HypBench: Hyperbolic Benchmark for Graph Neural Network Performance

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Graph Neural Networks (GNNs) have excelled in predicting graph properties in various applications ranging from identifying trends in social networks to drug discovery and malware detection. With the abundance of new architectures and increased complexity, GNNs are becoming highly specialized when tested on a few well-known datasets. However, how the performance of GNNs depends on the topological and features properties of graphs is still an open question. In this work, we introduce a comprehensive benchmarking framework for graph machine learning, called **HypBench**, focusing on the performance of GNNs across varied network structures. Utilizing the geometric soft configuration model in hyperbolic space, we generate synthetic networks with realistic topological properties and node feature vectors. This approach enables us to assess the impact of network properties, such as topology-feature correlation, degree distributions, local density of triangles, and homophily, on the effectiveness of different GNN architectures. Our results highlight the dependency of model performance on the interplay between network structure and node features, providing insights for model selection in various scenarios. This study contributes to the field by offering a versatile tool for evaluating GNNs, thereby assisting in developing and selecting suitable models based on specific data characteristics.

1 Introduction

Graph Neural Networks (GNNs), an extension of Convolutional Neural Networks tailored for graph-structured data [7–10] employ recursive message passing between nodes and their neighbors at varying hop distances. These models leverage both the topological information encoded in the graph structure and node-specific features to effectively map each node in the graph into a learnable embedding space. GNNs have evolved to encompass a wide variety of architectures and tasks, ranging from node and graph classifications to link prediction. Despite this growing interest in the development of GNNs, the fundamental issue of homogeneity in benchmarking datasets persists in GNN research, making it challenging to determine the most suitable GNN model for unseen datasets. In addition, since GNNs are data-driven models tailored to specific tasks, there is a potential concern of overfitting new architectures to given datasets, especially when the data have similar structural properties [11]. Thus, a fair comparison between different models in reproducible settings is required.

In this work, we propose a comprehensive benchmarking scheme for graph neural networks, which we call **HypBench**. Utilizing a hyperbolic soft configuration network model with features [12], henceforth referred as a *geometric network model*, we generate synthetic networks with realistic topological properties where node features can be correlated with the network topology. This highly flexible model allows us to evaluate GNNs in depth across various scenarios. Moreover, we suggest using the benchmark as a tool for optimal model selection by analyzing the inherent properties of the real dataset. Although the use of hyperbolic geometry might appear superfluous, it has been

Table 1: Comparison of HypBench and other synthetic network generators. The ✓indicates control over a given property, ✗ indicates lack of control, and $\propto$ indicates indirect control. The $P(k)$ is the degree distribution, $\bar{c}$ is the clustering coefficient, $\langle k \rangle$ is the average degree. Whereas $P(k_n)$ and $P(k_f)$ are the degree distributions of nodes and features, respectively, $\langle k_n \rangle$ and $\langle k_f \rangle$ are the average degrees of nodes and features, respectively, and $\mathcal{H}$ represents homophily.

	$P(k)$, Power-law	$\bar{c}$	$\langle k \rangle$	$P(k_n), P(k_f)$	$\langle k_n \rangle, \langle k_f \rangle$	Topology-Feature Correlation	$\mathcal{H}$	Underlaying mechanism/ Connectivity law
HypBench	✓, ✓	✓	✓	✓,✓	✓,✓	✓	✓	geometric
GraphWorld (SBM) [11]	✓, Quasi power-law	$\propto$	✓	✗, normal	✗, ✓	$\propto$	✓	probability matrix
GraphWorld (CABAM) [19]	✗, ✓	$\propto$	✓	✗, normal	✗, ✓	$\propto$	✓	preferential attachement
GraphWorld (LFR) [19]	✓,✓	$\propto$	✓	✗, normal	✗, ✓	$\propto$	✓	class-aware configuration model
GenCAT [20, 21]	✓, ✓	$\propto$	✓	✗, (normal, Bernoulli)	✗, ✓	$\propto$	✓	latent variables
FastSGG [22]	✓, ✓	$\propto$	✓	✗ , ✗	✗, ✗	✗	✓	preferential attachment
Darabi et. al. [23]	✓,✓	$\propto$	✓	✓,✓	✓,✓	$\propto$	✓	generative model
SkyMap [24]	✓,✓	$\propto$	✓	✓,✓	✓,✓	$\propto$	✓	generative model

demonstrated that it is the simplest method for generating geometric random graphs that uniquely combine several key characteristics: They have power-law degree distributions, exhibit small-world properties, and are highly clustered, meaning they have a high density of triangles [13]. These traits closely mirror those observed in real complex networks.

We aim to show the crucial factors, including the network's structural properties and the degrees of correlation between nodes and features—controlled by the framework parameters—that influence the performance of graph machine learning models. Employing the proposed benchmark, we systematically compare the performance of well-known GNNs and a graph transformer. We also evaluate models that solely utilize node features. Our study aims to evaluate machine learning models in two fundamental graph-based tasks: node classification (NC) and link prediction (LP). Here, we highlight the main contributions of our empirical study, which provides insights into the suitability of the various models under different network conditions. It will, thus, benefit applications and the community focused on developing new GNN algorithms.

- Our framework generates benchmark networks with tunable levels of topology-feature correlation, homophily, clustering coefficient, degree distributions, and average degrees. This approach covers the most important properties of a wide range of real datasets, providing a comprehensive tool for their analysis. The code and the datasets will be publicly available at `https://github.com/networkgeometry/hyperbolic-benchmark-gnn` under the MIT License. We also created a class to generate the synthetic network directly using the PyTorch Geometric library. We have also provided a short tutorial on how to load the synthetic network, visualize it, and run prediction tasks on a simple GCN model `https://github.com/networkgeometry/hyperbolic-benchmark-gnn/tree/main/pytorch_geometric_example`.
- The stronger the correlation between the network topology and node features, the better GNNs and feature-based models perform in NC and LP.
- The hyperbolic-based models, specifically HGCN and HNN achieve the highest AUC scores [18] in LP task. Remarkably, HNN, despite being solely a feature-based method, outperforms traditional graph-based models across various parameters.
- In the NC task, where no information about the graph data is available, emphasis should be placed on model interpretability and time complexity. This is crucial since the accuracy of graph-based models tends to be uniformly high, making these factors significant differentiators.

