Flexible Diffusion Scopes with Parameterized Laplacian for Heterophilic Graph Learning

Qincheng Lu[1], **Jiaqi Zhu**[1], **Sitao Luan**[1,2,*], **Xiao-Wen Chang**[1,*]
{qincheng.lu@mail, sitao.luan@mail, jiaqi.zhu@mail, chang@cs}.mcgill.ca
[1]McGill University; [2] Mila - Quebec Artificial Intelligence Institute; *Corresponding Author

Abstract

The ability of Graph Neural Networks (GNNs) to capture long-range and global topology information is limited by the scope of conventional graph Laplacian, leading to unsatisfactory performance on some datasets, particularly on heterophilic graphs. To address this limitation, we propose a new class of parameterized Laplacian matrices, which provably offers more flexibility in controlling the diffusion distance between nodes than the conventional graph Laplacian, allowing long-range information to be adaptively captured through diffusion on graph. Specifically, we first prove that the diffusion distance and spectral distance on graph have an order-preserving relationship. With this result, we demonstrate that the parameterized Laplacian can accelerate the diffusion of long-range information, and the parameters in the Laplacian enable flexibility of the diffusion scopes. Based on the theoretical results, we propose topology-guided rewiring mechanism to capture helpful long-range neighborhood information for heterophilic graphs. With this mechanism and the new Laplacian, we propose two GNNs with flexible diffusion scopes: namely the Parameterized Diffusion based Graph Convolutional Networks (PD-GCN) and Graph Attention Networks (PD-GAT). Synthetic experiments reveal the high correlations between the parameters of the new Laplacian and the performance of parameterized GNNs under various graph homophily levels, which verifies that our new proposed GNNs indeed have the ability to adjust the parameters to adaptively capture the global information for different levels of heterophilic graphs. They also outperform the state-of-the-art (SOTA) models on 6 out of 7 real-world benchmark datasets, which further confirms their superiority.

1 Introduction

Combining graph signal processing and Convolutional Neural Networks (CNNs) [1], Graph Neural Networks (GNNs) has achieved remarkable success on machine learning tasks with non-Euclidean data [2–11]. In GNNs, high-order neighbors are recursively incorporated through diffusion across multiple stacked layers. In each layer, unlike CNNs where neighbors are weighted differently, the convolutional kernel in the vanilla GNNs [3] and many other popular variants [4, 6] assign the same weight in a neighbourhood, or use weights determined by commonly used normalization of adjacency or Laplacian matrices.

Recent studies show that the optimal choice of a normalized Laplacian is data-dependent [12, 13]. Aggregation with learnable weights has been proposed to enrich GNNs expressiveness [14]. However, limited knowledge about long-range neighbors and the global graph structure prevents conventional local aggregation from achieving optimal performance [15]. For example, while GAT [5] allows for learnable weights, it generates node representations solely based on the representations of its direct neighbours. This works well on the homophilic graphs [16, 17], where a node and its neighbours are likely to have the same label. However, as the limited and fixed diffusion scope, GAT suffers from

Q. Lu et al., Flexible Diffusion Scopes with Parameterized Laplacian for Heterophilic Graph Learning. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

significant performance loss in node classification tasks involving heterophilic graphs [18, 19], where nodes from different classes tend to be connected.

Non-local neighborhood information is found to be helpful to deal with heterophily problem for GNNs [18, 20–25], but it has not been fully explored considering the design of graph Laplacian and its diffusion scope. We will address this issue in this paper. The main contributions of our work are as follows: **(1)** We propose a new class of parameterized normalized graph Laplacian matrices, which offer better control over the diffusion process, and include several widely used normalized Laplacians as special cases. Through the parameters in new Laplacian, we can adjust the diffusion and spectral distances between nodes by altering spectral properties of the graph, and thus enable flexible message passing to capture local and global information adaptively. **(2)** We establish a theorem to prove that spectral distance can be used as a surrogate function of diffusion distance, which significantly reduces the computational cost for comparing relative distance between nodes. The theoretical results substantially extends the scope of [8] and overcomes some of its shortcomings. **(3)** Based on the new graph Laplacian and the theorem, we propose two architectures, the Parameterized Diffusion based Graph Convolutional Networks (PD-GCN) and Graph Attention Networks (PD-GAT), which enable flexible diffusion [1]. The empirical results demonstrate the effectiveness and superiority of the proposed models compared with SOTA GNNs for node classification tasks on graph across various homophily levels, especially on heterophilic graphs. The proposed strategy is characterized by its flexibility and seamless integration with other types of graph aggregation.

