Hyperbolic Geometric Latent Diffusion Model for Graph Generation

Xingcheng Fu [1] **Yisen Gao** [2 1] **Yuecen Wei** [3] **Qingyun Sun** [4] **Hao Peng** [4] **Jianxin Li** [4] **Xianxian Li** [1]

Abstract

Diffusion models have made significant contributions to computer vision, sparking a growing interest in the community recently regarding the application of them to graph generation. Existing discrete graph diffusion models exhibit heightened computational complexity and diminished training efficiency. A preferable and natural way is to directly diffuse the graph within the latent space. However, due to the non-Euclidean structure of graphs is not isotropic in the latent space, the existing latent diffusion models effectively make it difficult to capture and preserve the topological information of graphs. To address the above challenges, we propose a novel geometrically latent diffusion framework HypDiff. Specifically, we first establish a geometrically latent space with interpretability measures based on hyperbolic geometry, to define anisotropic latent diffusion processes for graphs. Then, we propose a geometrically latent diffusion process that is constrained by both radial and angular geometric properties, thereby ensuring the preservation of the original topological properties in the generative graphs. Extensive experimental results demonstrate the superior effectiveness of HypDiff for graph generation with various topologies.

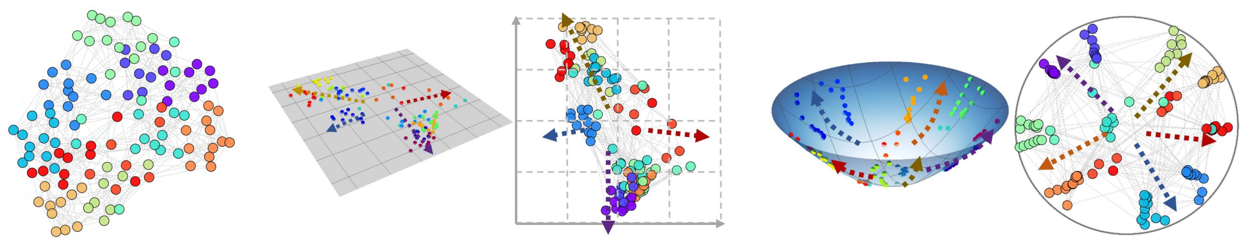
1. Introduction

Graphs in the real world contain variety and important of topologies, and these topological properties often reflect physical laws and growth patterns, such as rich-clubs, small-worlds, hierarchies, fractal structures, etc. Traditional random graph models based on graph theory, such as Erdos-Renyi (Erdős et al., 1960), Watts-Strogatz (Watts & Strogatz, 1998) and Barabasi-Albert (Barabási & Albert, 1999), etc., need artificial heuristics to build the algorithms for single nature topologies and lack the flexibility to model various complex graphs. Therefore, many deep learning models have been developed for graph generation, such as Variational Graph Auto-Encoder (VGAE) (Kipf & Welling, 2016), Graph Generative Adversarial Networks(GraphGAN) (Wang et al., 2018a), and other technologies. Recently, the Denoising Diffusion Probabilistic Model(DDPM) (Ho et al., 2020) have demonstrated great power and potential in image generation, attracting huge attention from the community of graph learning.

For graph generation, a straightforward idea involves designing discretized diffusion methods for the graph structural information. (Vignac et al., 2022; Jo et al., 2022; Luo et al., 2022), and the other way is to develop advanced graph encoders to preserve structural information throughout the diffusion process within a continuous potential space (Xu et al., 2021; 2023). However, because of the irregular and non-Euclidean structure of graph data, the realization of the diffusion model for graphs still has two main limitations: **(1) High computational complexity.** The core to graph generation is to handle the discreteness, sparsity and other topological properties of the non-Euclidean structure. Since the Gaussian noise perturbation used in the vanilla diffusion model is not suitable for discrete data, the discrete graph diffusion model usually has high time and space complexity due to the problem of structural sparsity. Moreover, the discrete graph diffusion model relies on a continuous Gaussian noise process to create fully connected, noisy graphs (Zhang et al., 2023; Ingraham et al., 2019) which loses structural information and underlying topological properties. **(2) Anisotropy of non-Euclidean structure.** Different from the regular structure data (e.g. pixel matrix or grid structure), the "irregular" non-Euclidean structure embeddings of graph data are anisotropic in continuous latent space (Elhag et al., 2022). As shown in Figure 1(b), the node embeddings of a graph in Euclidean space exhibit significant anisotropy in several specific directions. Recently, some studies (Yang et al., 2023) have shown that isotropic diffusion of the node embedding of the graph in the latent

[1]Key Lab of Education Blockchain and Intelligent Technology, Ministry of Education, Guangxi Normal University, Guilin, China [2]Institute of Artificial Intelligence, Beihang University, Beijing, China [3]School of Software, Beihang University, Beijing, China [4]Beijing Advanced Innovation Center for Big Data and Brain Computing, School of Computer Science and Engineering, Beihang University, Beijing, China. Correspondence to: Xingcheng Fu <fuxc@gxnu.edu.cn>, Jianxin Li <lijx@act.buaa.edu.cn>, Xianxian Li <lixx@gxnu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



(a) Original structure. (b) Euclidean latent space. (c) Hyperbolic latent space.

Figure 1. Visualization of node embeddings by singular value decomposition (SVD); (a) Original structure visualization of the NCAA football graph and different colors indicate different labels(teams); (b) Visualization of node embeddings in 2D Euclidean space and planar projection; (c) Visualization of node embeddings in 2D hyperbolic space and Poincaré disk projection.

space will treat the anisotropic structural information as noise, and this useful structural information will be lost in the denoising process.

Hyperbolic geometric space is widely recognized as an ideal continuous manifold for representing discrete tree-like or hierarchical structures (Cannon et al., 1997; Ungar, 1999; Krioukov et al., 2010; Sun et al., 2024b), and has been widely studied and applied to various graph learning tasks (Sun et al., 2021; Tifrea et al., 2019; Nickel & Kiela, 2017; Sala et al., 2018; Chami et al., 2019; Sun et al., 2024a). Inspired by these studies, we find that hyperbolic geometry has great potential to address non-Euclidean structural anisotropy in graph latent diffusion processes. As shown in Figure 1(c), in hyperbolic space, we can observe that the distribution of node embeddings tends to be isotropic globally, while anisotropy is preserved locally. In addition, hyperbolic geometry unifies angular and radial measures of polar coordinates as shown in Figure 2(a), and can provide geometric measures with physical semantics and interpretability (Papadopoulos et al., 2012). It is exciting that hyperbolic geometry can provide a geometrically latent space with graph geometric priors, able to help deal with the anisotropy of graph structures by special geometric measures.

Based on the above insights, we aim to establish a suitable geometrically latent space based on hyperbolic geometry to design an efficient diffusion process to the non-Euclidean structure for topology-preserving graph generation tasks. However, there are two primary challenges: (1) the additivity of continuous Gaussian distributions is undefined in hyperbolic latent space; (2) devising an effective anisotropic diffusion process for non-Euclidean structures.

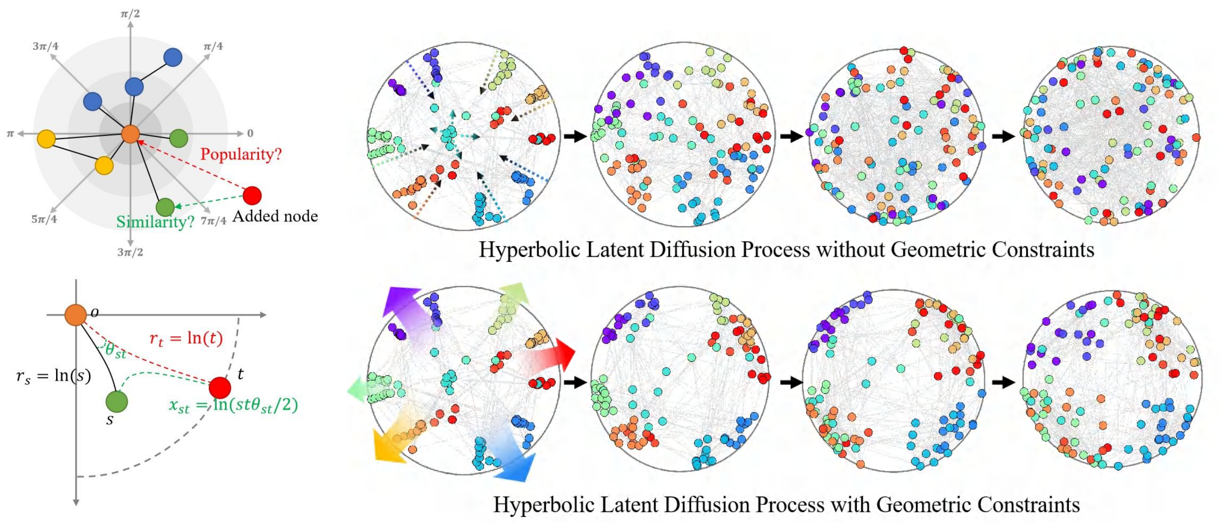
Contributions. To address the challenges, we propose a novel **Hyp**erbolic Geometric Latent **Diff**usion (**HypDiff**) model for the graph generation. For the additive issue of continuous Gaussian distribution in hyperbolic space, we propose an approximate diffusion process based on radial measures. Then the angular constraint was utilized to constrain the anisotropic noise to preserve more structural prior, guiding the diffusion model to finer details of the graph structure. Our contributions are summarized as:

- We are the first to study the anisotropy of non-Euclidean structures for graph latent diffusion models from a geometric perspective, and propose a novel hyperbolic geometric latent diffusion model HypDiff.
- We proposed a novel geometrically latent diffusion process based on radial and angular geometric constraints in hyperbolic space, and addresses the additivity of continuous Gaussian distributions and the issue of anisotropic noise addition in hyperbolic space.
- Extensive experiments on synthetic and real-world datasets demonstrate a significant and consistent improvement of HypDiff and provide insightful analysis for graph generation.

2. Related Works

2.1. Graph Generative Diffusion Model

Different from that learn to generate samples once, like GAN (Wang et al., 2018a;b; Dai et al., 2018), VGAE (Yu et al., 2018; Xu & Durrett, 2018; Grattarola et al., 2019) or GraphRNN (You et al., 2018), the diffusion model (Ho et al., 2020) aims to gradually convert the sample into pure noise by a parameterized Markov chain process. Some recent works (Xu et al., 2021; 2023) employ advanced graph encoders to effectively preserve the inherent structural information throughout the diffusion process within a continuous potential space. Gaussian noise is added on the distribution of nodes and edges of the graph (Vignac et al., 2022), and Gaussian processes are performed on the neighborhood or spectral domain of the graph (Vignac et al., 2022; Jo et al., 2022; Luo et al., 2022). However, existing discrete diffusion models have many challenges in capturing the non-Euclidean structure and preserving underlying topologies.


π/2
3π/4
π/4
π
0
Popularity?
5π/4
Similarity?
7π/4
Added node
3π/2
o
$r_t = \ln(t)$
θ_{st}
$r_s = \ln(s)$
t
s
$x_{st} = \ln(st\theta_{st}/2)$
Hyperbolic Latent Diffusion Process without Geometric Constraints
Hyperbolic Latent Diffusion Process with Geometric Constraints

(a) Geometric interpretation. (b) Hyperbolic latent diffusion.

Figure 2. (a) Geometric interpretation of the hyperbolic geometry, which unifies the radius and angle measurements in polar coordinates and interprets as popularity and similarity respectively; (b) Hyperbolic latent diffusion processing with isotropic/anisotropic noise;

2.2. Hyperbolic Graph Learning

Hyperbolic geometric space was introduced into complex networks earlier to represent the small-world and scale-free complex networks (Krioukov et al., 2010; Papadopoulos et al., 2012). With high capacity and hierarchical-structure-preserving ability, hyperbolic geometry is also used in NLP (Nickel & Kiela, 2017; Tifrea et al., 2019) to learn word representations with hypernym structure. For graph neural networks, hyperbolic space is recently introduced into graph neural networks (Liu et al., 2019; Chami et al., 2019; Sun et al., 2021; 2022). $\mathcal{P}$-VAE (Mathieu et al., 2019) and Hyper-ANE (Liu et al., 2018) extend VAE and GAN into the hyperbolic versions to learn the hierarchical representations. To sum up, hyperbolic geometry provides an intuitive and efficient way of understanding the underlying structural properties of the graph.

3. Methodology

In this section, we present our **Hyp**erbolic geometric latent **Diff**usion model (**HypDiff**) for addressing the two main challenges. The key insight is that we leverage hyperbolic geometry to abstract the implicit hierarchy of nodes in the graph and introduce two geometric constraints to preserve important topological proprieties, such as scale-free, navigability, and modularity. Considering the successful experiences of graph latent diffusion models (Xu et al., 2023), we adopt a two-stage training strategy framework in our practice. We first train the hyperbolic autoencoder to obtain the pre-trained node embeddings, and then train the hyperbolic geometric latent diffusion process. The architecture is shown in Figure 3.