2 Related work

With the continuous evolution of graph machine learning, there is a growing necessity to comprehend and evaluate the performance of GNN architectures. In this respect, benchmarking can provide a fair and standardized way to compare different models. The Open Graph Benchmark (OGB) [25] stands as a versatile tool to assess the performance of the GNNs. Yet, its emphasis on a limited range of actual networks indicates that it does not encompass all network characteristics and falls short in terms of parameter manipulation. Consequently, this highlights the necessity for creating benchmarking tools based on synthetic data. Such tools would allow for the assessment of GNNs in a controlled environment and across a more extensive array of network properties [22, 20, 21]. One of them is GraphWorld [11], which is a synthetic network generator utilizing the Stochastic Block Model (SBM) to generate graphs with communities. It employs a parametrized community distribution and an edge probability matrix to randomly assign nodes to clusters and establish connections. Node features are also generated using within-cluster multivariate normal distributions. A fixed edge probability matrix in SBM prevents GraphWorld from faithfully replicating a predefined degree sequence and generating graphs with true power-law distributions. To overcome this limitation, Ref. [19] integrates Graphworld with two other generators: Lancichinetti-Fortunato-Radicchi (LFR) [26] and CABAM (Class-Assortative graphs via the Barabási-Albert Model) [27]. This integration broadens the coverage of the graph space, specifically for the NC task. Darabi et al. [23] decompose synthetic graph generation into three modules: structure generation via a generalized stochastic Kronecker product, feature synthesis using deep generative models, and a feature-structure aligner. This modular approach allows for flexible parameter manipulation and scaling to very large graphs. Similarly, SkyMap [24] introduces a generative model tailored for GNN benchmarking that leverages mixing and class-feature matrices to precisely control graph topology and node attributes. In this paper, we propose an alternative network generators: a framework based on the geometric soft configuration model. This model's underlying geometry straightforwardly couples the network topology with node features and labels. This capability enables independent control over the clustering coefficient in both the unipartite network of nodes and the bipartite network of nodes and features, irrespective of the degree distributions of nodes and features (see Fig. S1 in the Supplementary Materials). Table 1 presents a comparison between HypBench and several state-of-the-art benchmarking frameworks, highlighting the properties each can control.

3 Geometric Network Model

Graph-structured data are characterized by group of N_n nodes that create a complex network $\mathcal{G}_n$, along with a collection of N_f features linked to these nodes. Typically, these features are converted into binary form. Therefore, for any given node i, its feature set is depicted as a vector $\vec{f_i} \in \{0,1\}^{N_f}$. This vector has elements that are either zero or one, depending on whether a specific feature is present or absent. An alternative approach considers that nodes and features form a bipartite graph, $\mathcal{G}_{n,f}$, with nodes defining one of the types of elements of the graph and features the other [12]. Within this approach, the full information of the data is encoded into the two networks $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$. Interestingly, in [12], the topological properties of the bipartite graph $\mathcal{G}_{n,f}$ of real graph-structured datasets were first studied, showing a rich and complex topological organization. A remarkable finding in [12, 28] is the detection of strong correlations between the graphs $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$ within real datasets. This suggested the possibility to describe $\mathcal{G}_{n,f}$ as a geometric random graph in the same hyperbolic space used to describe the graph $\mathcal{G}_n$ so that the shared metric space would mediate the correlation between them. Building on this concept, the study in [12] introduces a generative model that produces networks $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$, with experimental results demonstrating their ability to accurately reproduce key topological properties observed in real datasets, including degree distribution, clustering coefficient, and average nearest neighbors degree function (see Fig. A.2.1 in Appendix B and Fig. S2 in the Supplementary Materials). We will now provide a detailed description of this model, which is used to create synthetic datasets for evaluating the performance of GNNs. A summary of the procedure is provided in Algorithm 1 in Appendix C.

3.1 The $\mathbb{S}^1/\mathbb{H}^2$ model

The network $\mathcal{G}_n$ is modeled by the $\mathbb{S}^1/\mathbb{H}^2$ model [29–31]. This geometric soft configuration model produces synthetic networks with realistic topological properties, including arbitrary degree distributions [29, 30, 32], the small-world property [33–35], self-similarity [29], and high clustering coefficient [36, 30, 32, 37, 38], to name just a few. See also [13] and references therein.

In the $\mathbb{S}^1$ model a node is endowed with a hidden degree κ, representing its popularity, influence or importance. Hidden (or expected) degrees are distributed by an arbitrary probability density $\rho(\kappa)$, with $\kappa \in (\kappa_0, \infty)$. In this way, the model has the flexibility to reproduce a variety of degree distributions. Each node is also assigned with an angular position θ in the similarity space, represented as a one-dimensional sphere [1]. Then, pairs of nodes are connected with a probability function taking

[1]The model with arbitrary dimensions has been defined in [39].

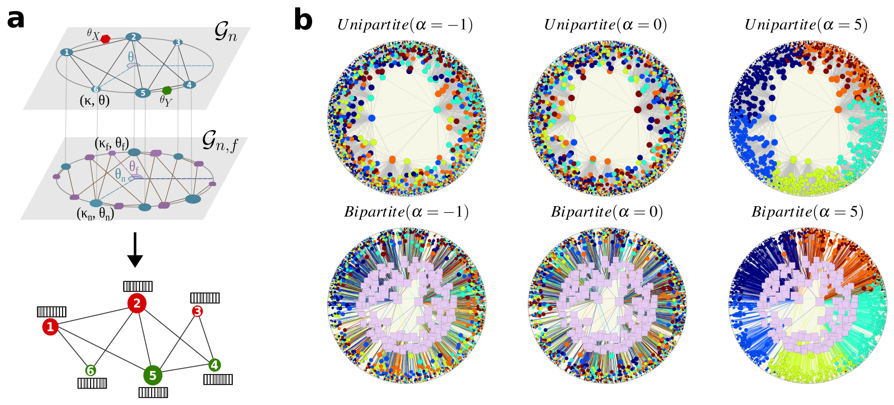

Figure 1: (a) Representation of HypBench to generate rich graph-structured data. (b) Hyperbolic representation of a synthetic network with $N_n = 2000$ nodes represented as circles, where colors indicate their labels, and $N_f = 200$ features depicted as purple squares. The size of the symbols is proportional to the logarithms of the degrees of nodes and features. The parameters for the $\mathbb{S}^1$ model are $\beta = 3$, $\gamma = 3.5$, and $\langle k \rangle = 30$, and for the bipartite-$\mathbb{S}^1$ model, $\beta_b = 3$, $\gamma_n = 3.5$, $\gamma_f = 2.1$, and $\langle k_n \rangle = 3$. Only edges with an effective distance $\chi < 1$ are depicted in the figure.

the form of a gravity law, balancing the interplay between node angular distances and their hidden degrees:

$$p(\kappa, \kappa', \Delta\theta) = \frac{1}{1+\chi^\beta} \quad \text{with} \quad \chi \equiv \frac{R\Delta\theta}{\mu\kappa\kappa'}, \tag{1}$$

where $\Delta\theta = \pi - |\pi - |\theta - \theta'||$ is the angular separation between the nodes in the similarity space, $R = N_n/2\pi$ denotes the radius of the similarity space, $\beta > 1$ governs the level of clustering coefficient, and

$$\mu = \frac{\beta}{2\pi\langle k \rangle} \sin\frac{\pi}{\beta} \tag{2}$$

controls the average degree of the network $\langle k \rangle$. Notice that the radius is chosen such that in the limit $N_n \gg 1$ the curvature of the similarity space vanishes and the process converges to a Poisson point process on $\mathbb{R}$ of density one. With these choices, the expected degree of a node with hidden degree κ is $\langle k(\kappa) \rangle = \kappa$, so that by controlling the distribution of hidden degrees, we can adjust the resulting degree distribution.