This paper is organized as follows: notation and background knowledge are introduced in Section 2; in Section 3, we propose the new class of Laplacian, show its properties, prove the theoretical results on spectral and diffusion distances and present the proposed architectures; in Section 4, we show the experimental results and comparisons on synthetic and real-world graphs, and conduct ablation study.

2 Preliminaries

In this section, we introduce notation and background knowledge. We use **bold** font for vectors and matrices. For a matrix $\mathbf{B} = (b_{ij})$, $|\mathbf{B}| = (|b_{ij}|)$ and its i^{th} row is denoted as $\mathbf{B}_{i,:}$ or $\mathbf{b}_i^\mathsf{T}$. We use $\mathbf{B}||\mathbf{C}$ and $[\mathbf{B}, \mathbf{C}]$ to denote column and row concatenation of matrices $\mathbf{B}$ and $\mathbf{C}$, respectively. We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote an undirected connected graph with the vertex set $\mathcal{V}$ (with $|\mathcal{V}| = N$) and the edge set $\mathcal{E}$. We have a node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ whose i^{th} row is the transpose of the feature vector $\mathbf{x}_i \in \mathbb{R}^{d_0}$ of node v_i. The learned node representation matrix at the l^{th} layer of GNNs is denoted by $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d_l}$. For a node $v_i \in \mathcal{V}$, $\mathcal{N}(v_i) \subseteq \mathcal{V}$ denotes the set of neighbouring nodes of v_i. The adjacency matrix of $\mathcal{G}$ is denoted by $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{N \times N}$ with $a_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. The degree matrix of $\mathcal{G}$ is $\mathbf{D} = \mathrm{diag}(d_{ii}) \in \mathbb{R}^{N \times N}$ with $d_{ii} = \sum_j a_{ij}$. Three Laplacian matrices, namely the combinatorial Laplacian, random-walk normalized Laplacian and symmetric normalized Laplacian are respectively defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \quad \mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L}, \quad \mathbf{L}_{\mathrm{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}. \tag{1}$$

We use $\boldsymbol{\phi}^{(k)}$ to denote the k^{th} eigenvector of a Laplacian corresponding to the k^{th} smallest positive eigenvalue $\lambda^{(k)}$.

2.1 Local Graph Aggregation

Most modern GNNs adopt the message-passing framework [26], in which the representation $\mathbf{h}_u$ of node u is generated by iterative local aggregation of its neighbors and its own representation from the previous layer [26]. The local graph aggregation at the l^{th} layer is expressed as:

$$\mathbf{H}^{(l)} = \sigma(\mathbf{S}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{2}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is a learnable parameter and $\sigma(\cdot)$ is a non-linear activation function. Here $\mathbf{S}$ is the aggregation operator that provides weights for neighboring messages. Common choices for a fixed $\mathbf{S}$ include adjacency matrices corresponding to $\mathbf{L}$, $\mathbf{L}_{\mathrm{rw}}$, and $\mathbf{L}_{\mathrm{sym}}$. Alternatively, learnable weights can be assigned. One approach to assigning learnable weights is graph attention [27], where, $r_{ij}^{(l)}$, the unnormalized attention score that node v_i gives to node v_j at the l-th layer is defined as

$$r_{ij}^{(l)} = \mathrm{LeakyReLU}\big((\mathbf{a}^{(l)})^\mathsf{T}[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)}||\mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}]\big), \tag{3}$$

where $\mathbf{a}^{(l)} \in \mathbb{R}^{2d_l}$ is trainable. The aggregation at the l-th layer is weighted by $\mathbf{S}_{ij}^{(l)} = \mathrm{softmax}_j(r_{ij}^{(l)})$.