3.1. Hyperbolic Geometric Autoencoding

Since our work aims to improve the diffusion process in a continuous potential space for graph data, we first embed the graph data $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ into a low-dimensional hyperbolic geometric space. We consider a hyperbolic variant of the auto-encoder, consisting of the *hyperbolic geometric encoder* and the *Fermi-Dirac decoder*. Where the *hyperbolic geometric encoder* encodes the graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ into the hyperbolic geometric space to obtain a suitable hyperbolic representation, and the *Fermi-Dirac decoder* decodes the hyperbolic representation back into the graph data domain.

Hyperbolic Encoder and Decoder. Based on the idea of differential geometry, hyperbolic machine learning usually maps a point in hyperbolic space to its tangent plane for computing, and then maps the result back to the hyperbolic space. These mapping operations are defined as follows.

The hyperbolic manifold $\mathbb{H}^d$ and the tangent space $\mathcal{T}_{\mathrm{x}}$ can be mapped to each other via *exponential map* and *logarithmic map* (Ganea et al., 2018b). The *exponential map* $\mathrm{expmap}_{\mathbf{x}}^{c}(\cdot)$ and *logarithmic map* $\mathrm{logmap}_{\mathbf{x}}^{c}(\cdot)$ are defined as:

$$\mathrm{expmap}_{\mathbf{x}}^{c}(\mathbf{v}) = \mathbf{x} \oplus_c \frac{1}{\sqrt{c}} \tanh\left(\frac{\sqrt{c}\lambda_{\mathbf{x}}^{c}\,\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}}{\|\mathbf{v}\|},$$
$$\mathrm{logmap}_{\mathbf{x}}^{c}(\mathbf{u}) = \frac{2}{\sqrt{c}\lambda_{\mathbf{x}}^{c}} \tanh^{-1}\left(\sqrt{c}\,\|-\mathbf{x} \oplus_c \mathbf{u}\|\right)\frac{-\mathbf{x} \oplus_c \mathbf{u}}{\|-\mathbf{x} \oplus_c \mathbf{u}\|}, \tag{1}$$

where $\oplus_c$ is Möbius addition and $\lambda_x^c = \frac{2}{1+c\|\mathbf{x}\|^2}$.

Then, we can leverage Multi-Layer Perceptrons(MLP) or Graph Neural Networks(GNNs) by exponential and logarithmic mapping as hyperbolic geometric encoders.

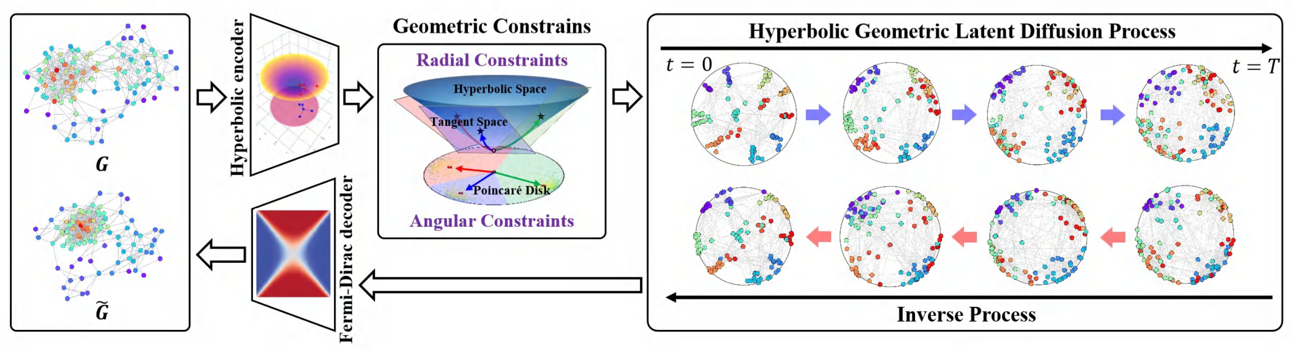


Figure 3. An illustration of HypDiff architecture.

In this paper, we use Hyperbolic Graph Convolutional Neural Networks(HGCN) (Chami et al., 2019) as the hyperbolic geometric encoder. For each layer ℓ of HGCN transforms and aggregates the neighbors' hidden feature of previous layer $\ell - 1$ of the updated node representation $\mathbf{x}^{\ell}$, the aggregation AGG^c and the non-linear activation $\rho^{\otimes^c}$ are defined as follows:

$$\begin{aligned}\text{AGG}^c\left(\mathbf{x}^{\ell}\right) &= \text{expmap}^c_{\mathbf{x}^{\ell}}\left(\sum_{j\in\mathcal{N}(i)} w_j \text{logmap}^c_{\mathbf{x}^{\ell}}\left(\mathbf{x}^{\ell}\right)\right), \\ \mathbf{x}^{\ell+1} &= \rho^{\otimes^c}\left(\text{AGG}^c\left(\left(\mathbf{W}^{\ell+1}\otimes^{c^{\ell}}\mathbf{x}^{\ell}\right)\oplus^{c^{\ell}}\mathbf{b}^{\ell+1}\right)\right),\end{aligned} \tag{2}$$

Optimization of Autoencoding. Due to the additive failure of the Gaussian distribution in hyperbolic space, we cannot directly use *Riemannian normal distribution* or *wrapped normal distribution*. Instead of hyperbolic diffusion embedding (Lin et al.) using the product space of multiple manifolds, we propose a new diffusion process in hyperbolic space, which will be described in detail in Section 3.2. Following $\mathcal{P}$-VAE (Mathieu et al., 2019), for compute efficiency, the Gaussian distribution of hyperbolic space is approximated by the Gaussian distribution of the tangent plane $\mathcal{T}_{\mu}$. The optimization of hyperbolic geometric autoencoding is as follows:

$$\mathcal{L}_{\text{HAE}} = -\mathbb{E}_{q_{\phi}(\mathbf{z_x}|\mathbf{x})}\text{logmap}^c_{\mathbf{o}} p_{\xi}\left(\mathbf{x}|\mathbf{z_x}\right), \tag{3}$$

where $\log^c_{\mathbf{o}}$ is the logarithmic mapping of the north pole (origin) $\mathbf{o}$ of hyperbolic space to simplify the computation.

3.2. Hyperbolic Geometric Latent Diffusion Process

Unlike the linear addition in Euclidean space, hyperbolic space utilizes Möbius addition, posing challenges for diffusion over a hyperbolic manifold. Furthermore, the isotropic noise leads to a rapid reduction of signal-to-noise ratio making it difficult to preserve topological information, and for the detailed results and analysis please refer to Appendix B. In light of these issues, we propose a novel diffusion process to address both of them.

Hyperbolic Anisotropic Diffusion. The anisotropy of the graph in the latent space contains an inductive bias of the graph structure, where the most critical challenge is how to determine the dominant directions of the anisotropic features. In additionally, on hyperbolic manifolds, neither the wrapped normal distribution of the isotropic setup nor the anisotropic setup satisfies this property:

$$\begin{aligned}&\boldsymbol{\eta} \not\sim \boldsymbol{\eta}_1 \oplus_c \boldsymbol{\eta}_2, \\ &\boldsymbol{\eta} \sim \mathcal{N}^c_{\mathbb{H}}\left(0, (\sigma_1^2 + \sigma_2^2)I\right), \\ &\boldsymbol{\eta}_1 \sim \mathcal{N}^c_{\mathbb{H}}\left(0, \sigma_1^2 I\right), \boldsymbol{\eta}_2 \sim \mathcal{N}^c_{\mathbb{H}}\left(0, \sigma_2^2 I\right).\end{aligned} \tag{4}$$

where c is Hyperbolic curvature and $\mathcal{N}^c_{\mathbb{H}}$ is the Wrapped Gaussian distribution. We propose a hyperbolic anisotropic diffusion framework to solve both challenges. The detailed proof process can be found in the Appendix C.1. The core idea is to select the main diffusion direction (i.e., angle) based on the similarity clustering of nodes, which is equivalent to dividing the hyperbolic latent space into multiple sectors. Then we project the nodes of each cluster onto its center's tangent plane for diffusion.

Let $\mathbf{h}$ denote the embedding of the graph in the hyperbolic space and $\mathbf{h}_i$ denote the i-th node in it. Let $\mathbf{h}_i$ belong to the k-th cluster and its clustering center coordinates are μ_k, then the node $\mathbf{h}_i$ is represented in the tangent space of μ_k as x_{0_i}:

$$\mathbf{x}_{0_i} = \text{logmap}^c_{\mu_k}\left(h_i\right). \tag{5}$$

where μ_k is the central point of cluster k obtained by Hyperbolic-Kmeans (h-kmeans) (Hajri et al., 2019) algorithm. Note that the clusters can be obtained by any clustering algorithm based on similarity in the pre-processing stage. Moreover, the hyperbolic clustering parameter k has the following property:

Theorem 3.1. *Given the hyperbolic clustering parameter $k \in [1, n]$, which represents the number of sectors dividing the hyperbolic space (disk). The hyperbolic anisotropic* *diffusion is equivalent to directional diffusion in the Klein model $\mathbb{K}_c^n$ with multi-curvature $c_{i\in|k|}$, which is an approximate projecting onto the tangent plane set $\mathcal{T}_{\mathbf{o}_{i\in\{|k|\}}}$ of the centroids $\mathbf{o}_{i\in\{|k|\}}$.*

The proof is in the Appendix C.2. This property elegantly establishes the relationship between our approximation algorithm and the Klein model with multiple curvatures. Our algorithm exhibits specific behaviors based on the value of k, it allows for a more flexible and nuanced representation of anisotropy based on the underlying hyperbolic geometry, enabling improved accuracy and efficiency in subsequent noise addition and training.

Geometric Constraints. Hyperbolic geometry can naturally and geometrically describe the connection pattern of nodes during graph growth (Papadopoulos et al., 2012). The Popularity-Similarity Optimization (PSO) model is a generative network model to describe how random geometric graphs grow in hyperbolic spaces. Specifically, in hyperbolic geometric space, the new node distance can represent a convenient single-metric by using a combination of the two geometric attractiveness priors, *radial popularity* and *angular similarity*. As shown in Figure 2(a), the popularity of a node can be abstracted by its radial coordinates and the similarity can be expressed by its angular coordinate distances in the hyperbolic space, and more detail can be referred to Appendix D.

Our goal is to model a diffusion with geometric radial growth, and where this radial growth is consistent with hyperbolic properties. Considering that we need to maintain this kind of hyperbolic growth tendency in the tangent plane, we use the following formulas:

$$x_t = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\epsilon + \delta\tanh[\sqrt{c}\lambda_o^c t/T_0]x_0, \quad (6)$$

where ϵ is Gaussian noise and δ is the radial popularity coefficient that controls the diffusion strength of each node in hyperbolic space. T_0 is a constant to control the speed of control of radial growth rate. $\lambda_x^c = \frac{2}{1+c\|\mathbf{x}\|^2}$

Then, we discuss the content only on a cluster tangent plane. The main reason why the general diffusion model does not perform well on the graph is the decline of the fast signal-to-noise ratio. Inspired by directional diffusion model (Yang et al., 2023), we designate the direction of the geodesic between each cluster's center point and the north pole $\mathbf{o}$ as the target diffusion direction while imposing constraints for forward diffusion processes. Specifically, the angular similarity constraints for each node i can be obtained by:

$$\begin{aligned}&\mathbf{z} = \operatorname{sgn}\left(\operatorname{logmap}_{\mathbf{o}}^c\left(\mathbf{h}_{\mu_i}\right)\right) * \epsilon,\\&\epsilon \sim \mathcal{N}\left(0, I\right),\end{aligned} \quad (7)$$

where z represents the angle constrained noise,ϵ is the Gaussian noise, $\mathbf{h}_{\mu_i}$ is the clustering center corresponding to the i-th node. When the number of steps is spread out enough, our results would satisfy some normal distribution.

Combining the radial and angular constraints, our geometric diffusion process can be described as:

$$x_t = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\mathbf{z} + \delta\tanh[\sqrt{c}\lambda_o^c t/T_0]x_0, \quad (8)$$

Theorem 3.2. *Let x_t indicate the node x at the t-step in the forward diffusion process Eq (8). As $t\to\infty$, the low-dimensional latent representation $\mathbf{x}_t$ of node x satisfies:*

$$\lim_{t\to\infty}\mathbf{x}_t \sim \mathcal{N}_f\left(\delta\mathbf{x}_0, I\right). \quad (9)$$

where $\mathcal{N}_f$ is an approximate folded normal distribution. More detail and proof can be referred to in the Appendix E, and more implementation details are provided in Appendix F.

