymbol{X})$ on the tokens to place them on the product-stereographic manifold prior to encoding through FPS-T.

Recall that vanilla Transformer utilizes skip-connections and layer normalization to mitigate vanishing gradients towards better convergence (Vaswani et al., 2017). In FPS-T, we switch to

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

Table 1. Graph reconstruction results in average mAP scores and 95% confidence intervals across 5 random seeds.

Dataset 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	89.56±0.03	99.08±0.00	84.62±0.13	97.75±0.03
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	**99.00**±0.08	**99.18**±0.06	**86.06**±0.06	**97.90**±0.16

The product-stereographic feed-forward network $\text{FFN}_{\otimes\boldsymbol{\kappa}}$, for which we use two stereographic linear layers with a stereographic activation in between, fuses representations from distinct geometries together.

Lastly, each l-th Transformer layer operates on a distinct product-stereographic space $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^l}$ where $\boldsymbol{\kappa}^l = (\kappa^l_1, \ldots, \kappa^l_H)$ together forms the geometric signature. 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.*, $\boldsymbol{X}^{\text{in}}_{l+1} = (\exp_{\mathbf{0}}^{\otimes\boldsymbol{\kappa}_{l+1}} \circ \log_{\mathbf{0}}^{\otimes\boldsymbol{\kappa}_l})(\boldsymbol{X}^{\text{out}}_l)$). In case of classification tasks where logits are computed, the stereographic logit layer operates on the stereographic space of the last layer (*i.e.*, $\mathfrak{st}^d_{\otimes\boldsymbol{\kappa}^L}$ where L denotes the number of layers).

2.5. Cost Linearization of Stereographic Attention

One drawback of the graph embedding method above is its computational cost that becomes intractable with large graphs. As computing the attention score matrix takes time and memory quadratic to the input sequence length, a graph with N nodes and M edges incurs an asymptotic cost of $\mathcal{O}((N+M)^2)$, or $\mathcal{O}(N^4)$ for dense graphs. Therefore, we linearize the attention score computation similar to the kernelization procedure introduced by Linearized Attention (Katharopoulos et al., 2020). Let $\tilde{\boldsymbol{Q}}_i = \text{PT}_{\boldsymbol{V}_i \to \mathbf{0}}(\boldsymbol{Q}_i)$ and $\tilde{\boldsymbol{K}}_j = \text{PT}_{\boldsymbol{V}_j \to \mathbf{0}}(\boldsymbol{K}_j)$ be the tangent vectors on the origin prior to taking the dot-product. Then,, we can approximate the aggregation step as: $\left(\sum_{j=1}^n \frac{\langle \tilde{\boldsymbol{Q}}_i, \tilde{\boldsymbol{K}}_j \rangle \lambda^\kappa_{\boldsymbol{V}_j}}{\sum_{k=1}^n \langle \tilde{\boldsymbol{Q}}_i, \tilde{\boldsymbol{K}}_k \rangle (\lambda^\kappa_{\boldsymbol{V}_k} - 1)} \boldsymbol{V}_j\right) \approx \left[\phi(\tilde{\boldsymbol{Q}}) \left(\phi'(\tilde{\boldsymbol{K}})^T \tilde{\boldsymbol{V}}\right)\right]_i$ 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, which matches the complexity of message-passing GCNs, while taking the non-Euclidean geometry into account. In our experiments, we use the linearized FPS-T and find that this approach performs well in practice.

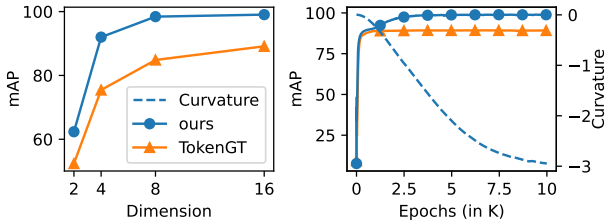

Figure 2. **Left:** Test mAP scores using smaller feature dimensions. **Right:** mAP (solid) and curvature (dashed) of FPS-T vs. TokenGT during training on Web-Edu.

3. Experiments

We empirically test FPS-T on graph reconstruction and node classification. Overall, we initialize all curvatures as zero to demonstrate its practicality by not requiring tuning over different curvature combinations. We compare against Euclidean (GCN, GAT, SAGE, SGC), hyperbolic (HGCN, HGNN, HAT), and mixed-curvature (κ-GCN, $\mathcal{Q}$-GCN) message-passing GCNs. We also test TokenGT, which is equivalent to FPS-T with fixed-zero curvatures.