[1] Code available at https://github.com/wzzlcss/Flexible_Diffusion_for_Graph

2.2 Spectral-based Nodes Relative Distances

The relative position of nodes on the graph could be told by eigenvectors of Laplacian matrices [28]. As the graph data is non-Euclidean, various measurements are proposed to describe the distance between nodes [29–31]. Among them, the diffusion distance is often used to model how node v_i influence node v_j by considering random walks along edges [8]. At the initial step, the random walk starts at a node, and moves to one of its neighbours at the next step. The diffusion distance between v_i and v_j is proportional to the probability that the random walk starting at node v_i meets the random walk starting at node v_j at step t. Based on $\mathbf{L}_{\text{rw}}$, the **diffusion distance** is calculated as (see [32]):

$$d_t(v_i, v_j; \mathbf{L}_{\text{rw}}) = \Big(\sum_{k=1}^{N-1} e^{-2t\lambda^{(k)}} (\phi_i^{(k)} - \phi_j^{(k)})^2 \Big)^{\frac{1}{2}}. \tag{4}$$

Another measure of the distance between v_i and v_j is the **spectral distance** defined as (see [31]):

$$d_s(v_i, v_j; \mathbf{L}_{\text{rw}}) = |\phi_i^{(1)} - \phi_j^{(1)}|. \tag{5}$$

Note that $\boldsymbol{\phi}^{(1)}$ gives positional information of nodes [28, 33, 34]. A more general definition of d_s involves the k eigenvectors corresponding to the k smallest eigenvalues, but here we take $k = 1$ for simplicity and informativeness.

2.3 Related Works

Design of Graph Laplacian and Adjacency Matrices. Efforts have been made to overcome the drawbacks of message passing with the conventional graph Laplacian or adjacency matrices. This includes works that incorporate novel attention mechanisms into GNNs [35–39], as well as transformer-based GNNs that consider fully connected graphs [28, 34]. The parameterized graph shift operator (PGSO) [12] generalizes conventional adjacency matrices, while ωGNN [14] offers trainable weighting factors for aggregation. The Directional Graph Network (DGN) [8] proposes aggregation matrices, namely the directional average matrix $\mathbf{B}_{\text{av}}$ and the directional derivative matrix $\mathbf{B}_{\text{dx}}$, which contain weights that represent the topological importance based on eigenvectors of the conventional Laplacian. Based on a given eigenvector $\boldsymbol{\phi}$, they are defined as:

$$\mathbf{B}_{\text{av}}(\boldsymbol{\phi}) = |\nabla\boldsymbol{\phi}|,\ \mathbf{B}_{\text{dx}}(\boldsymbol{\phi}) = \nabla\boldsymbol{\phi} - \text{diag}(\nabla\boldsymbol{\phi}\mathbf{1}),\ \nabla\phi_{ij} = \phi_j - \phi_i \text{ if } e_{ij} \in \mathcal{E} \text{ else } 0. \tag{6}$$

The proposed flexible diffusion with the new class of parameterized Laplacian matrices in our work differs from these works as it offers a theoretical way to control the spectral properties of graphs and allows adaptation to different homophily levels. Our proposed method enhances conventional message passing, and is independent of the specific local aggregation framework or attention mechanism it is combined with, offering the potential to improve various GNNs, including graph transformers.

Heterophily and Long-range Topology Information. On heterophily graphs, nodes tend to connect others that have different labels. Regarding to heterophilic graph learning, recent studies have found that the global receptive field is more desired and necessary than local information [23, 40–42], as the information from the same class are more likely to come from distant nodes. However, GNNs will suffer from information loss with an increased number of diffusion steps, a phenomenon known as over-squashing [43]. Thus, the difficulty in obtaining long-range neighborhood information without compromising its quality explains the unsatisfactory performance of GNNs on heterophilic graphs. Meanwhile, transformer-based GNNs still fall short on existing heterophilic benchmarks [34], suggesting that naively creating edges between distant nodes cannot handle this issue. Various rewiring methods also have been proposed [20, 43–46]. Our work addresses the learning challenge on heterophilic graphs by accelerating diffusion from long-range neighbors, and is characterized by both the flexibility of the parameterized diffusion and the topology-guided rewiring strategy.

3 Parameterized Normalized Laplacian and Flexible Diffusion

We first propose a new parameterized normalized Laplacian matrix, which defines a new class of graph Laplacians with its directional information that adapts to the graph's homophily levels, as described in Section 3.1. Then we discuss how this new Laplacian enables local aggregation to become aware of the global graph structure. Specifically, we propose two variants that incorporate the new parameterized normalized Laplacian into GNNs in Section 3.2: the Parameterized Diffusion

based Graph Convolutional Networks (PD-GCN) and Graph Attention Networks (PD-GAT). Based on the new Laplacian, we propose the topology-guided graph rewiring strategy, and its ability to capture long-range neighborhood information in heterophily graphs is theoretically justified in Section 3.3.