Figure 2(b) illustrates examples of the diffusion process with/without geometric constraints in hyperbolic space. We can observe that by adding isotropic noise to the hyperbolic latent diffusion process, the final diffusion result is completely random noise. In contrast, the hyperbolic latent diffusion process with geometric constraints can significantly preserve the anisotropy of the graph. In other words, after the graph diffusion, the result still preserves the important inductive bias of the graph below rather than the completely random noise, which will directly affect the performance and generation quality of the denoising process

Training and generation. Then, we follow the standard denoising process (Ho et al., 2020; Yang et al., 2023) and train a denoising network to simulate the process of reverse diffusion. We use a denoising network architecture of DDM based on UNET for training to predict $\mathbf{x}_0$, as follows:

$$\mathcal{L}_{\text{HDM}} = \mathbb{E}\left\|f_\theta\left(X_t, A, t\right) - X_0\right\|^2. \quad (10)$$

Note that the loss function of our geometric diffusion model remains consistent with DDPM (Ho et al., 2020) based on Theorem 3.2. The proof refers to the Appendix F.

Regarding the generation, we propose an efficient sampling method based on theorem 3.1. Furthermore, we demonstrate that it is possible to sample at once in the same tangent space instead of sampling in different cluster center tangent spaces to improve efficiency. As to the denoising process, we adopt a denoising process that can be used in generalized diffusion models(Yang et al., 2023). Specifically, where a recovery operator and a noise addition operator are abstracted for use in various diffusion methods. All the specifics regarding each stage of the diffusion process, along with the theoretical derivation, are documented in the Appendix F.

Similar to other hyperbolic learning model (Krioukov et al., 2010; Chami et al., 2019; Ganea et al., 2018a), we utilize

Algorithm 1 Training of HypDiff

```
Input: Graph $\mathcal{G} = \{\mathbf{X}, \mathbf{A}\}$; Number of training epochs $E$;
Parameter: $\theta$ initialization;
Output: Predicted raw embedding $\hat{x_{\mathbb{H}}}$
Encoding node to hyperbolic space $x_{\mathbb{H}} \leftarrow$ Eq. (3);
Compute $k$-clusters by h-Kmeans;
Project the embeddings onto each $\mathcal{T}_{\mathbf{o}_{i\in\{|k|\}}}$
for $e = 1$ to $E$ do
    Get the embeddings $x_{\mathbb{H}^t}$ of $t$-steps Eq. (8) ;
    Predict the raw embeddings $\hat{x_{\mathbb{H}}}$ ;
    Compute the loss $\mathcal{L} = \mathcal{L}_{\mathrm{HDM}} \leftarrow$ Eq. (10);
    Update $\theta \leftarrow \theta - \eta\nabla\theta$.
end for
```

the *Fermi-Dirac decoder* (Krioukov et al., 2010; Nickel & Kiela, 2017) to compute the connection probability.

$$\mathcal{F}_c\left(u, v; \theta^{\mathcal{F}}\right) = \left[e^{(\mathbf{d}^c_{\mathbb{H}}(u,v)^2-\rho)/\tau} + 1\right]^{-1}, \qquad (11)$$

where $\mathbf{d}^c_{\mathcal{H}}$ is the hyperbolic distance and (ρ, τ) are hyper-parameters of Fermi-Dirac distribution. Finally, the results generated by each tangent space are mapped back into the original hyperbolic space to reconstruct the final generated graph. The diffusion and reverse processes are summarized in Algorithm 1 and Algorithm 2.

Complexity Analysis. Let $G = (X, E)$ be one of the graphs set G^s, where X is the n-dimensional node eigen-vector and E is the $m * m$-dimensional adjacency matrix of the graph. s is the number of graphs in the graph set G^s. **Time complexity**: The time complexity of hyperbolic graph encoding is $O((1(t) + k)md)$. For the forward diffusion process, the complexity is $O(md)$. The training of denoising networks is essentially the same as other diffusion models and does not require additional computing time as $O(md) * 1(t)$. Overall, the total time complexity of the diffusion process is $O(1(t) * 2md) + O((k + 2)md)$ in one epoch. **Space complexity** In our approach, since we embed the graphs in hyperbolic space, each graph is represented as a $m * d$-dimensional vector in the hyperbolic space, which means that our diffusion scale is $O(smd)$. For a more detailed complexity analysis please refer to Appendix G.

4. Experiment

In this section, we conduct comprehensive experiments to demonstrate the effectiveness and adaptability of HypDiff [1] in various datasets and tasks. We first presented the experimental settings and then showcased the results.

[1]The code is available at `https://github.com/RingBDStack/HypDiff`.

4.1. Datasets

We estimate the capabilities of HypDiff in various downstream tasks while conducting experiments on synthetic and real-world datasets. In addition, we construct and apply node-level and graph-level datasets for node classification and graph generation tasks. We elaborate on more details as follows.

Synthetic Datasets. We first use two famous graph theoretical models, *Stochastic Block Model (SBM)* and *Barabási-Albert (BA)*, to generate a node-level synthetic dataset with 1000 nodes for node classification, respectively. (1) **SBM** portrays five equally partitioned communities with the edge creation of intra-community $p = 0.21$ and inter-community $q = 0.025$ probabilities. (2) **BA** is grown by attaching new nodes each with random edges between 1 and 10. Then we employ four generic datasets with different scales of nodes $|V|$ for graph generation tasks. Then, four datasets are generated for the graph-level task. (3) **Community** contains 500 two-community small graphs with $12 \leq |V| \leq 20$. Each graph is generated by the Erdős-Rényi model with the probability for edge creation $p = 0.3$ and added $0.05\,|V|$ inter-community edges with uniform probability. (4) **Ego** comprises 1050 3-hop ego-networks extracted from the PubMed network with $|V| \leq 20$. Nodes indicate documents and edges represent their citation relationship. (5) **Barabási-Albert (G)** is a generated graph-level dataset by the BA model (aka. BA-G to distinct node-level BA) with 500 graphs where the degree of each node is greater than four. (6) **Grid** describes 100 standard 2D grid graphs which have each node connected to its four nearest neighbors.

Real-world Datasets. We also carry out our experiments on several real-world datasets. For the node classification task, we utilize (1) two citation networks of academic papers including **Cora** and **Citeseer**, where nodes express documents and edges represent citation links, and (2) **Polblogs** dataset which is political blogs and is a larger size dataset we used. With the graph generation task, we exploit four datasets from different fields. (3) **MUTAG** is a molecular network whose each graph denotes a nitro compound molecule. (4) **IMDB-B** is a social network, symbolizing the co-starring of the actors. (5) **PROTEINS** is a protein network in which nodes represent the amino acids and two nodes are connected by an edge if they are less than 6 Angstroms apart. (6) **COLLAB** is a scientific collaboration dataset, reflecting the collaboration of the scientists. Statistics of the real-world datasets Table H can be found in Appendix H.

4.2. Experimental Setup

Baselines. To evaluate the proposed HypDiff , we compare it with well-known or state-of-the-art graph learning methods which include: (1) **Euclidean graph representation methods**: VGAE (Kipf & Welling, 2016) designs a

Table 1. Summary of node classification Micro-F1 and Macro-F1 scores (%) based on the average of five runs on synthetic and real-world datasets. (Result: average score ± standard deviation (rank); **Bold**: best; Underline: runner-up.)

Method	Synthetic Datasets				Real-world Datasets						Avg. R.
	SBM		BA		Cora		Citeseer		Polblogs		
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	
VGAE	20.5±2.1	15.4±1.1	37.4±1.7	15.9±2.3	79.7±0.4	78.1±0.2	63.8±1.4	55.5±1.3	79.4±0.8	79.4±0.8	4.6
ANE	39.9±1.1	33.9±1.8	46.0±3.0	19.3±2.7	69.3±0.1	66.4±0.1	50.2±0.1	49.5±0.6	80.8±0.1	80.7±0.1	4.3
GraphGAN	38.6±0.5	38.9±0.3	43.6±0.6	24.6±0.5	71.7±0.1	69.8±0.1	49.8±1.0	45.7±0.1	77.5±0.6	76.9±0.4	4.8
$\mathcal{P}$-VAE	57.9±1.3	53.0±1.5	38.4±1.4	20.0±0.3	79.6±2.2	77.5±2.5	**67.9±1.7**	60.2±1.9	79.4±0.1	79.4±0.1	3.2
Hype-ANE	18.8±0.3	11.9±0.1	56.9±2.4	31.6±1.2	80.7±0.1	79.2±0.3	64.4±0.3	58.7±0.0	83.6±0.4	83.6±0.4	3.0
HypDiff	**70.5±0.1**	**69.4±0.1**	**58.3±0.1**	**40.0±0.1**	**82.4±0.1**	**81.2±0.1**	67.8±0.2	**60.4±0.3**	**85.7±0.1**	**85.4±0.1**	**1.1**

variational autoencoder for graph representation learning. ANE (Dai et al., 2018) trains a discriminator to align the embedding distribution with a predetermined fixed prior. GraphGAN (Wang et al., 2018b) learns the sampling distribution for negative node sampling from the graph. (2) **Hyperbolic graph representation learning**: $\mathcal{P}$-VAE (Mathieu et al., 2019) is a variational autoencoder utilizing the Poincaré ball model within hyperbolic geometric space. Hype-ANE (Liu et al., 2018) is a hyperbolic adversarial network embedding model that extends ANE into hyperbolic geometric space. (3) **Deep graph generative models**: VGAE (Kipf & Welling, 2016) can be used for graph generation tasks by treating each graph as a batch size. GraphRNN (You et al., 2018) is a deep auto-regressive generative model that focuses on graph representations under different node orderings. (4) **Graph diffusion generative models**: GDSS (Jo et al., 2022) simultaneously diffuses node features and adjacency matrices to learn their scoring functions. DiGress (Vignac et al., 2022) is a discrete denoising diffusion model that progressively recovers graph properties by manipulating edges. GraphGDP (Huang et al., 2022) is a position-enhanced graph score-based diffusion model for graph generation. EDGE (Chen et al., 2023) is a discrete diffusion process for large graph generation.

Settings. A fair parameter setting for the baselines is the default value in the original papers and appropriate adjustments for new datasets. For HypDiff, the encoder is 2-layer HGCN with 256 representation dimensions, the edge dropping probability to 2%, the learning rate to 0.001. Additionally, the diffusion processing set diffusion strength δ as 0.5, and the denoising network follows the setting in DDM(Yang et al., 2023). We use Adam as an optimizer and set L2 regularization strength as 1e-5. For the metric, we use the F1 scores of the node classification task and the maximum mean discrepancy scores of Degree, Cluster, and Spectre and the F1 score of precision-recall and density-coverage (F1 pr and F1 dc) to evaluate graph generation results.

The richer experimental results under the other indicators are shown in Appendix I. All experiments adopt the implementations from the PyTorch Geometric Library and Deep Graph Library. The reported results are the average scores and standard deviations over 5 runs. All models were trained and tested on a single Nvidia A100 40GB GPU.

4.3. Performance Evaluation

We show the F1 scores of the node classification task in Table 1 and the statistics of MMD distance and F1 scores between the original and generated graph in the graph generation task in Table 2, Table 3and Table C.3. A higher score reported in F1 indicates a more accurate prediction of the node and fidelity of the generated graph. At the same time, a smaller MMD distance suggests better generative capabilities of the model from the perspective of graph topological properties.

Node classification. HypDiff demonstrates superior performance which outperforms nearly all baseline models, achieving the highest ranking and revealing excellent generalization. This implies that HypDiff can preserve essential properties within complex structures, enabling better distinctive and utility of the dependencies between nodes across hierarchical levels in hyperbolic space.

Graph Generation. Successively, we focused on validating the graph generation capability of HypDiff. The results of the MMD metrics (You et al., 2018) for the graph generation task are reported in Table 2 and Table 3. Our MMD is computed with the RBF kernel which is a more stable and comprehensive metric to measure the diversity and realism of generated graphs Using the finer-grained metrics, we observed our approach's outstanding performance. We are further concerned with the fidelity and diversity of the generated results which yielded conclusions consistent with the previous and are reported in Table C.3. Specifically, HypDiff depicts superior overall performance compared to the state-of-the-art model auto-regressive model GraphRNN and discrete diffusion method DiGress. Furthermore, our model can effectively capture the local structure through similarity constraints and achieve competitive performance on highly connected graph data.

Table 2. Generation results about the MMD distance between the original and generated graphs.
(Result: scores (rank) and average rank; **Bold**: best; Underline: runner-up.)

Method	Synthetic Datasets						Real-world Datasets					
	Community			BA-G			MUTAG			PROTRINS		
	Degree	Cluster	Spectre	Degree	Cluster	Spectre	Degree	Cluster	Spectre	Degree	Cluster	Spectre
VGAE	0.365	0.025	0.507	0.775	1.214	0.398	0.255	2.000	0.744	0.705	0.979	0.700
GraphRNN	**0.002**	0.027	**0.004**	**0.122**	0.262	0.007	0.537	0.013	0.476	**0.009**	0.071	0.017
GDSS	0.094	0.031	0.052	0.978	0.468	0.917	0.074	0.021	**0.003**	1.463	0.168	0.013
DiGress	0.226	0.158	0.194	0.654	1.171	0.268	0.100	0.351	0.082	0.108	0.062	0.079
GraphGDP	0.046	0.016	0.042	0.698	0.188	0.053	0.127	0.057	0.050	0.103	0.240	0.088
EDGE	0.021	0.013	0.040	0.282	**0.010**	0.090	**0.024**	0.597	0.468	0.033	0.523	0.024
HypDiff	**0.002**	**0.010**	0.028	0.216	0.021	**0.004**	0.048	**0.001**	0.040	0.133	**0.004**	**0.012**

Table 3. Generation additional results about the MMD distance between the original and generated graphs.