Interestingly, the $\mathbb{S}^1$ model exhibit an isomorphism with a purely geometric model in hyperbolic space, referred to as the $\mathbb{H}^2$ model [30]. Hyperbolic space is a homogeneous and isotropic manifold with constant negative curvature. In two dimensions, and in its native representation, each point in $\mathbb{H}^2$ can be described in polar coordinates (r, θ). However, the distance between two points cannot be measured using the standard Euclidean metric. Instead, the hyperbolic distance x between two points with radial coordinates r and r' and separated by an angular distance $\Delta\theta$ is computed via the hyperbolic law of cosines

$$\cosh x = \cosh r \cosh r' - \sinh r \sinh r' \cos\Delta\theta, \tag{3}$$

where we have set the curvature to -1. Returning to $\mathbb{S}^1$, the isomorphism with hyperbolic space is realized by mapping nodes' hidden degrees to radial positions within a disk of radius $R_{\mathbb{H}^2}$ in the hyperbolic plane as

$$r = R_{\mathbb{H}^2} - 2\ln\frac{\kappa}{\kappa_0}, \quad \kappa \geq \kappa_0, \tag{4}$$

where the radius in $\mathbb{H}^2$ is given by $R_{\mathbb{H}^2} = 2\ln\frac{2R}{\mu\kappa_0^2}$. Low degree nodes with hidden degree κ_0 are mapped at the boundary of the hyperbolic disk, whereas high degree nodes get located close to the center of the disk. After this mapping, the connection probability in Eq. (1) becomes

$$p(x) = \frac{1}{1+e^{\frac{\beta}{2}(x-R_{\mathbb{H}^2})}} \text{, with } \; x = r + r' + 2\ln\frac{\Delta\theta}{2} \tag{5}$$

where x approximates the hyperbolic distance between two nodes at radial coordinates r and r' with angular separation of $\Delta\theta$ [30, 31]. Thus, in this representation, the probability of connections is just a function of the hyperbolic distance. Interestingly, this mapping compresses two key aspects into a single metric: node popularity–reflected in their degrees–and the similarity space in which they reside, captured by their angular coordinates. Hyperbolic models are compelling because they naturally reproduce many topological features seen in real-world complex networks, including heterogeneous degree distributions, the small-world property, and high clustering. Moreover, when bounded within a finite region, the hyperbolic plane is intrinsically hierarchical, making it an intuitive framework for representing the layered organization of complex data structures.

3.2 The bipartite-$\mathbb{S}^1/\mathbb{H}^2$ model

In real-world graph-structured data, nodes' features are correlated with the topology of the graph, i.e., nodes that exhibit similarities in the network topology $\mathcal{G}_n$ also share common features [40, 12]. The bipartite-$\mathbb{S}^1$ model induces this correlation by placing both $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$ in the same similarity space.

In this model, every node is given a pair of hidden variables, (κ_n, θ_n). Here, κ_n denotes the node's expected degree in the bipartite node-feature graph, while θ_n (equal to θ) represents its angular coordinate, matching that in $\mathcal{G}_n$. In a similar fashion, features are assigned two hidden variables, (κ_f, θ_f), accounting for their expected degrees and their angular locations in the shared similarity space. The likelihood of a link forming between a node and a feature, characterized by hidden degrees κ_n and κ_f and separated by an angular distance $\Delta\theta$, is expressed as

$$p_b(\kappa_n, \kappa_f, \Delta\theta) = \frac{1}{1+\chi^{\beta_b}}, \quad \text{where} \quad \chi \equiv \frac{R\Delta\theta}{\mu_b \kappa_n \kappa_f}. \tag{6}$$

In this equation,

$$\mu_b = \frac{\beta_b}{2\pi\langle k_n\rangle} \sin\frac{\pi}{\beta_b} \tag{7}$$

is a crucial parameter that defines the average degree of nodes $\langle k_n\rangle$ and features $\langle k_f\rangle = \frac{N_n}{N_f}\langle k_n\rangle$ and β_b controls bipartite-clustering coefficient, as a measure of the coupling between the resulting topology and the underlying metric space. This model, akin to the $\mathbb{S}^1$ model, guarantees that the expected degrees of nodes and features with hidden degrees κ_n and κ_f are $\langle k_n(\kappa_n)\rangle = \kappa_n$ and $\langle k_f(\kappa_f)\rangle = \kappa_f$, respectively. The hidden variables for both nodes and features can be generated from arbitrary distributions or tailored to mimic the topology of a specific real-world network.

As in the case of the $\mathbb{S}^1$ model, nodes and features in $\mathcal{G}_{n,f}$ can be mapped into the hyperbolic disc of radius $R^b_{\mathbb{H}^2} = 2\ln\frac{2R}{\mu_b \kappa_{n,0}\kappa_{f,0}}$ similarly as in Eq. (4). This mapping leads to a connection probability between nodes and features with the same functional form as in Eq. (5), replacing β by β_b and $R_{\mathbb{H}^2}$ by $R^b_{\mathbb{H}^2}$.

3.3 Assigning labels to nodes

To complete our theoretical framework, we need to specify how labels are assigned to nodes, ensuring a correlation between node labels in $\mathcal{G}_n$ and their features. We use the underlying similarity space to induce and tune these correlations. We introduce a node labeling strategy for any number of clusters $\mathcal{N}_L$, each associated with a label, and a tunable homophily level [41–43], crucial for GNNs [44, 45]. First, the cluster centroids are randomly placed in the similarity space. The probability of node i being assigned to cluster X is given by:

$$P_X(i) = \frac{\Delta\theta_{iX}^{-\alpha}}{\sum\limits_{Y\in\mathcal{N}_L} \Delta\theta_{iY}^{-\alpha}}. \tag{8}$$

where $\Delta\theta_{iX}$ is the angular distance between node i and centroid X. The α parameter tunes homophily: a high α assigns labels based on proximity to centroids, while $\alpha = 0$ assigns labels randomly. Negative α values also induce homophily, but nodes are labeled by the farthest centroid.

Fig. 1 (a) illustrates the generation of graph-structured data with N_n nodes, represented as blue circles, and N_f features, depicted as purple rounded squares. The top part represents the unipartite network $\mathcal{G}_n$, connecting nodes generated with the $\mathbb{S}^1$ model. The positions of the cluster centroids are shown as θ_X and θ_Y. The sketch is generated with a high value of α so that nodes are assigned labels of the nearest cluster centroids. Below $\mathcal{G}_n$, we depict the bipartite-$\mathbb{S}^1$ model where nodes and features share the same similarity space. With these two models, we can generate the synthetic graphs with the class labels and the binarized feature vectors shown at the bottom.

In Fig. 1(b) we depict the hyperbolic representation of one synthetic dataset generated with our model with realistic parameters. Typically, this corresponds to highly clustered, small-world, and heterogeneous degrees in the case of $\mathcal{G}_n$. As for $\mathcal{G}_{n,f}$, typical topologies have quite homogeneous nodes' degree distribution and very heterogeneous features degree distribution [12]. Each column in Fig. 1(b) shows the two networks $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$ with different values of the parameter α. In the bipartite network, nodes with homogeneous degrees are closer to the edge of the disk, while features exhibiting heterogeneous degrees are closer to the center of the circle, connected to nodes within the same angular sector for higher value of $|\alpha|$.