3.1 Parameterized Normalized Laplacian

In order to gain a more refined control over the diffusion on a graph, we define a new class of Laplacian matrices, denoted by $\mathbf{L}^{(\alpha,\gamma)}$, where the two parameters γ and α determine its spectral properties. Then, we propose an efficient method to compare the relative diffusion distances.

Definition 3.1. *A parameterized normalized Laplacian matrix is defined as*

$$\mathbf{L}^{(\alpha,\gamma)} = \gamma[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-\alpha}\mathbf{L}[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\alpha-1} \tag{7}$$

and the corresponding parameterized normalized adjacent matrix is defined as

$$\mathbf{P}^{(\alpha,\gamma)} = \mathbf{I} - \mathbf{L}^{(\alpha,\gamma)}, \tag{8}$$

where the parameters $\gamma \in (0, 1]$ and $\alpha \in [0, 1]$.

The normalized Laplacian matrices defined in Eq. (1) are special cases of this new class of Laplacian matrices: when $\alpha = 1$ and $\gamma = 1$, $\mathbf{L}^{(\alpha,\gamma)} = \mathbf{L}_{\text{rw}}$; when $\alpha = \frac{1}{2}$ and $\gamma = 1$, $\mathbf{L}^{(\alpha,\gamma)} = \mathbf{L}_{\text{sym}}$. Although we cannot choose α and γ such that $\mathbf{L}^{(\alpha,\gamma)}$ becomes $\mathbf{L}$, we have the following result:

$$\lim_{\gamma\to 0}\frac{1}{\gamma}\mathbf{L}^{(\alpha,\gamma)} = \lim_{\gamma\to 0}(\gamma\mathbf{D} + (1-\gamma)\mathbf{I})^{-\alpha}\mathbf{L} \times \lim_{\gamma\to 0}(\gamma\mathbf{D} + (1-\gamma)\mathbf{I})^{\alpha-1} = \mathbf{L}, \tag{9}$$

which implies that when γ is small enough, an eigenvector of $\mathbf{L}^{(\alpha,\gamma)}$ is a good approximation to an eigenvector of $\mathbf{L}$. The following theorem justifies the definition of $\mathbf{P}^{(1,\gamma)}$ as a random walk matrix, which generalizes the classic random walk matrix $\mathbf{D}^{-1}\mathbf{A}$.

Theorem 3.1. *The $\mathbf{P}^{(\alpha,\gamma)}$ defined in (8) is non-negative (i.e., all of its elements are non-negative), and when $\alpha = 1$, $\mathbf{P}^{(\alpha,\gamma)}\mathbf{1} = \mathbf{1}$. See the proof in Appendix B.1.*

The properties of eigenvalues of $\mathbf{L}^{(\alpha,\gamma)}$ are given in the following Theorem.

Theorem 3.2. *Suppose the graph $\mathcal{G}$ is connected. Then the symmetric $\mathbf{L}^{(1/2,\gamma)} \in \mathbb{R}^{n\times n}$ has the eigendecomposition:*

$$\mathbf{L}^{(1/2,\gamma)} = \mathbf{U}\mathbf{\Lambda}^{(\gamma)}\mathbf{U}^{\mathsf{T}}, \tag{10}$$

where $\mathbf{U} \in \mathbb{R}^{N\times N}$ is orthogonal and $\mathbf{\Lambda}^{(\gamma)} = \text{diag}(\lambda^{(i)}(\gamma))$,

$$0 = \lambda^{(0)}(\gamma) < \lambda^{(1)}(\gamma) \le \cdots \le \lambda^{(N-1)}(\gamma) \le 2. \tag{11}$$

Each $\lambda^{(i)}(\gamma)$ is strictly increasing with respect to γ for $i = 1 : N-1$. Furthermore, $\mathbf{L}^{(\alpha,\gamma)}$ has the eigendecomposition

$$\mathbf{L}^{(\alpha,\gamma)} = \left([\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\frac{1}{2}-\alpha}\mathbf{U}\right)\mathbf{\Lambda}^{(\gamma)}\left([\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\frac{1}{2}-\alpha}\mathbf{U}\right)^{-1}, \tag{12}$$

i.e., $\mathbf{L}^{(\alpha,\gamma)}$ share the same eigenvalues as $\mathbf{L}^{(1/2,\gamma)}$ and the columns of $[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\frac{1}{2}-\alpha}\mathbf{U}$ are the corresponding eigenvectors. See proof in Appendix B.2.