Method	Synthetic Datasets						Real-world Datasets					
	Ego			Grid			IMDB-B			COLLAB		
	Degree	Cluster	Spectre	Degree	Cluster	Spectre	Degree	Cluster	Spectre	Degree	Cluster	Spectre
VGAE	0.414	0.156	0.456	**0.050**	2.000	0.145	0.514	1.405	0.700	0.418	1.174	0.700
GraphRNN	0.206	0.539	0.157	0.203	0.043	**0.042**	0.137	**0.252**	0.423	0.044	0.036	0.510
GDSS	1.034	0.143	0.667	0.111	**0.005**	0.886	0.904	1.729	0.748	0.773	1.589	0.502
DiGress	0.110	0.056	0.122	0.689	1.115	0.203	0.166	0.425	0.159	0.022	**0.008**	**0.003**
GraphGDP	0.059	0.115	0.054	0.872	1.001	0.174	0.123	0.638	0.257	0.092	0.291	0.138
EDGE	**0.023**	**0.048**	0.023	0.223	0.072	0.802	0.041	0.874	**0.026**	0.023	0.569	0.034
HypDiff	0.075	0.090	**0.015**	0.422	0.665	0.137	**0.034**	0.257	0.030	**0.016**	0.023	0.009

4.4. Analysis of HypDiff

In this subsection, we present the experimental results to intuitively convey our discovery and initiate a series of discussions and analyses.

Ablation Study. This study is to highlight the role of geometric constraints of HypDiff. We conducted experiments on three real-world datasets to validate the node classification performance and removed radial popularity (**HypDiff (w/o P)**), angular similarity (**HypDiff (w/o S)**) and total geometric prior(**HypDiff (w/o PS)**) components as the variant models. We show the results in Figure 4. The radial popularity is evident in facilitating hyperbolic diffusion processes, thereby showcasing the advantage of hyperbolic geometry in capturing the underlying graph topology. Furthermore, the angular similarity also significantly preserves the local structure of the graph, compensating for the limitations of hyperbolic space in capturing local connectivity patterns. In summary, the hyperbolic geometric prior plays a crucial role in capturing non-Euclidean structures.

Sensitivity Analysis of Geometric Constraints. To investigate the impact of the number of clusters k and the geometric prior coefficient δ on the model performance, we conducted the sensitivity analysis on the real-world and synthetic graph datasets.

As to cluster k can be understood as the strength of the angular constraint, the results of three datasets with different structures are shown in Fig 5 (Left). Specifically, **Cora** has a real-world connected structure, **SBM** has a complex community structure, and **Fractal** has self-similarity and hierarchy properties. It can be observed that k has different sensitivities in different structured datasets, indicating that different graph structures have different approximate accuracies for anisotropy capture. (1) Cora: Being a real dataset, Cora exhibits a highly complex structure. A larger value of k can better capture its data characteristics. However, excessively large values of k, such as k=500, introduce significant anisotropy. This excessive anisotropy poses challenges in learning the dataset's probability distribution. (2) SBM: Despite having only five major categories, SBM displays a complex structure within each community. Increasing k yields improved results, likely due to the intricacies within each community that benefit from a higher resolution. (3) Hierarchical Fractal: Interestingly, changes in k have minimal impact on the training effectiveness of Hierarchical Fractal. This phenomenon may cause by the self-similarity inherent in its fractal structure, since it preserves consistent local geometric properties.

Correspondingly, geometric prior coefficient δ can be understood as the strength of the radial constraint, the results of three real-world datasets are shown in Fig 5 (Right). The

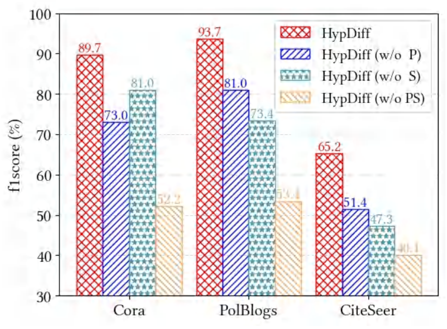


Figure 4. Ablation study results.

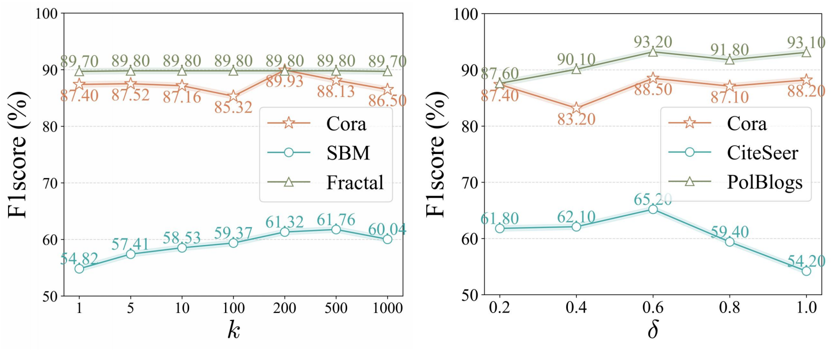


Figure 5. Sensitivity analysis of geometric constraints.

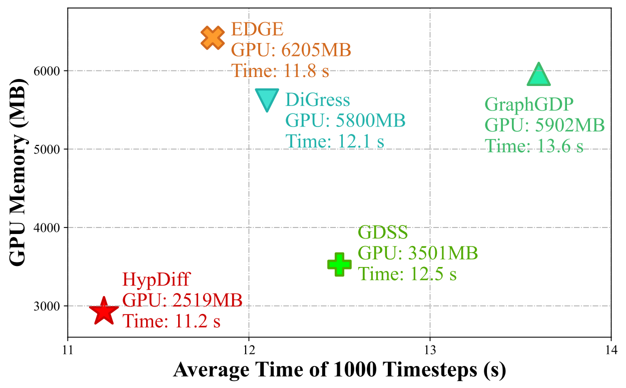


Figure 6. Efficiency analysis on IMDB-B for graph generation.

stronger the constraint, the smaller the diffusion step in the radial direction of the hyperbolic space. In other words, the radial constraint adjusts the degree of coupling between graph diffusion and hyperbolic geometry, approximating Euclidean space when radial growth at polar coordinates is linear. The experiment results depicted in the figure illustrate a progressive improvement as the parameter δ gradually increases from very small values. This improvement is attributed to the retention of more geometric a priori information during the diffusion process. However, as δ continues to increase, there is an initial decline in effectiveness, possibly due to the heightened geometric prior information. Actually, the increase in geometric information also leads to an increase in anisotropy during diffusion, resulting in a reduced variance among nodes within the same tangent space. Consequently, learning an optimal probability distribution becomes more challenging. As the geometric prior information further increases, a rebound effect is observed, indicating that the increment in prior information surpasses the interference caused by anisotropy on node representation. In additional, it can be observed that the tree-like graphs requires lower radial constraints, while the graph with high connectivity requires stronger radial constraints.

Diffusion Efficiency Analysis. We report the training time for HypDiff and other graph diffusion baselines with the same configurations on IMDB-B. We conduct experiments with the hardware and software configurations listed in Section 4.2. We report the results from the time and space costs of the diffusion process. The result is shown in Figure 6, our HypDiff comprehensively outperforms other baselines in diffusion time and GPU memory cost. Compared with the discrete graph diffusion model, our model directly diffuses each node of the graph with structure-preserving based on the latent diffusion model, so the space complexity is much lower than that of the direct diffusion of discrete and sparse structural information(e.g. adjacent/Laplace matrix). The performance of each dataset is in the Appendix J,

Visualization. We compare the contributions of two diffusion generation models, HypDiff and GDSS, to graph generation tasks by visualizing networks generated by five well-accepted graph theoretical models. By comparing the ability to generate network structures with different topological features, we demonstrate the superiority of our model in terms of topological feature capability preservation. Also, for highly connected grid structures, we discuss the shortcomings of our model in this regard. We discuss and show the visualization as Figure C.3 in the Appendix I.2.

5. Conclusion

In this paper, we introduce hyperbolic geometric before dealing with the conflict problem between discrete graph data and continuous diffusion model, and propose a novel hyperbolic geometric diffusion model named HypDiff. We propose an improved hyperbolic Gaussian noise generation method based on radial popularity to deal with the additive failure of Gaussian distributions in hyperbolic space. The geometric constraints of angular similarity are applied to the anisotropic diffusion process, to preserve as much various local structure information as possible. Extensive experiments conducted on both synthetic and real-world graphs demonstrate the comprehensive capability of HypDiff.

Acknowledgements

The corresponding authors are Jianxin Li and Xianxian Li. This paper is supported by the National Science and Technology Major Project of China (No.2022ZD0117800), and the National Natural Science Foundation of China (No.U21A20474 and 62302023). We owe sincere thanks to all co-authors for their valuable efforts and contributions.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

Cannon, J. W., Floyd, W. J., Kenyon, R., Parry, W. R., et al. Hyperbolic geometry. *Flavors of geometry*, 31:59–115, 1997.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *NeurIPS*, pp. 4869–4880, 2019.

Chen, X., He, J., Han, X., and Liu, L. Efficient and degree-guided graph generation via discrete diffusion modeling. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4585–4610. PMLR, 2023.

Dai, Q., Li, Q., Tang, J., and Wang, D. Adversarial network embedding. In *AAAI*, pp. 2167–2174, 2018.

Elhag, A. A., Corso, G., Stärk, H., and Bronstein, M. M. Graph anisotropic diffusion for molecules. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

Erdős, P., Rényi, A., et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *NeurIPS*, pp. 5350–5360, 2018a.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *NeurIPS*, pp. 5350–5360, 2018b.

Grattarola, D., Livi, L., and Alippi, C. Adversarial autoencoders with constant-curvature latent manifolds. *Applied Soft Computing*, 81:105511, 2019.

Hajri, H., Zaatiti, H., and Hébrail, G. Learning graph-structured data using Poincaré embeddings and Riemannian K-means algorithms. *CoRR*, abs/1907.01662, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Huang, H., Sun, L., Du, B., Fu, Y., and Lv, W. Graphgdp: Generative diffusion processes for permutation invariant graph generation. In Zhu, X., Ranka, S., Thai, M. T., Washio, T., and Wu, X. (eds.), *ICDM 2022*, pp. 201–210. IEEE, 2022.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. Generative models for graph-based protein design. *NeurIPS*, 32, 2019.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. In *ICML*, pp. 10362–10383. PMLR, 2022.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. In *NeurIPS*, 2016.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguná, M. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Lin, Y. E., Coifman, R. R., Mishne, G., and Talmon, R. Hyperbolic diffusion embedding and distance for hierarchical representation learning. In *ICML*, volume 202, pp. 21003–21025.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *NeurIPS*, pp. 8228–8239, 2019.

Liu, X., Tang, S., and Wang, J. Generative adversarial graph representation learning in hyperbolic space. In *CCIR*, pp. 119–131, 2018.

Luo, T., Mo, Z., and Pan, S. J. Fast graph generative model via spectral diffusion. *arXiv:2211.08892*, 2022.

Ma, T., Chen, J., and Xiao, C. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *NeurIPS*, pp. 7113–7124, 2018.

Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., and Whye Teh, Y. Continuous hierarchical representations with poincaré variational auto-encoders. In *NeurIPS*, 2019.

Muscoloni, A. and Cannistraci, C. V. A nonuniform popularity-similarity optimization (npso) model to efficiently generate realistic complex networks with communities. *New Journal of Physics*, 20(5):052002, 2018.

Naeem, M. F., Oh, S. J., Uh, Y., Choi, Y., and Yoo, J. Reliable fidelity and diversity metrics for generative models. In *ICML 2020*, pp. 7176–7185. PMLR, 2020.

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. A differentiable gaussian-like distribution on hyperbolic space for gradient-based learning. In *ICML*, 2019.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*, pp. 6338–6347, 2017.

Papadopoulos, F., Kitsak, M., Serrano, M. Á., Boguná, M., and Krioukov, D. Popularity versus similarity in growing networks. *Nature*, pp. 537–540, 2012.

Ravasz, E. and Barabási, A.-L. Hierarchical organization in complex networks. *Physical review E*, 67(2):026112, 2003.

Said, S., Bombrun, L., and Berthoumieu, Y. New riemannian priors on the univariate normal model. *Entropy*, 16 (7):4015–4031, 2014.

Sala, F., De Sa, C., Gu, A., and Ré, C. Representation tradeoffs for hyperbolic embeddings. In *ICML*, pp. 4460–4469. PMLR, 2018.