4 HypBench

The HypBench framework depicted in Fig. 1 combines the $\mathbb{S}^1/\mathbb{H}^2$ and bipartite-$\mathbb{S}^1/\mathbb{H}^2$ models within a unified similarity space. Additionally, it incorporates a method for label assignment. This integration facilitates the creation of networks exhibiting a wide range of structural properties and varying degrees of correlation between nodes and their features. Specifically, our framework allows us to control the following properties:

- **Degree distributions**. The framework allows us to generate networks with arbitrary node and feature degree distributions by fixing the sets of hidden degrees in $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$. A useful choice is the power-law distribution

$$\rho(\kappa) = (\gamma-1)\kappa_0^{\gamma-1}\kappa^{-\gamma} \quad \text{with} \quad \kappa_0 = \frac{\gamma-2}{\gamma-1}\langle k\rangle \tag{9}$$

  and parameter γ controlling the heterogeneity of the degree distribution. This leaves us with three parameters γ, γ_n, γ_f fixing the hidden degree distributions in the $\mathbb{S}^1$ and bipartite-$\mathbb{S}^1$ models, respectively.
- **Network average connectivity**. Another critical factor is the network average connectivity. We have the freedom to choose the average degree of the network $\mathcal{G}_n$, $\langle k\rangle$, as well as the average degree of nodes in the bipartite node-feature network $\mathcal{G}_{n,f}$, $\langle k_n\rangle$, by changing μ and μ_b parameters in Eqs. (2) and (7). Notice that the average degree of features is fixed by the identity $\langle k_f\rangle = \frac{N_n}{N_f}\langle k_n\rangle$.
- **Clustering coefficient and topology-features correlations**. The clustering coefficient is regulated by parameters β and β_b, binding the topology of both $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$ to the common similarity space through the triangle inequality. A higher value of β implies that pairs of nodes that are close in the hyperbolic space (with small effective distance χ) are more likely to get connected in $\mathcal{G}_n$. Likewise, a high value β_b indicates that nodes and features separated by a short hyperbolic distance have a high probability to be connected in $\mathcal{G}_{n,f}$. Consequently, when $\beta > 1$ and $\beta_b > 1$ similar nodes tend to be connected and, at the same time, share common features. Thus, β and β_b not only influence the clustering coefficient in the networks but also adjust the correlation between topology and node features. We should stress that the ability to directly control the value of clustering coefficient, and thereby topology-feature correlations, is one of the key distinctive features of HypBench as compared to other benchmarking schemes.
- **Homophily**. For the task of NC, $\mathcal{N}_l$ identifies the number of classes and α governs the strength of the nodes' labels concentration around their centroids, allowing us to control the level of homophily in the network, defined as [42]

$$\mathcal{H} = \frac{1}{N_n}\sum_{i=1}^{N_n}\frac{n_l(i)}{k_i}, \tag{10}$$

  where $n_l(i)$ is the number of neighbors of node i with the same label as i and k_i its degree. A high value of $\mathcal{H}$ indicates a tendency for similar nodes to connect with each other. For detail analysis on how homophily changes in the synthetic networks see Section A.

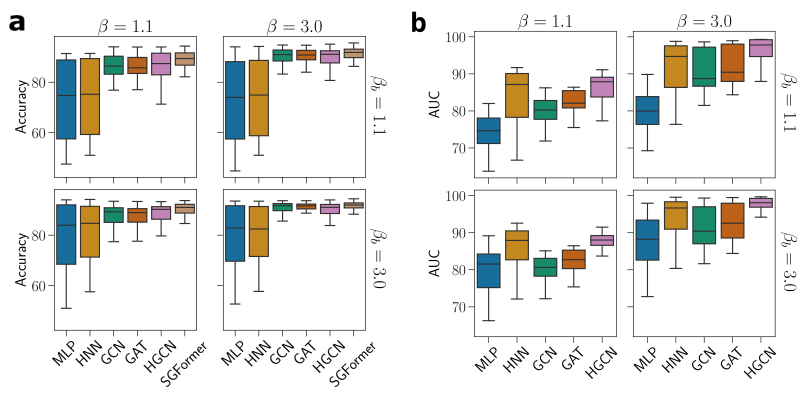

Figure 2: The impact of topology-feature correlation controlled by β and β_b on the performance of graph machine learning models in two tasks: (a) node classification and (b) link prediction. We set $\mathcal{N}_L = 6$ and $\alpha = 10$ for the NC task. The box ranges from the first quartile to the third quartile. A horizontal line goes through the box at the median. The whiskers go from each quartile to the minimum or maximum. The results are averaged over all other parameters.

5 Experiments

5.1 Parameter Space

We use the HypBench framework introduced in Section 4 and customize its parameters to generate a collection of datasets with $N_n = 5000$ nodes and $N_f = 2000$ features, displaying a wide spectrum of properties. The degree distributions for nodes in $\mathcal{G}_n$ and for nodes and features in $\mathcal{G}_{n,f}$ are tailored to exhibit either heterogeneity or homogeneity. This is achieved by setting $\gamma, \gamma_n, \gamma_f$ to the values $\{2.1, 3.5\}$. The average degrees vary between low and high values by adjusting $\langle k \rangle$, and $\langle k_n \rangle$ to $\{3, 30\}$. The clustering coefficient in both networks, along with the degree of correlation between them, is set to range between low and high values by fixing β and β_b to $\{1.1, 3\}$. For the task of NC, the number of labels are $\mathcal{N}_l = \{2, 3, 6, 10\}$. Homophily of node labels is varied by setting $\alpha = \{-1, 1, 5, 10\}$. For each unique combination of parameter values, we generate 10 network realizations with node features, resulting in a comprehensive benchmark for model evaluation. Consequently, for the LP task, machine learning models are tested on a benchmark comprising $2^7 \times 10 = 1280$ instances. In the case of NC, where we introduce two additional parameters, α and $\mathcal{N}_l$, the size of the benchmark is $2^7 \times 4^2 \times 10 = 20480$.

5.2 Machine learning models

In this work, we focus on two primary methodologies: feature-based methods, which entail node embedding based on their features: MLP, HNN [17], and GNNs, which integrate both features and network topology: GCN [14], GAT [15], HGCN [16] and a graph transformer called SGFormer [49].

Implementation: In the LP task, the links are randomly partitioned into training (85%), validation (5%), and test (10%) sets. For the NC task, the random distribution of nodes across the training, validation and test is set to 70%, 15% and 15%, respectively [16]. The LP and NC tasks are executed following the methodology outlined in [16]. We average results over five test-train splits. We trained the models on NVidia GeForce RTX 3080 GPU with Python 3.9, Cuda 11.7 and PyTorch 1.13. The original implementation of SGFormer focuses solely on the node classification (NC) task, so we evaluate this model exclusively on that task.