The following theorem shows the monotonicity of the diffusion distance with respect to the spectral distance under certain conditions. Here both distances are defined in terms of eigenvectors of $\mathbf{L}^{(1,\gamma)}$, cf. (4) and (5), which involve the eigenvectors of $\mathbf{L}_{\text{rw}}$.

Theorem 3.3. *Let v_i, v_j and v_m be nodes of the graph $\mathcal{G}$ such that $d_s(v_m, v_j; \mathbf{L}^{(1,\gamma)}) < d_s(v_i, v_j; \mathbf{L}^{(1,\gamma)})$. Then there is a constant C such that for $t \ge C$,*

$$d_t(v_m, v_j; \mathbf{L}^{(1,\gamma)}) < d_t(v_i, v_j; \mathbf{L}^{(1,\gamma)}), \tag{13}$$

with the reduction in distance being proportional to to $e^{-\lambda^{(1)}(\gamma)}$.

Generally, for the diffusion distance and spectral distance defined by $\mathbf{L}^{(\alpha,\gamma)}$ [28, 47], the monotonicity demonstrated in Theorem 3.3[2] also holds. Therefore, the spectral distance can be used as a good

[2]Theorem 3.3 substantially extends the scope of [8], which deals with the diffusion distance defined by $\mathbf{L}_{\text{rw}}$. Furthermore, Theorem 3.3 overcomes some shortcomings of [8], see details in Appendix B.3.

indicator of the diffusion distance, *i.e.,* if node i has a larger spectral distance to node j than node m, then node i would also have a larger diffusion distance to node j after enough time steps. According to (4), we needs to find all eigenvalues and eigenvectors of the Laplacian to calculate the diffusion distance, which is computationally expensive. But as Theorem 3.3 shows, we can easily compute the spectral distance and use it as a surrogate function of diffusion distance, which is much more efficient. Since a smaller diffusion distance implies greater influence between two nodes in GNNs, DGN [8] defines the directional aggregation operator that embeds the diffusion distance. However, DGN [8] requires k eigenvectors of $\mathbf{L}_{\text{rw}}$, where a larger k is preferred for better capturing of the actual diffusion process. We can encode such global structural information directly into the local aggregation by utilizing the spectral distance, which only involves the first non-trivial eigenvectors.

Furthermore, $\mathbf{L}^{(\alpha,\gamma)}$ allows for nuanced control over message passing. For graphs with different homophily levels, the node would prefer neighborhood information from different diffusion distances. Specifically, nodes with small diffusion distance would be favored on homophilic graphs and nodes with large diffusion distance would be helpful on heterophilic graphs to reduce the effect of noises [48]. Adjusting α and γ can change the eigenvalues and eigenvectors of $\mathbf{L}^{(\alpha,\gamma)}$, thereby altering diffusion distance. For instance, we can employ a smaller γ on heterophilic graphs to help the propagation of long-range neighborhood information and employ a larger γ on homophilic graphs to let GNNs focus on useful local information. Therefore, α and γ can be tuned as hyperparameters to enhance GNNs expressiveness according to the homophily levels.

3.2 Parameterized Diffusion Augmented GNNs

We present the proposed methods, which aim to adapt the diffusion process to the graph homophily levels. Two GNNs, the Parameterized Diffusion based GCN (PD-GCN) and GAT (PD-GAT), achieve such flexible diffusion through the new parameterized normalized Laplacian $\mathbf{L}^{(\alpha,\gamma)}$. These paradigms overcome the limitations of message passing on the graph structures encoded by the conventional Laplacian, and can accelerate the diffusion of long-range information in heterophilic graphs.

PD-GCN: The one-hop neighborhood aggregation in GCN [3] is a first-order approximation of the graph convolution using a polynomial filter on the graph Laplacian [49]. Accordingly, we propose Parameterized Diffusion based Graph Convolutional Networks (PD-GCN), where the aggregation utilizes the parameterized normalized adjacent matrix $\mathbf{P}^{(\alpha,\gamma)} = \mathbf{I} - \mathbf{L}^{(\alpha,\gamma)}$ instead. In PD-GCN, the l^{th} layer computes the updated representation as:

$$\mathbf{H}^{(l)} = \sigma(\mathbf{P}^{(\alpha,\gamma)}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{14}$$