Sun, L., Zhang, Z., Zhang, J., Wang, F., Peng, H., Su, S., and Yu, P. S. Hyperbolic variational graph neural network for modeling dynamic graphs. In *AAAI*, 2021.

Sun, L., Zhang, Z., Ye, J., Peng, H., Zhang, J., Su, S., and Yu, P. S. A self-supervised mixed-curvature graph neural network. In *AAAI*, 2022.

Sun, L., Hu, J., Zhou, S., Huang, Z., Ye, J., Peng, H., Yu, Z., and Yu, P. S. Riccinet: Deep clustering via a riemannian generative model. In *WWW*, 2024a.

Sun, L., Huang, Z., Wang, Z., Wang, F., Peng, H., and Yu, P. S. Motif-aware riemannian graph neural network with generative-contrastive learning. In *AAAI*, 2024b.

Tifrea, A., Bécigneul, G., and Ganea, O. Poincare glove: Hyperbolic word embeddings. In *ICLR*, 2019.

Ungar, A. A. The hyperbolic pythagorean theorem in the poincaré disc model of hyperbolic geometry. *The American mathematical monthly*, 106(8):759–763, 1999.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *ICLR*, 2022.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., and Guo, M. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*, pp. 2508–2515, 2018a.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., and Guo, M. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*, pp. 2508–2515, 2018b.

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

Xu, J. and Durrett, G. Spherical latent spaces for stable variational autoencoders. *EMNLP*, 2018.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. Geodiff: A geometric diffusion model for molecular conformation generation. In *ICLR*, 2021.

Xu, M., Powers, A. S., Dror, R. O., Ermon, S., and Leskovec, J. Geometric latent diffusion models for 3d molecule generation. In *ICML*, pp. 38592–38610. PMLR, 2023.

Yang, R., Yang, Y., Zhou, F., and Sun, Q. Directional diffusion models for graph representation learning. *arXiv:2306.13210*, 2023.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. Graphrnn: Generating realistic graphs with deep autoregressive models. In *ICML*, volume 80, pp. 5694–5703. PMLR, 2018.

Yu, W., Zheng, C., Cheng, W., Aggarwal, C. C., Song, D., Zong, B., Chen, H., and Wang, W. Learning deep network representations with adversarially regularized autoencoders. In *SIGKDD*, pp. 2663–2671, 2018.

Zhang, M., Qamar, M., Kang, T., Jung, Y., Zhang, C., Bae, S.-H., and Zhang, C. A survey on graph diffusion models: Generative ai in science for molecule, protein and material. *arXiv:2304.01565*, 2023.

A. Summary of Notations.

Table C.1. Summary of notations.

Symbol	Description
G	Graph
X	Feature matrix
A	Adjacent matrix
d	Dimension of the latent space
n	Number of nodes
$\mathbb{H}$	Hyperbolic space
$\mathbb{K}$	Klein model
$\dot{T}_\mu \mathbb{H}^n$	tangent space of node μ
c	Hyperbolic curvature
η	Random variable
$\operatorname{expmap}_\mu^c(u)$	Exponential map.
$\operatorname{logmap}_\mu^c(x)$	Logarithmic map
λ_x^c	Curvature metric parameters
$\oplus_c$	Möbius' addition
$\mathcal{N}_{\mathrm{H}}^c$	Wrapped normal distribution
$\mathcal{N}(0, I)$	Gaussian distribution
$\mathcal{N}_f(\delta \mathbf{x}_0, I)$.	Approximate folded normal distribution
$\sqrt{\alpha_t}$	Diffusion coefficient
δ	Radial constraint coefficient
k	Hyperbolic clustering parameter
x_0	Initial diffusion embedding coordinates
ϵ	Gaussian noise
z	Angular constrained Gaussian noise
h	Hyperbolic embedding coordinates

B. Anisotropic Diffusion with/without Angular Noise

We refer to the anisotropic angular noise mentioned in Directional Diffusion Model(DDM) (Yang et al., 2023):

$$\epsilon' = \operatorname{sgn}(x_0) \odot |\bar{\epsilon}| \tag{C.1}$$

where ϵ denotes the Gaussian noise, x_0 denotes the graph nodes represent embedding coordinates.

Figure C.1 shows the SNR results of angular noise and white noise in the diffusion process, respectively. With the iterations (diffusion steps) increasing, the $SNR_{Angular}$(red) curve of angular noise is earlier than the SNR_{White}(blue) curve of white noise. The blue curve shows that the isotropic white noise rapidly masks the underlying anisotropic structure or signal during the standard diffusion process, which indicates the local details of the graph structure are greatly lost. HypDiff (red curve) benefits from the angular similarity constraint and can effectively preserve the anisotropic community structure during the diffusion process.

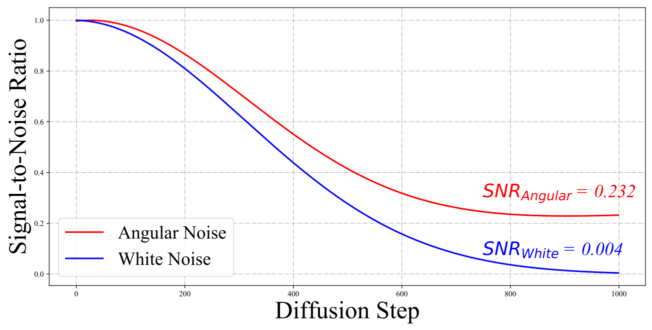


Figure C.1. Curve of Signal-to-Noise Ratio of different diffusion steps.

C. Proof and Analysis

C.1. Proof of the absence of additivity of the package normal distribution

Description of symbols. $\mathbb{B}_c^d$ is a d-dimentional Poincaré Ball space with curvature c. $\mathbf{R}^d$ is a d-dimensional Euclidean space.$\mathbf{G}(\mathbf{z})$ is the metric tensor of the hyperbolic space. $\mathbf{d_p^c}$ is the hyperbolic distance. We first introduce the expression of metric tensor $\mathbf{G}(\mathbf{z})$ in hyperbolic space and the hyperbolic distance $\mathbf{d_p^c}$:

$$\begin{aligned}\mathbf{G}(\mathbf{z}) &= \begin{pmatrix}1 & 0 \\ 0 & \left(\frac{\sinh(\sqrt{c}r)}{\sqrt{c}r}\right)^2 I_{d-1}\end{pmatrix}, \\ \mathbf{d_p^c}(\boldsymbol{z},\boldsymbol{y}) &= \frac{1}{\sqrt{c}}\cosh^{-1}\left(1+2c\frac{\|z-\boldsymbol{y}\|^2}{(1-c\|\boldsymbol{z}\|^2)(1-c\|\boldsymbol{y}\|^2)}\right).\end{aligned} \tag{C.2}$$

Following the $\mathcal{P}$-VAE (Nickel & Kiela, 2017) we can then obtain the differential and integral operators by transforming under hyperbolic polar coordinates and Euclidean space as:

$$\begin{aligned}ds^2_{\mathbb{B}_c^d} &= (\lambda_z^c)^2(dz_1^2+\cdots+dz_d^2) = \frac{4}{(1-c\|x\|^2)^2}dz^2 \\ &= \frac{4}{(1-c\rho^2)^2}(d\rho^2+\rho^2 ds^2_{\mathbb{S}^{d-1}}),\end{aligned} \tag{C.3}$$

let $\mathbf{r}=\mathbf{d_p^c}$, we have

$$\begin{aligned}\mathbf{r} &= \int_0^\rho \lambda_t^c dt = \int_0^\rho \frac{2}{1-ct^2}dt = \int_0^{\sqrt{c}\rho}\frac{2}{1-t^2}\frac{dt}{\sqrt{c}} \\ &= \frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\rho).\end{aligned} \tag{C.4}$$

Then, we have

$$\begin{aligned}ds^2_{\mathbb{B}_c^d} &= \frac{4}{(1-c\rho^2)^2}\frac{1}{4}(1-c\rho^2)^2dr^2 + \left(2\frac{\rho}{1-c\rho^2}\right)^2 ds^2_{\mathbb{S}^{d-1}} \\ &= dr^2 + \left(2\frac{\frac{1}{\sqrt{c}}\tanh(\sqrt{c}\frac{r}{2})}{1-c\left(\frac{1}{\sqrt{c}}\tanh(\sqrt{c}\frac{r}{2})\right)^2}\right)^2 ds^2_{\mathbb{S}^{d-1}} \\ &= dr^2 + \left(\frac{1}{\sqrt{c}}\sinh(\sqrt{c}r)\right)^2 ds^2_{\mathbb{S}^{d-1}},\end{aligned} \tag{C.5}$$

when $\mathbf{c}\to\mathbf{0}$,it recovers the Euclidean line element as

$$\mathbf{ds^2_{R^d}} = \mathbf{dr^2} + \mathbf{r^2 ds^2_{S^{d-1}}}. \tag{C.6}$$

Then, we have the integral calculation as

$$\begin{aligned}\int_{\mathbb{B}_c^d} f(\boldsymbol{z})d\mathcal{M}(\boldsymbol{z}) &= \int_{\mathbb{B}_c^d} f(z)\sqrt{|G(z)|}dz \\ &= \int_{\mathcal{T}_\mu\mathcal{B}_c^d\cong\mathcal{R}^d} f(\boldsymbol{v})\sqrt{|G(\boldsymbol{v})|}d\boldsymbol{v} \\ &= \int_{\mathbb{R}_+}\int_{\mathbb{S}^{d-1}} f(r)\sqrt{|G(r)|}dr r^{d-1} ds_{\mathbf{S}^{d-1}} \\ &= \int_{\mathbb{R}_+}\int_{\mathbb{S}^{d-1}} f(r)\left(\frac{\sinh(\sqrt{c}r)}{\sqrt{c}r}\right)^{d-1} dr r^{d-1} ds_{\mathbf{S}^{d-1}} \\ &= \int_{\mathbb{R}_+}\int_{\mathbb{S}^{d-1}} f(r)\left(\frac{\sinh(\sqrt{c}r)}{\sqrt{c}}\right)^{d-1} dr ds_{\mathbb{S}^{d-1}}.\end{aligned} \tag{C.7}$$

Proof. We introduce the probability density distribution of the wrapped normal distribution. The wrapped normal distribution is mapped to a hyperbolic space by taking the normal distribution in the tangent plane $\mathcal{T}_\mu \mathbb{B}_{\mathbf{c}}^{\mathbf{d}}$ through the exponential map (Said et al., 2014). One can obtain samples as follows:

$$\mathbf{z} = \exp_\mu^c \left(G(\mu)^{-\frac{1}{2}} v \right) = \exp_\mu^c \left(\frac{v}{\lambda_\mu^c} \right), \text{with} v \sim \mathcal{N}(\cdot|\mathbf{0}, \Sigma). \tag{C.8}$$

Anisotropic. In the anisotropic settings, its probability density can be given by:

$$\begin{aligned} \mathcal{N}_{\mathbb{B}_c^d}^{\mathrm{W}}(z|\mu, \Sigma) &= \mathcal{N}\left(G(\boldsymbol{\mu})^{1/2} \log_{\boldsymbol{\mu}}(\boldsymbol{z}) \,|\mathbf{0}, \Sigma) \left(\frac{\sqrt{c} d_p^c(\boldsymbol{\mu}, \boldsymbol{z})}{\sinh(\sqrt{c} d_p^c(\boldsymbol{\mu}, \boldsymbol{z}))} \right)^{d-1} \right. \\ &= \mathcal{N}\left(\lambda_\mu^c \log_\mu(z) \middle| \mathbf{0}, \Sigma \right) \left(\frac{\sqrt{c} d_p^c(\boldsymbol{\mu}, \boldsymbol{z})}{\sinh(\sqrt{c} d_p^c(\boldsymbol{\mu}, \boldsymbol{z}))} \right)^{d-1}. \end{aligned} \tag{C.9}$$

We can plug its density with introducing the variable $v = r\alpha = \lambda_\mu^c \log_\mu(z)$ and utilizing the metric tensor, and we have

$$\begin{aligned} &\int_{\mathbb{B}_c^d} \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z|\mu, \Sigma) d\mathcal{M}(z) \\ &= \int_{\mathcal{T}_\mu \mathbb{B}_c^d \cong \mathbb{R}^d} \mathcal{N}(\boldsymbol{v} \mid \mathbf{0}, \Sigma) \left(\frac{\sqrt{c}\, \|\boldsymbol{v}\|_2}{\sinh(\sqrt{c}\, \|\boldsymbol{v}\|_2)} \right)^{d-1} \sqrt{|G(\boldsymbol{v})|} d\boldsymbol{v} \\ &= \int_{\mathbb{R}^d} \mathcal{N}(\boldsymbol{v} \mid \mathbf{0}, \Sigma) \left(\frac{\sqrt{c}\, \|\boldsymbol{v}\|_2}{\sinh(\sqrt{c}\, \|\boldsymbol{v}\|_2)} \right)^{d-1} \left(\frac{\sinh(\sqrt{c}\, \|\boldsymbol{v}\|_2)}{\sqrt{c}\, \|\boldsymbol{v}\|_2} \right)^{d-1} d\boldsymbol{v} \\ &= \int_{\mathbb{R}^d} \mathcal{N}(\boldsymbol{v} \mid \mathbf{0}, \Sigma)\, d\boldsymbol{v}. \end{aligned} \tag{C.10}$$