6 Results

We begin by analyzing how topology-feature correlation affects the performance of machine learning models by varying parameters β and β_b, which control the coupling between $\mathcal{G}_n$, $\mathcal{G}_{n,f}$, and the shared metric similarity space. For both NC and LP tasks, Fig. 2 shows that higher topology-feature correlation ($\beta = \beta_b = 3$) improves performance for both feature-based and GNN models compared

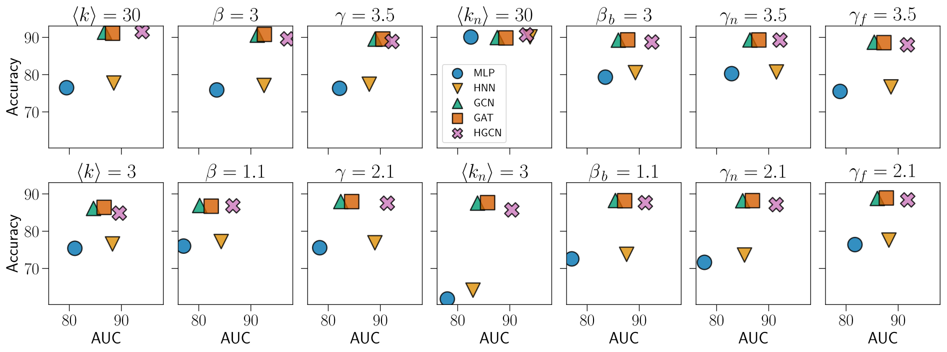

Figure 3: Impact of each individual parameter on the performance of NC and LP. The parameters are varied individually between high and low values, with performance averaged over all other parameters. In the case of NC, we set $\mathcal{N}_L = 6$ and $\alpha = 10$.

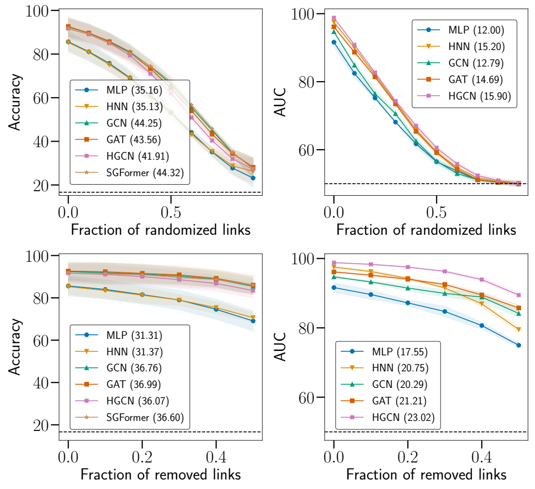

Figure 4: Robustness analysis of graph machine learning models under structural perturbations: link randomization (top row) and link removal (bottom row), evaluated for NC (left column) and LP (right column). For the unipartite network $\mathcal{G}_n$, we set the parameters to $\gamma = 3.5$, $\beta = 3$, and $\langle k \rangle = 3$. For the bipartite network $\mathcal{G}_{n,f}$, the parameters are $\gamma_n = 3.5$, $\gamma_f = 3.5$, $\beta_b = 3$, and $\langle k_n \rangle = 3$. For the NC task, we use $\alpha = 10$ and set the number of label classes to $\mathcal{N}_L = 6$. The results are averaged over 10 realizations. The black horizontal dashed line indicates the performance of a random baseline model. Numbers in parentheses in the legend represent the area between each model's robustness curve and the random baseline, serving as a quantitative measure of robustness.

to lower correlation ($\beta = \beta_b = 1.1$). In the NC task, feature-based models do not respond to changes in β and perform better with higher β_b. Conversely, GNN models, utilizing information from both $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$, benefit from high β and β_b values, not only in terms of average performance but also in terms of the spread around this average.

In Fig. 3, we present a summary of model performance on both downstream tasks, varying parameters individually between high and low values while averaging over others. Across most cases, HGCN outperforms or matches others in the LP task, and achieves competitive accuracy in the NC task, akin to other graph-based methods. In addition, for the NC task, the feature-based methods (MLP and HNN) are barely sensible to $\mathbb{S}^1$ model parameters, i.e., $\langle k \rangle$, β and γ. However, for the LP task, the topology-features correlation strongly impacts their AUC values, leading to MLP and HNN being sensitive to the parameters of the bipartite network, including β_b, γ_n, and γ_f, with a specific emphasis on $\langle k_n \rangle$. In contrast, GNNs are particularly responsive to variations in $\langle k_n \rangle$. It is worth mentioning that these observations highlight the sensitivity of graph machine learning methods to individual parameters.

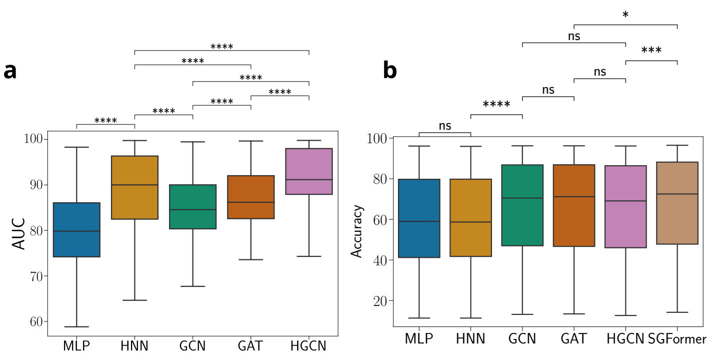

Figure 5: Aggregated results for (a) link prediction and (b) node classification tasks. The annotations denote the statistical significance levels derived from the Mann-Whitney U test. The label *ns* signifies p-value > 0.05, **** corresponds to p-value $\leq 10^{-4}$, and * to $0.01 \leq$ p-value ≤ 0.05.

In Fig. 4, we analyze the robustness of GNN models to structural perturbations. These include i) link randomization by swapping pairs of links chosen at random and ii) removal of links also chosen at random. These perturbations are applied to both the unipartite network of nodes $\mathcal{G}_n$ and the bipartite network between nodes and features $\mathcal{G}_{n,f}$ over an equal fraction of links in both networks, ensuring that no duplicate links are introduced. The evaluation is conducted for a selected set of parameters and for both NC and LP tasks. The top row presents the performance of graph machine learning models as a function of the fraction of randomized links. The bottom row shows models' performance as links are progressively removed, again applying the same removal fraction to both networks. To quantify their robustness, we measure the area between the performance curves of the models and the baseline representing random model performance. This figure illustrates that for the NC task, SGFormer, GCN, and GAT demonstrate similar and the highest robustness to link randomization and removal. In contrast, for the LP task, while HGCN, HNN, and GAT show similar and the highest performance under link randomization, HGCN stands out as the most robust model for link removal.

The global average picture of the performance of the analyzed models is shown in Fig. 5, where we averaged results over all the parameters in the HypBench framework. This analysis sheds light on model selection when no structural information is available about the data. We observe the superiority of the hyperbolic-based models in capturing essential network properties and connectivities for the LP task (Fig. 5(a)). Yet, HGCN outperforms HNN with statistical significance. As for the NC task, results are more ambiguous, with GNNs outperforming feature-based models overall. However, the distinctions in performance within the GNNs are not substantial. Hence, when dealing with unseen data, the simplicity of the GCN model makes it a more efficient choice.