here $\mathbf{P}^{(\alpha,\gamma)}$ serves as weights for aggregating neighboring nodes. While the vanilla GCN sets the weights solely based on paired nodes degrees, PD-GCN is able to encode global information into the local aggregation with its flexible diffusion scopes, surpassing the expressiveness of GCN for the following reasons: The first non-trivial eigenvector of the graph Laplacian tells the diffusion distance between nodes, as shown in Theorem 3.3, and thus indicates the dominant direction of diffusion. However, whether a graph is homophilic or heterophilic is independent of its spectral properties. As a result, the diffusion governed by the conventional Laplacian may not be optimal. In $\mathbf{L}^{(\alpha,\gamma)}$, with suitable values of α and γ that account for the graph's homophily levels, PD-GCN can adjust its diffusion scopes accordingly to perform more effective message propagation.

PD-GAT: The learnable aggregation weights in GAT are merely based on local node features, and does not adapt to the specific characteristics of the global topology. PD-GAT addresses this shortcoming by equipping the attention mechanism with parameterized edge features. Denote the first non-trivial eigenvector of $\mathbf{L}^{(\alpha,\gamma)}$ as $\boldsymbol{\phi}^{(1)}(\alpha,\gamma)$, the feature of $e_{ij} \in \mathcal{E}$ is defined as follows:

$$\mathbf{f}^{(i,j)}(\alpha,\gamma) = \left[\mathbf{B}_{\text{av}}(\boldsymbol{\phi}^{(1)}(\alpha,\gamma))_{ij}, \mathbf{B}_{\text{dx}}(\boldsymbol{\phi}^{(1)}(\alpha,\gamma))_{ij}\right]^{\mathsf{T}}, \tag{15}$$

where $\mathbf{B}_{\text{av}}(\boldsymbol{\phi}^{(1)}(\alpha,\gamma))$ and $\mathbf{B}_{\text{dx}}(\boldsymbol{\phi}^{(1)}(\alpha,\gamma))$ are matrices corresponding to the aggregation and diversification operations utilizing the vector field $\nabla\boldsymbol{\phi}^{(1)}(\alpha,\gamma)$ (see (6)). Here $\mathbf{B}_{\{\text{av,dx}\}}(\boldsymbol{\phi}^{(1)}(\alpha,\gamma))_{ij}$ contains information of the diffusion distance between v_i and v_j, which describes the relative position of v_i and v_j on the graph. Instead of regarding them as edge weights in DGN [8], we use this node relative distance information to assist the learning of attention between nodes. We propose to define the attention score at the l^{th} layer between v_i and v_j before normalization as:

$$r_{ij}^{(l)}(\alpha,\gamma) = \text{LeakyReLU}((\mathbf{a}^{(l)})^{\mathsf{T}}[\mathbf{W}_n^{(l)}\mathbf{h}_i^{(l-1)} || \mathbf{W}_n^{(l)}\mathbf{h}_j^{(l-1)} || \mathbf{W}_e^{(l)}\mathbf{f}^{(i,j)}(\alpha,\gamma)]), \tag{16}$$

where $\mathbf{W}_n^{(l)}$ and $\mathbf{W}_e^{(l)}$ are learnable weights for node representation and edge features respectively. We use $\alpha_{ij}^{(l)}(\alpha,\gamma)$ to denote the attention score after softmax normalization. The PD-GAT with multi-heads attention compute the new representation as:

$$\mathbf{h}_i^{(l)} = \left(\Big\Vert_{m=1}^{M} \sigma\Big(\sum_{j\in\{\mathcal{N}(i),i\}} \alpha_{ij}^{(l,m)}(\alpha,\gamma)\mathbf{W}^{(l,m)}\mathbf{h}_j^{(l-1)} \Big) \right) \mathbf{W}^{(l)} \tag{17}$$

where M is the number of heads. Compared with GAT, the additional number of parameters in PD-GAT introduced by (16) is negligible, since $\mathbf{f}^{(i,j)}(\alpha,\gamma) \in \mathbb{R}^2$.