Next, we derive whether the sum of two independent wrapped normally distributed variables still satisfies the wrapped normal distribution.

$$\begin{aligned} &\mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_1|\mu_1, \Sigma_1) * \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_2|\mu_2, \Sigma_2) \\ &= \int_{\mathbb{B}_c^d} \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z - z_2|\mu_1, \Sigma_1) \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_2|\mu_2, \Sigma_2) d\mathcal{M}(z_2) \\ &= \int_{\mathbb{R}^d} \mathcal{N}(\boldsymbol{v} - \boldsymbol{v_2} \mid \mathbf{0}, \Sigma_1) \mathcal{N}(\boldsymbol{v_2} \mid \mathbf{0}, \Sigma_2) \left(\frac{\sqrt{c}\, \|\boldsymbol{v} - \boldsymbol{v_2}\|_2}{\sinh(\sqrt{c}\, \|\boldsymbol{v} - \boldsymbol{v_2}\|_2)} \right)^{d-1} d\boldsymbol{v_2} \\ &\mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_1|\mu_1, \Sigma_1) * \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_2|\mu_2, \Sigma_2) \not\propto \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z|\mu, \Sigma). \end{aligned} \tag{C.11}$$

Isotropic. In the isotropic setting, the density of the wrapped normal is given by:

$$\begin{aligned} \mathcal{N}_{\mathsf{B_c}}^{\mathsf{W}}(z|\boldsymbol{\mu}, \sigma^2) &= \frac{d\nu^{\mathsf{W}}(z|\boldsymbol{\mu}, \sigma^2)}{d\mathcal{M}(\boldsymbol{z})} \\ &= (2\pi\sigma^2)^{-d/2} \exp\left(-\frac{d_p^c(\boldsymbol{\mu}, \boldsymbol{z})^2}{2\sigma^2} \right) \left(\frac{\sqrt{c} d_p^c(\boldsymbol{\mu}, \boldsymbol{z})}{\sinh(\sqrt{c} d_p^c(\boldsymbol{\mu}, \boldsymbol{z}))} \right)^{d-1}. \end{aligned} \tag{C.12}$$

Its integral form can be given by:

$$\begin{aligned} &\int_{B_c^d} \mathcal{N}_{E_c^d}^{\mathrm{W}}(z|\boldsymbol{\mu}, \sigma^2) d\mathcal{M}(z) \\ &= \int_{R_+} \int_{S^{d-1}} \frac{1}{Z^{\mathrm{R}}} e^{-\frac{r^2}{2\sigma^2}} r^{d-1} dr ds_{S^{d-1}}, \end{aligned} \tag{C.13}$$

where Z^R is the constant, and it is defined as

$$\begin{aligned} Z^{\mathrm{R}} =& \zeta \binom{d-1}{k} e^{\frac{(d-1-2k)^2}{2} c\sigma^2} \left[1 + \operatorname{erf}\left(\frac{(d-1-2k)\sqrt{c}\sigma}{\sqrt{2}}\right)\right], \\ \zeta =& \frac{2\pi^{d/2}}{\Gamma(d/2)} \sqrt{\frac{\pi}{2}} \sigma \frac{1}{(2\sqrt{c})^{d-1}} \sum_{k=0}^{d-1} (-1)^k. \end{aligned} \tag{C.14}$$

Thus, we can derive its additivity:

$$\begin{aligned} & \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_1|\mu_1, \Sigma_1) * \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_2|\mu_2, \Sigma_2) \\ =& \int_{\mathbb{B}_c^d} \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z - z_2|\mu_1, \Sigma_1) \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_2|\mu_2, \Sigma_2) d\mathcal{M}(z_2) \\ =& \int_{R_+} \int_{S^{d-1}} \frac{1}{Z^{\mathrm{R}^2}} e^{-\frac{(r-r_2)^2}{2\sigma^2}} (r - r_2)^{d-1} \gamma_p^c e^{-\frac{(r_2)^2}{2\sigma^2}} (r_2)^{d-1} dr ds_{S^{d-1}} \\ =& \int_{R_+} \int_{S^{d-1}} \frac{1}{Z^{\mathrm{R}^2}} e^{-\frac{(r^2 - 2rr_2 + 2r_2^2)}{2\sigma^2}} (r_2(r - r_2))^{d-1} \gamma_p^c dr ds_{S^{d-1}} \\ & \gamma_p^c = \left(\frac{\sqrt{c} d_p^c(\boldsymbol{\mu_1}, \boldsymbol{z_1})}{\sinh(\sqrt{c} d_p^c(\boldsymbol{\mu_1}, \boldsymbol{z_1}))}\right)^{d-1} \\ & \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_1|\mu_1, \Sigma_1) * \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z_2|\mu_2, \Sigma_2) \nsim \mathcal{N}_{\mathbb{B}_c^d}^{\mathbf{W}}(z|\mu, \Sigma). \end{aligned} \tag{C.15}$$

Thus, we have demonstrated the lack of additivity in the wrapped normal distribution. □

C.2. Proof of Theorem 3.1

In hyperbolic geometry, four commonly used equivalent models are the Klein model, the Poincare disk model, the Lorentz model, and the Poincare half-plane model. For our analysis in this study, we utilize the Lorentz model. (Ungar, 1999)

Lorentz model $\mathbb{H}^n$ can be denoted by a set of points $z(z \in \mathbb{R}^{n+1})$ through Lorentzian product:

$$\langle z, z' \rangle_{\mathcal{L}} = -z_0 z_0' + \sum_{i=1}^{n} z_i z_i', \tag{C.16}$$

$$\mathbb{H}^n = \left\{ z \in \mathbb{R}^{n+1} \colon \langle z, z \rangle_{\mathcal{L}} = \frac{1}{c}, z_0 > 0 \right\}. \tag{C.17}$$

Tangent space $T_{\boldsymbol{\mu}}\mathbb{H}^n$ is the tangent space of $\mathbb{H}^n$ at μ. $T_{\boldsymbol{\mu}}\mathbb{H}^n$ can be represented as the set of points that satisfy the orthogonality relation with respect to the Lorentzian product:

$$T_\mu \mathbb{H}^n := \{\boldsymbol{u} : \langle \boldsymbol{u}, \boldsymbol{\mu} \rangle_{\mathcal{L}} = 0\}. \tag{C.18}$$

Parallel transport and inverse parallel transport. For an arbitrary pair of point $\mu, \nu \in \mathbb{H}^n$, the parallel transport from ν to μ is defined as a map $\mathrm{PT}_{\nu \to \mu}$ from $T_{\boldsymbol{\nu}}\mathbb{H}^n$ to $T_{\boldsymbol{\mu}}\mathbb{H}^n$ that carries a vector in $T_{\boldsymbol{\nu}}\mathbb{H}^n$ along the geodesic from ν to μ without changing its metric tensor.

The explicit formula for the parallel transport on the Lorentz model is given by:

$$\mathrm{PT}^c_{\nu \to \mu}(v) = v + \frac{\langle \mu - \alpha\nu, v \rangle_{\mathcal{L}}}{\alpha + 1}(\nu + \mu), \tag{C.19}$$

where $\alpha = -\langle \nu, \mu \rangle_{\mathcal{L}}$.The inverse parallel transport is given by:

$$v = \mathrm{PT}^c_{\mu \to \nu}(u). \tag{C.20}$$

Exponential map and Logarithmic map. Exponential map $exp_m u$:$T_{\boldsymbol{\mu}}\mathbb{H}^n \rightarrow \dot{\mathbb{H}}^n$ is a map that we can use to project a vector ν in $T_{\boldsymbol{\mu}}\mathbb{H}^n$ to $\mathbb{H}^n$.The explicit formula for the exponential map on the Lorentz model is given by:

$$z = \exp_{\boldsymbol{\mu}}^c(\boldsymbol{u}) = \cosh\left(\|\boldsymbol{u}\|_{\mathcal{L}}\right)\boldsymbol{\mu} + \sinh\left(\|\boldsymbol{u}\|_{\mathcal{L}}\right)\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_{\mathcal{L}}}. \tag{C.21}$$

The logarithmic map is defined to compute the inverse of the exponential map, mapping the point back to the tangent space. It is given by:

$$u = \log_{\boldsymbol{\mu}}^c(z) = \frac{\operatorname{arccosh}(\alpha)}{\sqrt{\alpha^2 - 1}}(z - \alpha\boldsymbol{\mu}), \tag{C.22}$$

where $\alpha = -\langle \mu, z\rangle_{\mathcal{L}}$.

Klein model. This model of hyperbolic space is a subset of R^n given by K^n, and a point in the Klein model can be obtained from the corresponding point in the hyperbolic model by projection:

$$\pi_{\mathbb{H}\to\mathbb{K}}(\mathbf{x})_i = \frac{x_i}{x_0}. \tag{C.23}$$

with its inverse given by

$$\pi_{\mathbb{K}\to\mathbb{H}}^{-1}(\mathbf{x}) = \frac{1}{\sqrt{1 - \|\mathbf{x}\|^2}}(1, \mathbf{x}). \tag{C.24}$$

An interesting approach is that we can use the angle-preserving nature of the Klein model to construct a mapping from the Lorentz tangent plane to the Klein model via the spherical pole mapping P^c:

$$k_j = \mathrm{P}^c(x_j) = \frac{1}{-c + \sum_{i=1}^n x_i^2}(x_j^2). \tag{C.25}$$

Proof. In our model, m represents the number of clusters and $h_i(h_i \in \mathbb{H}^n)$ represents the points in class $i(i \in 1, 2, 3, ..., m)$, μ_i denotes the hyperbolic center of the cluster i.Each point h_i will be mapped to the tangent plane $T_{\boldsymbol{\mu_i}}\mathbb{H}^n$ of its center μ_i. Let x_i denote the point after mapping to the tangent plane, it can be calculated by:

$$x_i = \log_{\boldsymbol{\mu_i}}^c(h_i) \tag{C.26}$$

If we consider all the points as being in the tangent plane to the North Pole $T_{\boldsymbol{o}}\mathbb{H}^n$, then their corresponding coordinates are:

$$X = (x_1, x_2, ..., x_m) \tag{C.27}$$

For a curvature c_i, if the following equation is satisfied:

$$\log_{\boldsymbol{o}}^{c_i}(h_i) = \mathrm{PT}_{o\to\mu_i}^{c_0}(\log_{\boldsymbol{o}}^{c_0}(h_i)) \tag{C.28}$$

then it is possible to transform the tangent planes from the various centers to the tangent plane at the North Pole and unify them into the Klein model. The mapping point k is given by:

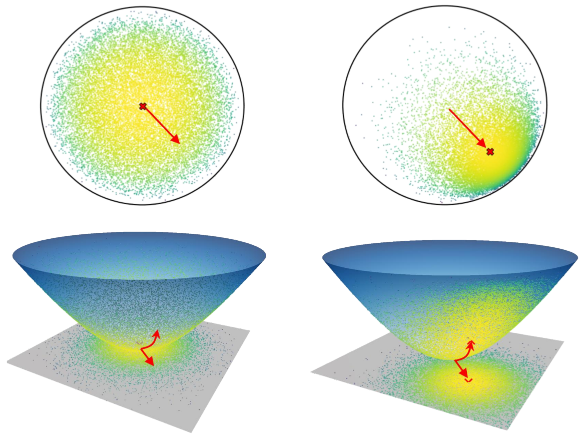
$$\begin{aligned} k_i &= \mathrm{P}^{c_0}(\log_{\boldsymbol{o}}^{c_0}(h_i)) \\ &= \mathrm{P}^{c_0}(\mathrm{PT}_{\mu_i\to o}^{c_0}(\log_{\boldsymbol{\mu_i}}(h_i))) \\ &= \mathrm{P}^{c_0}(\mathrm{PT}_{\mu_i\to o}^{c_0}(x_i)) \\ &= \pi_{\mathbb{H}\to\mathbb{K}}(\mathbf{h_i}) \\ &= \mathrm{P}^{c_i}(\log_o^{c_i}(h_i)). \end{aligned} \tag{C.29}$$

□

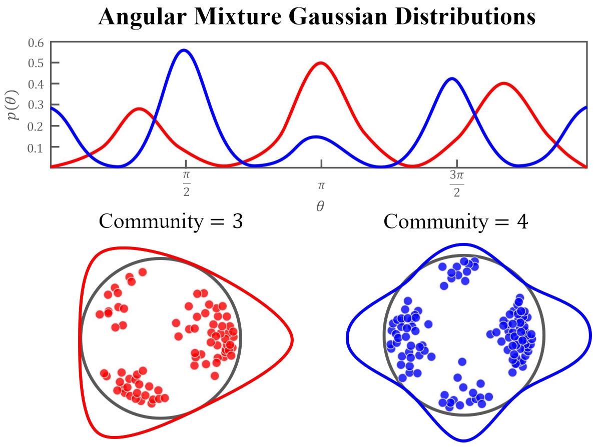
This implies that our approach essentially involves mathematically projecting points to approximate a Klein model comprising multiple curvatures. The process represents a topological reconstruction of the geometric space derived from the original graph structure, thereby enhancing our ability to capture the geometric properties inherent in the original graph.