7 Conclusion

In this study, we introduced HypBench framework for graph machine learning, and in particular for graph neural networks. This framework, leveraging the $\mathbb{S}^1/\mathbb{H}^2$ and bipartite-$\mathbb{S}^1/\mathbb{H}^2$ models, enables the generation of synthetic networks with controllable properties such as degree distributions, average degrees, clustering coefficients, and homophily levels. Our findings underscore the significance of topology-feature correlations in influencing GNN performance. Specifically, models embedded in the hyperbolic space, with its intrinsic hierarchical structure, outperformed the rest of the models in the LP task. This research contributes to the broader understanding of graph machine learning, providing insights into the suitability of various models under different network conditions. Furthermore, our benchmarking framework serves as a valuable tool for the community, enabling fair and standardized comparisons of new GNN architectures against established models. A promising direction for future work is to extend the bipartite-$\mathbb{S}^1/\mathbb{H}^2$ model to incorporate weighted features. This will enable HypBench to generate networks with non-binary features, allowing for more nuanced and flexible representations [52]. Additionally, the framework can be extended to accommodate community structures by using a non-homogeneous distribution of nodes within the $\mathbb{S}^1/\mathbb{H}^2$ similarity space, as done in [53] and [54].

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[4] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[5] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[6] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[7] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[8] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Xiaojie Guo. Graph neural networks: Foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 4840–4841, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3542609. URL `https://doi.org/10.1145/3534678.3542609`.

[9] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386. URL `https://ieeexplore.ieee.org/abstract/document/9046288`.

[10] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.01.001. URL `https://www.sciencedirect.com/science/article/pii/S2666651021000012`.

[11] J. Palowitch, A. Tsitsulin, B. Mayer, and B. Perozzi. Graphworld: Fake graphs bring real insights for gnns. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pages 3691–3701, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539203. URL `https://doi.org/10.1145/3534678.3539203`.

[12] Roya Aliakbarisani, M. Ángeles Serrano, and Marián Boguñá. Feature-enriched hyperbolic network geometry, 2023.

[13] M. Boguñá, I. Bonamassa, M. De Domenico, S. Havlin, D. Krioukov, and M. Á. Serrano. Network geometry. *Nat. Rev. Phys.*, 3:114–135, 2021. doi: 10.1038/s42254-020-00264-4.

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[16] I. Chami, Z. Ying, Ch. Ré, and J. Leskovec. Hyperbolic graph convolutional neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/0415740eaa4d9decbc8da001d3fd805f-Paper.pdf`.

[17] O. Ganea, G. Bécigneul, and T. Hofmann. Hyperbolic neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[18] Francisco Melo. *Area under the ROC Curve*, pages 38–39. Springer, New York, NY, 2013. ISBN 978-1-4419-9863-7. doi: 10.1007/978-1-4419-9863-7_209. URL `https://doi.org/10.1007/978-1-4419-9863-7_209`.

[19] M. Yasir, J. Palowitch, A. Tsitsulin, L. Tran-Thanh, and B. Perozzi. Examining the effects of degree distribution and homophily in graph learning models, 2023.

[20] Seiji Maekawa, Yuya Sasaki, George Fletcher, and Makoto Onizuka. Gencat: Generating attributed graphs with controlled relationships between classes, attributes, and topology. *Information Systems*, 115:102195, 2023. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2023.102195. URL `https://www.sciencedirect.com/science/article/pii/S0306437923000315`.

[21] Seiji Maekawa, Koki Noda, Yuya Sasaki, et al. Beyond real-world benchmark datasets: An empirical study of node classification with gnns. *Advances in Neural Information Processing Systems*, 35:5562–5574, 2022.

[22] Chaokun Wang, Binbin Wang, Bingyang Huang, Shaoxu Song, and Zai Li. Fastsgg: Efficient social graph generation using a degree distribution generation model. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 564–575, 2021. doi: 10.1109/ICDE51399.2021.00055.

[23] Sajad Darabi, Piotr Bigaj, Dawid Majchrowski, Artur Kasymov, Pawel Morkisz, and Alex Fit-Florea. A framework for large-scale synthetic graph dataset generation. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[24] Axel Wassington, Raúl Higueras, and Sergi Abadal. Skymap: a generative graph model for gnn benchmarking. *Frontiers in Artificial Intelligence*, 7:1427534, 2024.

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

[26] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78:046110, Oct 2008. doi: 10.1103/PhysRevE.78.046110. URL `https://link.aps.org/doi/10.1103/PhysRevE.78.046110`.

[27] Neil Shah. Scale-free, attributed and class-assortative graph generation to facilitate introspection of graph neural networks. In *KDD Mining and Learning with Graphs*, 2020.

[28] Robert Jankowski, Pegah Hozhabrierdi, Marián Boguñá, and M. Á Serrano. Feature-aware ultra-low dimensional reduction of real networks. *arXiv preprint arXiv:2401.09368*, 2024.

[29] M. Á. Serrano, D. Krioukov, and M. Boguñá. Self-Similarity of Complex Networks and Hidden Metric Spaces. *Phys. Rev. Lett.*, 100(7):078701, 2008. doi: 10.1103/PhysRevLett.100.078701.

[30] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82(3):036106, 2010. doi: 10.1103/PhysRevE.82.036106.

[31] M. Á. Serrano and M. Boguñá. *The Shortest Path to Network Geometry: A Practical Guide to Basic Models and Applications*. Elements in Structure and Dynamics of Complex Networks. Cambridge University Press, 2022. doi: 10.1017/9781108865791.

[32] L. Gugelmann, K. Panagiotou, and U. Peter. Random Hyperbolic Graphs: Degree Sequence and Clustering. In *Autom Lang Program (ICALP 2012, Part II), LNCS 7392*, 2012. doi: 10.1007/978-3-642-31585-5_51.

[33] M. A. Abdullah, N. Fountoulakis, and M. Bode. Typical distances in a geometric model for complex networks. *Internet Math.*, 1, 2017. doi: 10.24166/im.13.2017.

[34] T. Friedrich and A. Krohmer. On the Diameter of Hyperbolic Random Graphs. *SIAM J. Discrete Math.*, 32(2):1314–1334, 2018. doi: 10.1137/17M1123961.

[35] T. Müller and M. Staps. The diameter of KPKVB random graphs. *Adv. Appl. Probab.*, 51(2): 358–377, 2019. doi: 10.1017/apr.2019.23.

[36] M. Boguñá, D. Krioukov, P. Almagro, and M. Á. Serrano. Small worlds and clustering in spatial networks. *Phys. Rev. Res.*, 2:023040, Apr 2020. doi: 10.1103/PhysRevResearch.2.023040.

[37] E. Candellero and N. Fountoulakis. Clustering and the Hyperbolic Geometry of Complex Networks. *Internet Math.*, 12(1-2):2–53, 2016. doi: 10.1080/15427951.2015.1067848.

[38] N. Fountoulakis, P. van der Hoorn, T. Müller, and M. Schepers. Clustering in a hyperbolic model of complex networks. *Electron. J. Probab.*, 26:1 – 132, 2021. doi: 10.1214/21-EJP583.

[39] Robert Jankowski, Antoine Allard, Marián Boguñá, and M Ángeles Serrano. The d-mercator method for the multidimensional hyperbolic embedding of real networks. *Nature Communications*, 14(1):7585, 2023.

[40] Sarwan Ali, Muhammad Haroon Shakeel, Imdadullah Khan, Safiullah Faizullah, and Muhammad Asad Khan. Predicting attributes of nodes using network structure. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(2):1–23, 2021.

[41] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.

[42] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[43] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. New benchmarks for learning on non-homophilous graphs. *arXiv preprint arXiv:2104.01404*, 2021.