3.3 Topology-Guided Rewiring Mechanism for Long-range Diffusion

This subsection introduces a graph rewiring technique to further enhance the proposed parameterized diffusion on heterophilic graphs. Since diffusion distance is a criterion for the efficiency of message passing, we regard locally disconnected nodes with large diffusion distance as long-range neighbors. According to Theorem 3.3, large spectral distances also identify long-range neighbors. Therefore, the first non-trivial eigenvector $\boldsymbol{\phi}^{(1)}(\alpha,\gamma)$ of $\mathbf{L}^{(\alpha,\gamma)}$ can be regarded as a one-dimensional embedding of nodes, and we refer to it as the (parameterized) spectral embedding. We refer to the node $v_{(\alpha,\gamma)}$ as the gradient node, whose corresponding element in $\boldsymbol{\phi}^{(1)}(\alpha,\gamma)$ is the maximum, *i.e.,* $v_{(\alpha,\gamma)}$ is located at the rightmost position in the 1D spectral embedding. In heterophilic graphs, our proposed rewiring strategy connects other nodes in the graph with the gradient node under a certain configuration of $\mathbf{L}^{(\alpha,\gamma)}$. The reasons for its effectiveness on heterophilic graph learning are provided below.

On heterophilic graphs, our proposed rewiring strategy aims to accelerate the diffusion process starting from a disconnected long-range neighbors v_j to target node v_i. To find a proper candidate v_j efficiently, we assume that: **(1)** v_j is to the right of v_i in the spectral embedding without loss of generality; **(2)** v_i and the gradient node $v_{(\alpha,\gamma)}$, are locally disconnected in the original graph; **(3)** $\boldsymbol{\phi}_j^{(1)}(\alpha,\gamma) - \boldsymbol{\phi}_i^{(1)}(\alpha,\gamma) \geq \frac{1}{2}(\max_m \boldsymbol{\phi}_m^{(1)}(\alpha,\gamma) - \min_n \boldsymbol{\phi}_n^{(1)}(\alpha,\gamma))$ to reduce the number of long-range candidate neighbors of v_i. Based on Theorem 3.3, we know that the newly created connection by the rewiring strategy between v_i and $v_{(\alpha,\gamma)}$ reduces the number of diffusion steps required for a message sent by v_j to reach v_i. In other words, the diffusion from v_j to $v_{(\alpha,\gamma)}$, and then from $v_{(\alpha,\gamma)}$ to v_i, is faster than the diffusion from v_j to v_i when v_i and $v_{(\alpha,\gamma)}$ are disconnected. Therefore, the proposed rewiring benefit the long-range diffusion thus enhance the learning on heterophilic graphs.

4 Experiments

In this section, we evaluate the effectiveness of our proposed models on synthetic and real-world benchmark datasets. In Section 4.1, we conduct ablation study to validate the effectiveness of each proposed component. In Section 4.2, we generate synthetic graphs to study how the performance of the proposed parameterized diffusion varies with the graph homophily levels. In Section 4.3, we compare the proposed architectures with baselines and SOTA models on 7 real-world datasets and the results show that parameterized diffusion augmented GNNs outperform the SOTA models on 6 out of 7 node classification tasks. These results highlight the explainability of our parameterized diffusion with $\mathbf{L}^{(\alpha,\gamma)}$, and verify that the proposed GNNs indeed can adaptively capture global information by adjusting parameters in $\mathbf{L}^{(\alpha,\gamma)}$ for graphs with different levels of homophily.

4.1 Ablation Study

In this subsection, we conduct ablation study to investigate the effectiveness of (1) parameterized normalized Laplacian $\mathbf{L}^{(\alpha,\gamma)}$ and its corresponding adjacency matrix $\mathbf{P}^{(\alpha,\gamma)}$, (2) the topology-guided graph rewiring strategy and (3) the parameterized diffusion based GCN and GAT, referred to as PD-GCN and PD-GAT. The ablation results on 7 heterophily datasets are summarized in Table 1. Note that both the parameterized diffusion and the rewiring step are influenced by α and γ. The optimal α and γ used in the ablation studies are provided in Appendix E. Appendix F provides a comparison with message passing over graphs with virtual nodes [51].