D. Hyperbolic Geometric Priori of Graph

The Popularity-Similarity Optimization (PSO) model (Papadopoulos et al., 2012) is a generative network model to describe how random geometric graphs grow in hyperbolic Spaces to optimize the trade-off between node prevalence (abstracts by radial coordinates) and similarity (expressed by angular coordinate distances), which exhibits many common structural and dynamic features of realistic networks, such as clustering, small-worldness, scale-freeness and rich-clubness. It uses the node's birth time $t = 1, 2, \cdots, T$ to proxy popularity, giving preference to older nodes that are more likely to become popular and attract connections, which is similar to Key Opinion Leaders (KOLs) in social networks The angular distances between nodes denote their similarity distances by using cosine similarity or any other measure. The node t can be represented as polar coordinates (r_t, θ_t), and the objective is to establish new connections while optimizing the product between popularity and similarity. To simulate a realistic graph with scale-freeness (power-law degree distribution), let radial coordinate r_t of the new node t have $r_t = \ln t$, indicating that the node t prefers to connect popular nodes. In hyperbolic geometric space, the new node distance can represent a convenient single-metric by using a combination of the two geometric attractiveness priors, *radial popularity* and *angular similarity*.



(a) Wrapped Gaussian distributions in hyperbolic space.



(b) Angular distributions in hyperbolic space.

Figure C.2. (a) The Wrapped Gaussian distribution is commonly used to generate noise in hyperbolic space. As the μ approaches the edge of the Poincaré disk, the probability density at the edge of the disk increases, while the probability density at the center becomes sparse. It indicates a geometric interpretation for the disk radius, that there are always fewer nodes with popularity. (b) In hyperbolic space, the sectors of the disk are closely related to the community structure in the graph, and noise that follows a mixed Gaussian distribution is more likely to generate structures with communities. Therefore, angular constraints in the diffusion process are crucial for preserving community structure information.

D.1. Radial Popularity Coordinates.

In the polar coordinates of hyperbolic space, the radial coordinates of the nodes determine the distortion of the Gaussian distribution. As shown in Figure C.2. (a), the probability density of the central region is lower, while the probability density of the edge region is higher. It intuitively follows the laws of physics, the phenomenon of popularity is consistently governed by the rich club effects. We follow existing hyperbolic methods (Mathieu et al., 2019; Grattarola et al., 2019; Nagano et al., 2019) and use the *wrapped normal distribution* to constructive Gaussian processes in hyperbolic spaces.

D.2. Angular Similarity Coordinates.

The angular similarity measures the local connectivity of nodes and is closely related to the community structure of graphs. As shown in Figure C.2. (b), some existing works (Muscoloni & Cannistraci, 2018) couples the latent hyperbolic network geometry to this geometry to generate networks with strong clustering, scale-free degree distribution and a non-trivial community structure. Note that our method does not independently introduce noise to the angular similarity metric, but rather incorporates angular geometric constraints into a Gaussian random process to achieve a similar anisotropic effect.

E. Final Probability Density

First, we derive the form of the probability distribution of the forward diffusion process.

Definition: The folded normal distribution. If the probability distribution of Y follows the Gaussian distribution, with $Y \sim N\left(\mu, \sigma^2\right)$. Thus, $X = |Y|$, satisfies $X \sim FN\left(\mu, \sigma^2\right)$, where $FN\left(\mu, \sigma^2\right)$ denotes the folded normal distribution with mean μ and variance σ. The density of X is given by

$$f\left(x\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\left[e^{-\frac{1}{2\sigma^2}(x-\mu)^2} + e^{-\frac{1}{2\sigma^2}(x+\mu)^2}\right] \tag{C.30}$$

The density can be written in a more attractive form

$$f\left(x\right) = \sqrt{\frac{2}{\pi\sigma^2}}e^{-\frac{\left(x^2+\mu^2\right)^2}{2\sigma^2}}\cosh\left(\frac{\mu x}{\sigma^2}\right). \tag{C.31}$$

Specifically, when $\mu = 0$, the density can be represented by

$$f\left(x\right) = \sqrt{\frac{2}{\pi\sigma^2}}e^{-\frac{1}{2\sigma^2}x^2}. \tag{C.32}$$

which is also named the half-normal distribution.

Definition The random variable X obeys the probability distribution $\mathcal{N}_f\left(\mu, \sigma\right)$ if and only if the density of X can be given by

$$f(x) = \begin{cases} \sqrt{\frac{2}{\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}, & x \geq \mu \\ 0, & x < \mu \end{cases}. \tag{C.33}$$

Now we can prove the Theorem 3.2.

Proof. The angle-constrained noise z in the forward diffusion process is given by Eq (7). For the convenience of the later derivation, it can be assumed that $\operatorname{sgn}\left(\operatorname{logmap}_{\mathbf{o}}^c\left(\mathbf{h}_m\right)\right) = 1$.Thus, according to the definition of the folded normal distribution, it follows that:

$$z \sim FN\left(0, I\right), \tag{C.34}$$

Similarly, it can be easily obtained from Eq (8) and Eq (C.34) that the density of x_t in the diffusion process can be written by:

$$f(x_t) = \begin{cases} \sqrt{\frac{2}{\pi\sigma_t^2}}e^{-\frac{(x-\mu_t x_0)^2}{2\sigma_t^2}}, & x \geq \mu_t x_0 \\ 0, & x < \mu \end{cases}. \tag{C.35}$$

where $\mu_t = \sqrt{\overline{\alpha_t}} + \delta\tanh[\sqrt{c}\lambda_o^c(t)/T_0]$, and $\sigma_t = \left(1 - \overline{\alpha_t}\right)I$.

Thus, the probability density distribution of x_t satisfies:

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}_f\left(\mu_t, \sigma_t\right). \tag{C.36}$$

$$\lim_{t\to\infty}\mathbf{x}_t \sim \mathcal{N}_f\left(\delta\mathbf{x}_0, I\right). \tag{C.37}$$

□

F. Implementation Details of Diffusion Processing

F.1. Loss Prove

In DDPM, the loss function of the training network is derived from the following equation:

$$\mathbb{E}_q\left[D_{\mathrm{KL}}(p(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t>1}D_{\mathrm{KL}}(p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]. \tag{C.38}$$

Proof. To re-prove the validity of the loss function, it is first necessary to derive the probability distribution of the denoising process according to the Bayesian formulation:

$$\begin{aligned}
p\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0\right) &= \frac{p\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0\right)p\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0\right)}{p\left(\boldsymbol{x}_t|\boldsymbol{x}_0\right)} \\
&\propto exp[-\frac{(x_{t-1}-\mu_{t-1}x_0)^2}{2(1-\overline{\alpha}_{t-1})}-\frac{(x_t-(\sqrt{\alpha_t}x_{t-1}+\beta_t x_0)]^2}{4(1-\alpha_t)}+\frac{(x_t-\mu_t x_0)^2}{2(1-\overline{\alpha}_t)}] \\
&= exp\left\{-\frac{1}{2}[\frac{(x_{t-1}-\mu_{t-1}x_0)^2}{(1-\overline{\alpha}_{t-1})}+\frac{(x_t-(\sqrt{\alpha_t}x_{t-1}+\beta_t x_0)]^2}{2(1-\alpha_t)}-\frac{(x_t-\mu_t x_0)^2}{(1-\overline{\alpha}_t)}]\right\} \\
&= exp\left\{-\frac{1}{2}[(\frac{1}{1-\overline{\alpha}_{t-1}}+\frac{\alpha_t}{2(1-\alpha_t)})x_{t-1}^2-(\frac{2\mu_{t-1}x_0}{1-\overline{\alpha}_{t-1}}+\frac{2\sqrt{\alpha_t}(x_t-\beta_t x_0)}{(1-\alpha_t)})x_{t-1}+C(x_t,x_0)]\right\} \\
&\propto \mathcal{N}_f\left(\mu_q,\sigma_q\right).
\end{aligned} \tag{C.39}$$

where μ_q is given by:

$$\mu_q=\frac{2[\mu_{t-1}(1-\alpha_t)x_0+(1-\overline{\alpha}_{t-1})\sqrt{\alpha_t}(x_t-\beta_t x_0)]}{2-\alpha_t-\overline{\alpha}_t} \tag{C.40}$$

Based on the above derivation, our goal translates to:

$$\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) \\
&=\arg\min_\theta D_{\mathrm{KL}}(\mathcal{N}_f(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_q,\boldsymbol{\Sigma}_q(t))\parallel\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_{\boldsymbol{\theta}},\boldsymbol{\Sigma}_q(t))) \\
&=\arg\min_\theta\int_0^\infty\sqrt{\frac{2}{\pi\sigma_q^2}}\left[e^{-\frac{1}{2\sigma_q^2}(x-\mu_q)^2}\right]\log\frac{\sqrt{\frac{2}{\pi\sigma_q^2}}\left[e^{-\frac{1}{2\sigma^2}(x-\mu_q)^2}\right]}{\sqrt{\frac{2}{\pi\sigma^2}}e^{-\frac{1}{2\sigma^2}(x-\mu_\theta)^2}}dx \\
&=\arg\min_\theta\int_0^\infty\sqrt{\frac{2}{\pi\sigma_q^2}}\left[e^{-\frac{1}{2\sigma_q^2}(x-\mu_q)^2}\right][-\frac{1}{2\sigma^2}(x-\mu_q)^2+\frac{1}{2\sigma^2}(x-\mu_\theta)^2]log_2 edx \\
&=\arg\min_\theta\int_0^\infty\left|(x-\mu_q)^2-|(x-\mu_\theta)^2\right|dx \\
&=\arg\min_\theta\left|(\mu_\theta-\mu_q)^2\right|
\end{aligned} \tag{C.41}$$

Thus after a derivation similar to that of DDPM, we can define the loss function as:

$$\mathcal{L}=\mathbb{E}\left\|f_\theta\left(X_t,A,t\right)-X_0\right\|^2. \tag{C.42}$$

□

F.2. Sample

As demonstrated in Theorem 3.1, the individual tangent planes can be viewed as the tangent planes of a hybrid surface. The samples can also be sampled directly in the same tangent plane and denoised uniformly. Sampling from the target distribution is difficult, but sampling from its construction process is easy. We just need to compute an additional direction matrix. The direction transformation matrix is given by:

$$T=\mathrm{sgn}\left(\mathrm{logmap}_{\mathbf{o}}^c\left(\mathbf{h}_{m_1}\right),\mathrm{logmap}_{\mathbf{o}}^c\left(\mathbf{h}_{m_2}\right),...,\mathrm{logmap}_{\mathbf{o}}^c\left(\mathbf{h}_{m_i}\right)\right) \tag{C.43}$$

F.3. Reverse Denosing Process

In this section, we follow the same denoising process as in DDM (Yang et al., 2023) with the following algorithm. Specifically, the denoising process is given by:

$$\begin{aligned}
\hat{x}_0 &= R(x_s,s) \\
x_{s-1} &= x_s-D(\hat{x}_0,s)+D(\hat{x}_0,s-1)
\end{aligned} \tag{C.44}$$

Algorithm 2 Sampling from HypDiff

```
**Input:** Graph $\mathcal{G} = \{\mathbf{A}, \mathbf{X}\}$; Number of diffusion timesteps $T$;
**Output:** The generated hyperbolic embedding coordinates $\hat{x}_{\mathbb{H}}$.
sample $x_T$ from the folded normal distribution $\leftarrow$ Eq. (C.43)
**for** $t = T$ **to** $0$ **do**
    predict $\hat{x}_0$ with $\hat{x}_t$,$t$ and $A$;
    de-noise and predict $\hat{x}_{t-1} \leftarrow$ Eq. (C.44);
**end for**
mapping the generated $\hat{x}_0$ to the Poincaré embedding $\hat{x}_{\mathbb{H}}$
```

where R is the restoration operator, D is the addition noise operator, s represents the time steps.