[44] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

[45] Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. Understanding heterophily for graph neural networks. *arXiv preprint arXiv:2401.09125*, 2024.

[46] Christopher Bishop. *Pattern recognition and machine learning*. Springer, New York, NY, 2006.

[47] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.04.005. URL `https://www.sciencedirect.com/science/article/pii/S1063520310000552`.

[48] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf`.

[49] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 36:64753–64773, 2023.

[50] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems*, 2021.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Çaglar Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[52] Antoine Allard, M. Ángeles Serrano, Guillermo García-Pérez, and Marián Boguñá. The geometric nature of weights in real complex networks. *Nature Communications*, 8(1):14103, 2017. ISSN 2041-1723. doi: 10.1038/ncomms14103. URL `https://doi.org/10.1038/ncomms14103`.

[53] Konstantin Zuev, Marián Boguñá, Ginestra Bianconi, and Dmitri Krioukov. Emergence of soft communities from geometric preferential attachment. *Scientific Reports*, 5(1):9421, 2015. ISSN 2045-2322. doi: 10.1038/srep09421. URL `https://doi.org/10.1038/srep09421`.

[54] Guillermo García-Pérez, M. Ángeles Serrano, and Marián Boguñá. Soft communities in similarity space. *Journal of Statistical Physics*, 173(3):775–782, 2018. ISSN 1572-9613. doi: 10.1007/s10955-018-2084-z. URL `https://doi.org/10.1007/s10955-018-2084-z`.

[55] Yuling Wang, Xiao Wang, Xiangzhou Huang, Yanhua Yu, Haoyang Li, Mengdi Zhang, Zirui Guo, and Wei Wu. Intent-aware recommendation via disentangled graph contrastive learning. *arXiv preprint arXiv:2403.03714*, 2024.

[56] Yangqin Jiang, Chao Huang, and Lianghao Huang. Adaptive graph contrastive learning for recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pages 4252–4261, 2023.

A Homophily in the synthetic networks

Fig. A.1(a) depicts the relation between $\mathcal{H}$ and the parameter α, where a monotonic dependence on $|\alpha|$ can be observed. However, this relation is not symmetric for $\mathcal{N}_l > 2$. The lower $\mathcal{H}$ for negative α stems from the angular organization of the labeled nodes. Indeed, Fig. A.1(b) shows that for the high positive α, the distance between two maximally distant nodes is lower compared to the same negative α value. Notice that $\alpha = 0$ corresponds to a maximally random assignment of labels, which results in the expected homophily parameter taking the value $\langle \mathcal{H} \rangle = 1/\mathcal{N}_l$.

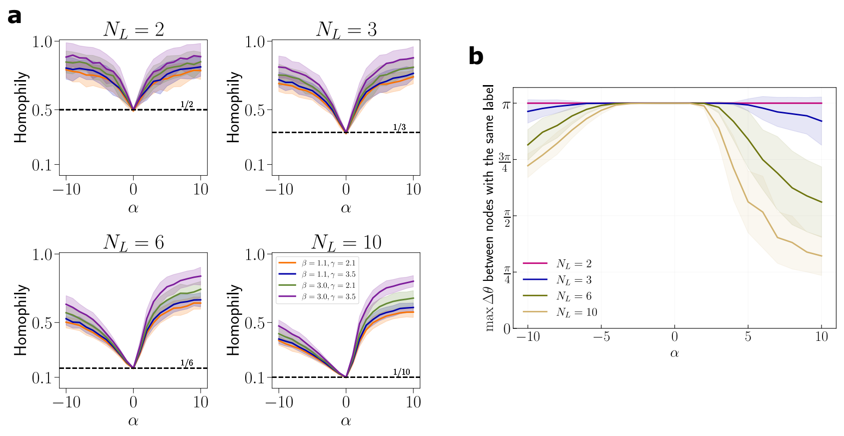

Figure A.1: (a) Value of homophily in the synthetic networks. Parameters of the networks: $N = 1000$, $\langle k \rangle = 30$. The black horizontal line indicates the random case, i.e., for $\alpha = 0$ which based on Eq. 8 gives us $\mathcal{H} = 1/N_L$ where N_L is the number of labels. The results are averaged over 100 realizations. (b) Maximum angular distance between nodes in the same community in function of parameter α. The results are averaged over 100 realizations.

B A comparative analysis of the topological properties of a real-world network and its synthetic counterpart produced using $\mathbb{S}^1/\mathbb{H}^2$ and bipartite-$\mathbb{S}^1/\mathbb{H}^2$ models

To assess how well the geometric models replicate the structural properties of the original network, we compare key topological properties of the real Cora dataset with those of its synthetic counterparts. Specifically, we analyze the degree distribution, clustering spectra (defined as the average clustering coefficient per degree class), and average nearest neighbors' degree for both the unipartite network $\mathcal{G}_n$ and the node-feature bipartite network $\mathcal{G}_{n,f}$ in the real and synthetic networks. The synthetic networks are generated using the $\mathbb{S}^1/\mathbb{H}^2$ and bipartite-$\mathbb{S}^1/\mathbb{H}^2$ models. The clustering coefficient, $\bar{c}$, for each node in $\mathcal{G}_n$ of degree k is calculated as the fraction of connected pairs among its neighboring nodes relative to the total possible pairs of neighbors $k(k-1)/2$. To compute the bipartite clustering, $\bar{c}_b$, for a given node in $\mathcal{G}_{n,f}$ of degree k_n, each pair of its neighboring features is considered connected if they share at least one common node, apart from the node in question. Then, bipartite clustering is computed similarly to the clustering coefficient in a unipartite network as the fraction between the number of connected pairs of features and $k_n(k_n-1)/2$. This definition also applies to the bipartite clustering coefficient of features. Finally, $\bar{k}_{nn}$ in both $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$ quantifies the average degree of the neighbors for a given node (of degrees k and k_n) or feature (of degree k_f). Fig. A.2 presents these comparisons, along with statistical variability across 100 realizations of the synthetic models. The results demonstrate that the model effectively reproduces the topological characteristics observed in the real network.

Fig. A.3 compares the AUC of the LP task in the Cora dataset with that in synthetic networks exhibiting similar structural properties. The results show that GNN models achieve comparable AUC scores on both the real and synthetic networks, indicating that the synthetic benchmarks effectively capture relevant network characteristics.