3.1. Graph Reconstruction

Setting. We experiment graph reconstruction of four different real-world networks used in Xiong et al. (2022): Web-Edu, Power, Facebook, and Bio-Worm. The goal of graph reconstruction is to learn continuous node representations of the given graph that preserve the edge connectivity structure through distances among the learned representations. We train the representations of the nodes $\boldsymbol{h}$ by minimizing the loss function that aims for preserving local connectivity:

$$\mathcal{L}(\boldsymbol{h}, \mathcal{G}) = \sum_{(u,v)\in\mathcal{E}} \log \frac{e^{-d(h_u,h_v)}}{\sum_{v':(u,v')\notin\mathcal{E}} e^{-d(h_u,h_{v'})}}.$$

For fair comparison, we use a single-layer with latent dimension 16 for all models. 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. For FPS-T, we use two attention heads with all curvatures initialized as zero. Other hyperparameters such as learning rates follow Xiong et al. (2022).

Results. Table 1 shows the results in mean average-precision (mAP), the average ratio of nearest nodes on the feature space that are actual neighbors of each node. We find that FPS-T outperforms all baselines, but more importantly, FPS-T shows significant performance gains compared to Euclidean TokenGT on three networks that are largely hyperbolic. For further analysis, we train a single-head FPS-T and TokenGT on Web-Edu. The right plot of Figure 2 shows the curvature and mAP scores during training. We find that the curvature is adjusted towards the hyperbolic domain, which matches with the overall sectional curvature of Web-Edu. The mAP score of FPS-T also converges to a larger

Table 2. Node classification results in average F1 scores and 95% confidence intervals across 10 random seeds.

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
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
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
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
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
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
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
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
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
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
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
$\mathcal{Q}$-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
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**

point as the curvature deviates away from zero, indicating that the non-Euclidean feature space can contain better local optima for graph reconstruction. The left plot of Figure 2 also reveals that FPS-T preserves the reconstruction performance better as we decrease the dimension from 16, as FPS-T using only 4 dimensions (92.00 mAP with 12.7k parameters) outperforms TokenGT with $d = 16$ (89.13 mAP with 53.6k parameters). This aligns with previous work that theoretically showed non-Euclidean spaces to be more dimension-efficient in embedding complex structures compared to Euclidean spaces (Sala et al., 2018).

3.2. Node Classification

Setting. For node classification we experiment on eight networks used by Pei et al. (2020): three WebKB networks (Texas, Cornell, and Wisconsin), an Actor co-occurrence network from Wikipedia, three citation networks (Citeseer, Pubmed, Cora), and an Airport network. 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 (Pei et al., 2020).

For all methods, we use 16 feature dimensions and train each model to minimize the cross-entropy loss using an Adam optimizer with learning rate $1e{-}2$. For models that use learnable curvatures (*i.e.*, κ-GCN and FPS-T), we use 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.

Results. Table 2 shows the results from node classification. Overall, our method shows the best accuracy on 6 out of 8 datasets, showing that FPS-T is effective across networks with various graph homophily. Especially in heterophilic networks, we find that the small receptive fields of message-passing GCNs are extremely ineffective, often being outperformed by MLPs that completely ignore the graph connectivity. On the other hand, FPS-T consistently outperforms MLP as well as GCN baselines, due to its capability to capture long-range interactions.

4. Conclusion

We propose FPS-T, a generalized Transformer architecture that can operate on mixed-curvature spaces with learnable curvatures. Combined with the tokenization technique of TokenGT (Kim et al., 2022), our model can embed graph structures more accurately and parameter-efficiently than its Euclidean counterpart by operating on the product-stereographic model. We also show that our model outperforms existing non-Euclidean GCN frameworks on node classification by capturing long-range interactions. By linearizing the attention mechanism through kernelized approximation, FPS-T runs in cost linear to the number of nodes and edges, allowing practical use on large-scale networks.

References

Bachmann, G., Bécigneul, G., and Ganea, O. Constant curvature graph convolutional networks. *ICML*, 2020.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. *NeurIPS*, 2019.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: Fast autoregressive transformers with linear attention. *ICML*, 2020.

Kim, J., Nguyen, D., Min, S., Cho, S., Lee, M., Lee, H., and Hong, S. Pure transformers are powerful graph learners. *NeurIPS*, 2022.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-GCN: Geometric graph convolutional networks. *ICLR*, 2020.

Sala, F., De Sa, C., Gu, A., and Re, C. Representation tradeoffs for hyperbolic embeddings. *ICML*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *NeurIPS*, 2017.

Xiong, B., Zhu, S., Potyka, N., Pan, S., Zhou, C., and Staab, S. Pseudo-Riemannian graph convolutional networks. *NeurIPS*, 2022.

Yang, M., Zhou, M., Liu, J., Lian, D., and King, I. Hrcf: Enhancing collaborative filtering via hyperbolic geometric regularization. *WWW*, 2022.