Next, we prove that this algorithm still satisfies our diffusion process:

$$z(x_t, t) = \frac{x_t - (\sqrt{\alpha_t} + \delta \tanh[\sqrt{c}\lambda_o^c(t)/T_0])\hat{x_0}}{\sqrt{1-\alpha_t}}, \tag{C.45}$$

$$D(\hat{x_0}, t) = (\sqrt{\alpha_t} + \delta \tanh[\sqrt{c}\lambda_o^c(t)/T_0])\hat{x_0} + \sqrt{1-\alpha_t}\hat{z}, \tag{C.46}$$

Thus, the denoising process can obtain x_{t-1}:

$$\begin{aligned} x_{t-1} =& x_t - D(\hat{x_0}, t) + D(\hat{x_0}, t-1) \\ =& x_t - [(\sqrt{\alpha_t} + \delta \tanh[\sqrt{c}\lambda_o^c(t)/T_0])\hat{x_0} + \sqrt{1-\alpha_t}\hat{z}] + [(\sqrt{\alpha_{t-1}} + \\ & \delta \tanh[\sqrt{c}\lambda_o^c(t-1)/T_0])\hat{x_0} + \sqrt{1-\alpha_{t-1}}\hat{z}] \\ =& (\sqrt{\alpha_{t-1}} + \delta \tanh[\sqrt{c}\lambda_o^c(t-1)/T_0])\hat{x_0} + \sqrt{1-\alpha_{t-1}}\hat{z} \end{aligned} \tag{C.47}$$

The complete denoising process algorithm is described in Algorithm 2.

G. Complexity Analysis

In this part, we will analyze the time complexity and space complexity of our algorithm using graph set data as an example. $G = (X, E)$ is one of the graphs set G^s, where X is the n-dimensional node eigenvector and E is the $m * m$-dimensional adjacency matrix of the graph. s is the number of graphs in the graph set G^s.

G.1. Time Complexity

Hyperbolic embedding and clustering: First, we need to embed each graph G into a hyperbolic space by HGCN. Let the graph G be embedded after passing through HGCN into the $m * d$-dimensional vector H^m. The time complexity of this process can be viewed as $O(md) * 1(t)$, where $1(t)$ denotes the time through the neural network. The process of clustering can be approximated with a time complexity of $O(kmd)$, where k denotes the number of clusters. Both the embedding process and the clustering process occupy a very short time compared to the forward diffusion and training denoising network process.

Diffusion-forward process: For the forward diffusion process, we need to adjust the noise direction according to the north pole to the center of the mass vector direction. Since calculating the direction transfer matrix can be prepared in advance before training, the complexity of this part is $O(md)$. The rest of the process is the same as the normal diffusion process, and it is sufficient to do the noise addition process once. The time complexity of this part is $O(md)$.

Training of denoising networks: The training of denoising networks is essentially the same as other diffusion models and does not require additional computing time. So, the time complexity of this part is $O(md) * 1(t)$

Overall, the time complexity of the diffusion process is $O(1(t) * 2md) + O((k + 2)md)$ in one epoch.

Generation: The main difference between our generation process and other methods is in our noise sampling. Since we need to sample in different tangent planes in hyperbolic space, which requires each time to judge the tangent plane where the corresponding sampling point is located, the time complexity of this process is O(m). Compared to the subsequent

Table C.2. Statistics of real-world datasets.

	Dataset	#Nodes	#Edges	#Features	#Avg. Degree	#Class
Link P.	**Cora**	2,708	5,429	1,433	3.90	7
	Citeseer	3,312	4,732	3,703	2.79	6
	Polblogs	1,490	19,025	500	25.54	3
	Dataset	**#Graphs**	**#Avg. Node**	**#Avg. Edge**	**#Max Num Node**	**#Class**
Graph G.	**MUTAG**	188	17.9	39.6	28	2
	IMDB-B	1,000	19.8	193.1	136	2
	PROTEINS	1,113	39.1	145.6	620	2
	COLLAB	5,000	74.5	4914.4	492	3

stepwise denoising process, the sampling process takes very little time, which means that we use almost the same amount of time as the other methods for generation.

G.2. Space Complexity

It is worth noting that compared to other graph diffusion models, our model has a significant advantage over other models due to its space complexity, which indirectly leads to a shorter training time than other models. Specifically, in other graph diffusion models, such as GDSS and Digress, since it is directly for the sparse adjacency matrix diffusion, then as to the graph set G^s, which is equivalent to the diffusion scale of $O(s \times m^2 \times d)$, the number of parameters that need to be trained for the denoising network is also namely huge.

However, in our approach, since we embed the graphs in hyperbolic space, each graph is represented as a $m * d$-dimensional vector in the hyperbolic space, which means that our diffusion scale is $O(sm^2d)$, and the corresponding number of parameters in the training network is much smaller than in other approaches (For large-scale graphs, m is much larger than d). This suggests that our approach has significant advantages in both time and space.

H. Datesets Description

We show the specific properties of the dataset in Table C.2.

I. Additional Results

In this section, we record the results for the datasets beyond the in-text presentations.

I.1. F1 Scores of Graph Generation Task

The results of the F1 pr and F1 dc metrics (Naeem et al., 2020) for the graph generation task are presented in Table C.3, and we obtain conclusions consistent with those described above where larger F1 means the model has better fidelity and diversity. Note that F1 pr is the harmonic mean of improved precision and recall and F1 dc is the harmonic mean of density and coverage. They are sensitive to detecting mode collapse and mode dropping.

Table C.3. Generation results about the F1 score of precision-recall and density-coverage (F1 pr and F1 dc) between the original and generated graphs.

Method	**Synthetic Datasets**								**Real-world Datasets**							
	Community		**BA-G**		**Ego**		**Grid**		**MUTAG**		**PROTEINS**		**IMDB-B**		**COLLAB**	
	F1 pr	F1 dc	F1 pr	F1 dc	F1 pr	F1 dc	F1 pr	F1 dc	F1 pr	F1 dc	F1 pr	F1 dc	F1 pr	F1 dc	F1 pr	F1 dc
VGAE	0.001	0.023	0.104	0.002	0.162	0.139	0.091	0.080	0.173	0.072	0.064	0.204	0.031	0.165	0.283	0.173
GraphRNN	0.732	**1.312**	0.016	0.003	0.008	0.186	0.333	0.109	0.643	0.334	0.804	0.842	0.475	0.575	0.260	0.295
GDSS	0.245	0.157	0.020	0.132	0.201	0.093	0.496	0.260	0.585	0.428	0.763	0.730	0.048	0.073	0.019	0.005
DiGress	0.645	0.695	0.724	0.783	0.409	0.091	0.761	**0.742**	0.713	0.549	0.867	0.193	0.546	0.358	0.918	**1.179**
GraphGDP	0.815	0.960	0.157	0.060	0.933	**0.997**	**0.815**	0.627	**0.880**	**0.838**	0.921	0.931	0.027	0.015	**0.989**	1.016
EDGE	**0.989**	1.207	0.920	0.864	0.609	0.770	0.671	0.287	0.667	0.679	0.951	**0.974**	**0.959**	**0.995**	0.933	1.015
HypDiff	0.812	0.864	**0.971**	**0.923**	**0.943**	0.734	0.792	0.641	0.730	0.680	**0.973**	0.943	0.730	0.670	0.981	0.653

I.2. Visualization

The visualization of HypDiff and GDSS by five well-accepted graph theoretical models is shown in Figure C.3 These graphs can represent typical general and complex topological properties: **BA** (Barabási & Albert, 1999) scale-free graphs have more tree structure. **SBM** (Holland et al., 1983) graphs have more community structure. **WS** (Watts & Strogatz, 1998) small-world graphs have more cyclic and clique structure. **Hierarchical Fractal** (Ravasz & Barabási, 2003) networks is constructed by self-organization and self-similarity. **Grid** (Ma et al., 2018) have regular (Euclidean) structures. While comparing the generated structure, we also color the node by using its degree. The results demonstrate that our HypDiff exhibits significantly enhanced proficiency in reproducing the original graph structure across BA, SBM, WS, and Fractal graphs, while consistently achieving a coherent distribution of node colors. Regarding the Grid, due to the weakness of hyperbolic space for capturing regular structure, HypDiff still generates nodes with high degrees (the red color distribution is uneven).

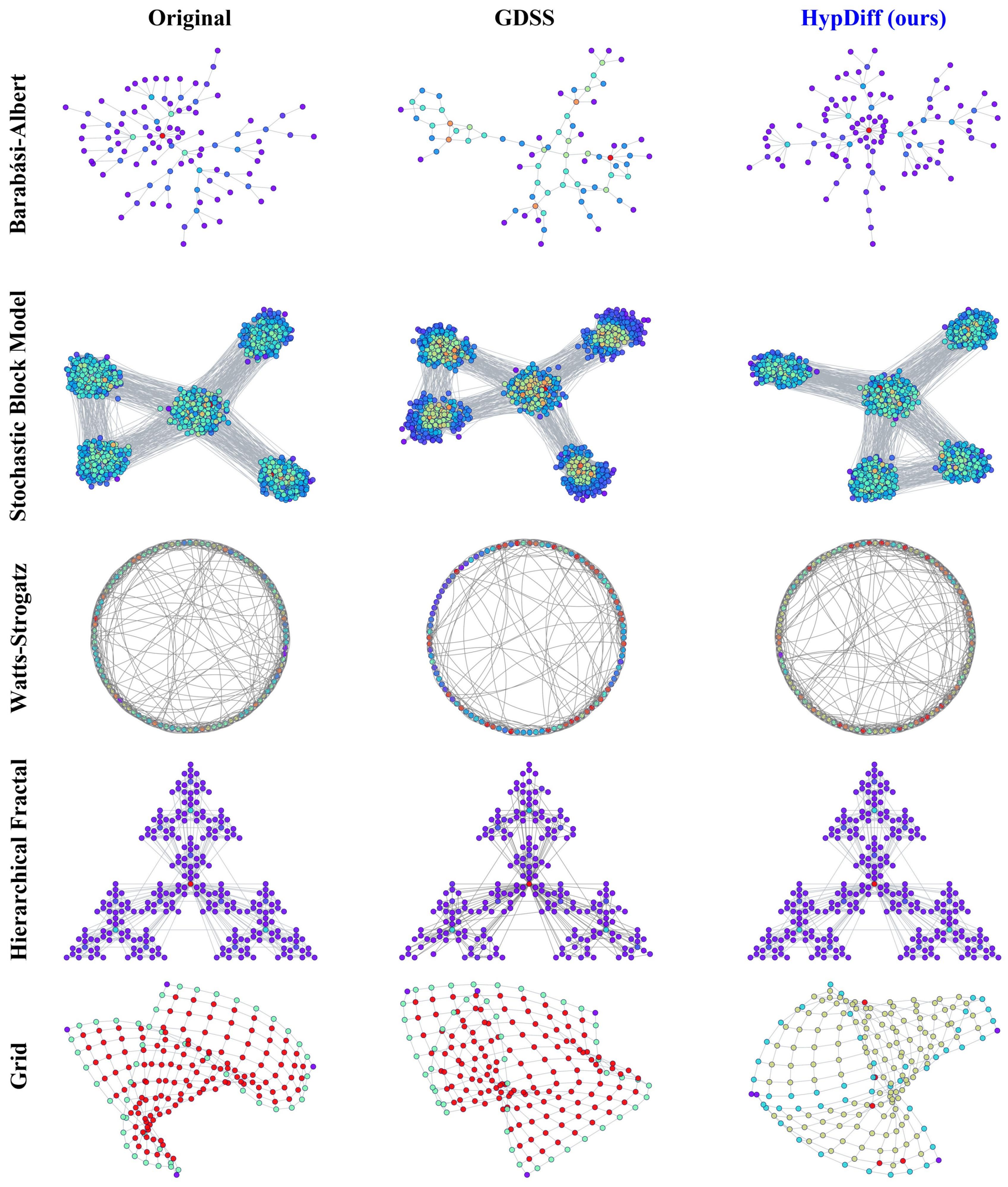


Figure C.3. Visualization of graph generations.

J. Analysis of GPU Occupancy and Runtime

We conducted a statistical analysis of GPU utilization during the model training and the average denoising time over 1000 steps during diffusion. The results, presented in Table C.4, indicate that our model performs comparably in terms of time to other models, yet it utilizes significantly less GPU. This implies that our model exhibits higher efficiency.

Table C.4. GPU memory usage during training and time used to denoise 1000 steps during generation.

Method	**Synthetic Datasets**						**Real-world Datasets**					
	Community		**BA-G**		**Ego**		**MUTAG**		**PROTEINS**		**IMDB-B**	
	Time(s)	GPU(MB)	Time(s)	GPU(MB)	Time(s)	GPU(MB)	Time(s)	GPU(MB)	Time(s)	GPU(MB)	Time(s)	GPU(MB)
GDSS	10.14	3475	11.80	7750	9.71	3883	10.79	3907	12.04	6305	12.5	3501
DiGress	9.62	3936	11.42	9012	12.28	4174	9.84	4125	11.74	6975	12.1	5800
GraphGDP	12.58	3802	14.36	13164	12.47	3848	12.85	3956	12.18	44708	13.6	5902
EDGE	9.87	2825	11.83	25236	10.24	2657	9.73	27603	10.95	26188	11.8	6205
HypDiff	10.03	2246	12.04	5697	10.15	2570	9.92	2720	10.72	4735	11.20	2519