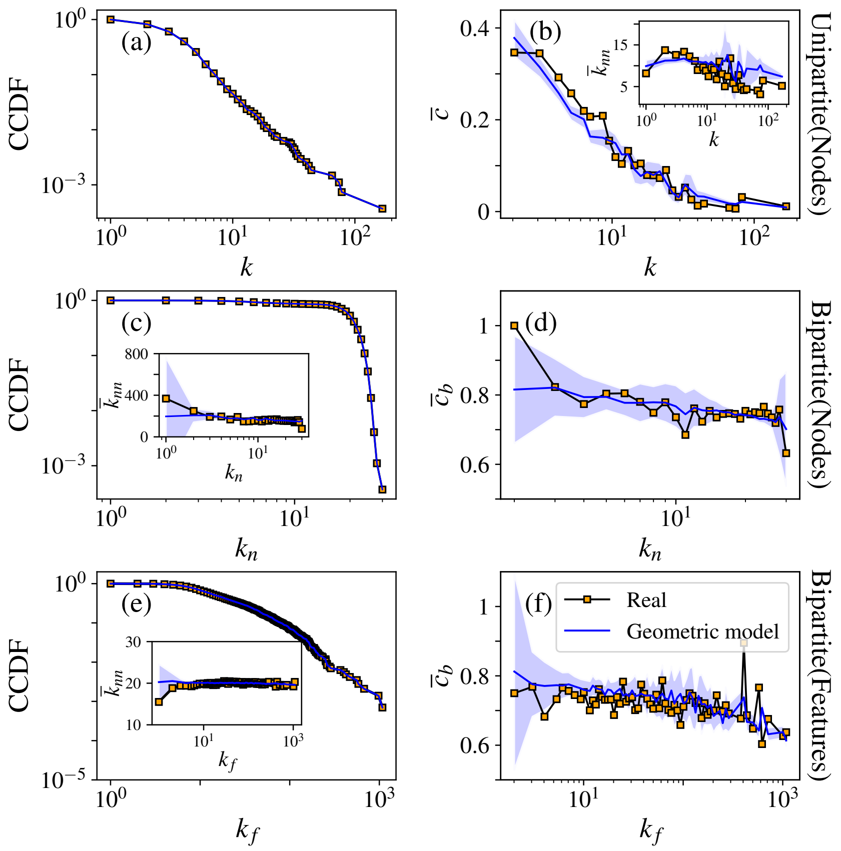

Figure A.2: Topological properties of Cora dataset and its synthetic counterpart generated by the geometric network models including $\mathbb{S}^1/\mathbb{H}^2$ and bipartite-$\mathbb{S}^1/\mathbb{H}^2$ models. (a),(c),(e) Complementary cumulative degree distribution of nodes in $\mathcal{G}_n$ and those of nodes and features in $\mathcal{G}_{n,f}$. (b),(d),(f) clustering spectrum of nodes as a function of node degrees in $\mathcal{G}_n$ and bipartite clustering sepctrum of nodes and features as a function of nodes and features degrees in $\mathcal{G}_{n,f}$. Insets are average nearest neighbors degree function in $\mathcal{G}_n$ and $\mathcal{G}_{n,f}$. Exponential binning is applied in the computation of $\bar{c}$ and $\bar{k}_{nn}$ for $\mathcal{G}_n$, as well as for $\bar{c}_b$ and $\bar{k}_{nn}$ of the features in $\mathcal{G}_{n,f}$. The blue shaded region indicates the two-σ intervals around the mean, obtained from 100 realizations of the geometric network model.

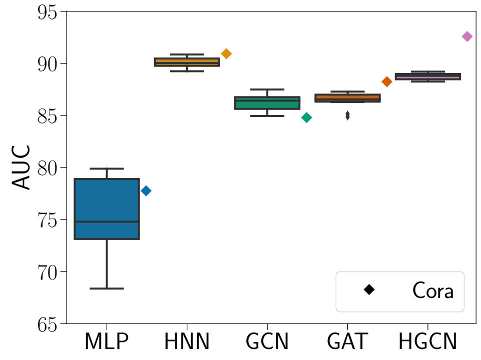

Figure A.3: AUC scores for the LP task using various GNN models on the real Cora network and on synthetic networks generated with parameters approximating Cora's structural characteristics. The synthetic network parameters are: $\beta = 1.1$, $\langle k \rangle = 3$, $\gamma = 3.5$; and $\beta_b = 1.1$, $\langle k_n \rangle = 30$, $\gamma_n = 3.5$, $\gamma_f = 3.5$. Box plots show the distribution of AUC scores across 10 realizations of the synthetic networks, while the diamond markers indicate the performance on the real Cora network.

C Algorithms to generate synthetic networks with features and node labels using HypBench framework

Algorithm 1 Algorithm to generate synthetic networks using HypBench

```
1: Generate the unipartite network of nodes $\mathcal{G}_n$
    1.1 Fix the number of nodes $N_n$, the level of clustering through parameter $\beta > 1$, the exponent of power-law degree distribution $\gamma$ and the target average degree $\langle k \rangle$
    1.2 Set $\mu = \frac{\beta}{2\pi\langle k \rangle} \sin \frac{\pi}{\beta}$
    1.2 Set similarity space radius $R = \frac{N_n}{2\pi}$
    1.3 Randomly assign each node an angular position $\theta \in [0, 2\pi]$ on the 1D sphere
    1.4 Assign every node a hidden degree $\kappa$ sampled from $\rho(\kappa) = (\gamma - 1)\kappa_0^{\gamma-1}\kappa^{-\gamma}$ with $\kappa_0 = \frac{\gamma-2}{\gamma-1}\langle k \rangle$. Alternatively, assign hidden degrees from any desired distribution $\rho(\kappa)$ with average degree $\langle k \rangle$
    1.5 Connect node pairs with probability
        $$p(\kappa, \kappa', \Delta\theta) = \frac{1}{1 + \chi^{\beta}} \text{ with } \chi = \frac{R\Delta\theta}{\mu\kappa\kappa'}$$
    and $\Delta\theta = \pi - |\pi - |\theta - \theta'||$
2: Generate the bipartite network of nodes and features $\mathcal{G}_{n,f}$
    2.1 Fix the number of features $N_f$, bipartite clustering level through parameter $\beta_b > 1$, average degree of nodes $\langle k_n \rangle$ and features $\langle k_f \rangle$, and the power-law exponents $\gamma_n$ (for nodes) and $\gamma_f$ (for features). The number of nodes $N_n$ is automatically accounted for from the identity $N_n\langle k_n \rangle = N_f\langle k_f \rangle$
    2.2 Set $\mu_b = \frac{\beta_b}{2\pi\langle k_n \rangle} \sin \frac{\pi}{\beta_b}$
    2.3 Uniformly distribute features along the same sphere as nodes in $\mathcal{G}_n$ with each feature having $\theta_f \in [0, 2\pi]$
    2.4 Place nodes at the same angular positions as in $\mathcal{G}_n$, i.e., $\theta_n = \theta$
    2.5 Assign every node and feature a hidden degree, $\kappa_n$ and $\kappa_f$, sampled from power-law distribution with exponents $\gamma_n$ and $\gamma_f$ or, alternatively, from any given distributions
    2.6 Connect each pair of node and feature with probability
        $$p_b(\kappa_n, \kappa_f, \Delta\theta) = \frac{1}{1 + \chi^{\beta_b}} \text{ with } \chi \equiv \frac{R\Delta\theta}{\mu_b\kappa_n\kappa_f}$$
3: Assigning labels to nodes
    3.1 Fix the number of clusters, each associated with a label $\mathcal{N}_L$ and the homophily level $\alpha$
    3.2 Place the "center" of each cluster to a random position in the circle
    3.3 Assign node $i$ to cluster $X$ with probability $P_X(i) = \frac{\Delta\theta_{iX}^{-\alpha}}{\sum_{Y \in \mathcal{N}_L} \Delta\theta_{iY}^{-\alpha}}$ where $\Delta\theta_{iX}$ is the angular distance between node $i$ and the center of cluster $X$
```