From Table 1, we make the following observations: (1) The rewiring strategy improves both baseline models and the proposed models; (2) Comparing PD-GAT and GAT, even without leveraging $\mathbf{L}^{(\alpha,\gamma)}$, *i.e.,* using only the default $\mathbf{L}_{\text{rw}}$ with $\alpha = 1$, $\gamma = 1$, both the rewiring strategy and the parameterized

Method	$\mathbf{L}^{(\alpha,\gamma)}$	$\mathcal{PD}$	$\mathcal{R}(\mathcal{G})$	roman-empire	amazon-ratings	minesweeper	tolokers	questions	squirrel filtered	chameleon filtered
GAT				80.87 ± 0.30	49.09 ± 0.63	92.01 ± 0.68	83.70 ± 0.47	77.43 ± 1.20	35.62 ± 2.06	39.21 ± 3.08
			✓	81.21 ± 0.71	47.27 ± 0.52	92.86 ± 0.53	84.28 ± 0.68	78.79 ± 1.01	40.45 ± 1.36	43.51 ± 5.06
	✓		✓	82.27 ± 0.60	47.69 ± 0.64	92.94 ± 0.55	84.59 ± 0.53	78.97 ± 1.04	41.49 ± 2.72	43.89 ± 4.67
PD-GAT		✓		82.50 ± 0.60	49.29 ± 0.57	92.07 ± 0.69	84.11 ± 0.41	77.82 ± 0.78	36.50 ± 1.37	41.84 ± 2.76
		✓	✓	86.82 ± 0.60	47.81 ± 0.54	93.15 ± 0.80	84.39 ± 0.49	79.40 ± 0.79	41.48 ± 2.34	43.09 ± 4.13
	✓	✓		83.46 ± 0.39	**49.69 ± 0.51**	92.15 ± 0.71	84.11 ± 0.41	78.66 ± 0.97	37.83 ± 1.54	43.37 ± 3.01
	✓	✓	✓	**87.27 ± 0.64**	48.03 ± 0.58	**93.27 ± 0.56**	**84.74 ± 0.59**	**79.55 ± 0.81**	**42.09 ± 2.65**	**44.16 ± 4.20**
GCN				73.69 ± 0.74	48.70 ± 0.63	89.75 ± 0.52	83.64 ± 0.67	76.09 ± 1.27	39.47 ± 1.47	40.89 ± 4.12
			✓	74.50 ± 0.58	48.32 ± 0.54	89.93 ± 0.60	83.41 ± 0.93	77.24 ± 1.14	40.92 ± 1.49	43.60 ± 1.91
	✓		✓	74.60 ± 0.65	48.45 ± 0.48	89.94 ± 0.61	83.46 ± 0.92	77.27 ± 1.14	41.12 ± 1.29	45.34 ± 4.53
PD-GCN		✓		73.97 ± 0.46	49.38 ± 0.80	91.60 ± 0.62	81.35 ± 0.66	74.80 ± 0.76	36.19 ± 2.45	41.65 ± 2.81
	✓	✓		**78.05 ± 0.49**	**49.49 ± 0.67**	**91.60 ± 0.62**	**83.83 ± 0.86**	**78.04 ± 0.91**	**43.31 ± 1.92**	**46.67 ± 3.56**

Table 1: Ablation study on real-world heterophily datasets proposed by [50]. Here, GCN and GAT are baseline models. A checkmark on $\mathbf{L}^{(\alpha,\gamma)}$ indicates the use of the parameterized normalized Laplacian (or the corresponding parameterized normalized adjacency $\mathbf{P}^{(\alpha,\gamma)}$) with optimally tuned α and γ, while an unchecked $\mathbf{L}^{(\alpha,\gamma)}$ denotes the use of the default $\mathbf{L}^{(1,1)} = \mathbf{L}_{\text{rw}}$ and $\mathbf{P}^{(1,1)} = \mathbf{P}_{\text{rw}}$. The term $\mathcal{PD}$ stands for "parameterized diffusion", which refers to the weighted aggregation with $\mathbf{P}^{(\alpha,\gamma)}$ for PD-GCN and the incorporation of edge features defined by $\mathbf{L}^{(\alpha,\gamma)}$ in the graph attention for PD-GAT, whereas unchecking $\mathcal{PD}$ means that the vanilla GCN or GAT is used. Lastly, $\mathcal{R}(\mathcal{G})$ stands for the graph rewiring strategy. Results for baselines with $\mathcal{R}(\mathcal{G})$ are obtained by applying baselines to the rewired graph, while results for baselines with both $\mathbf{L}^{(\alpha,\gamma)}$ and $\mathcal{R}(\mathcal{G})$ are from applying the rewiring technique with the optimally α and γ. Note that the rewiring technique is not applicable to PD-GCN, as PD-GCN assigns parameterized edge weights based on the original graph structure. The best results are highlighted in **bold** format respectively for GCN and GAT-based architectures.

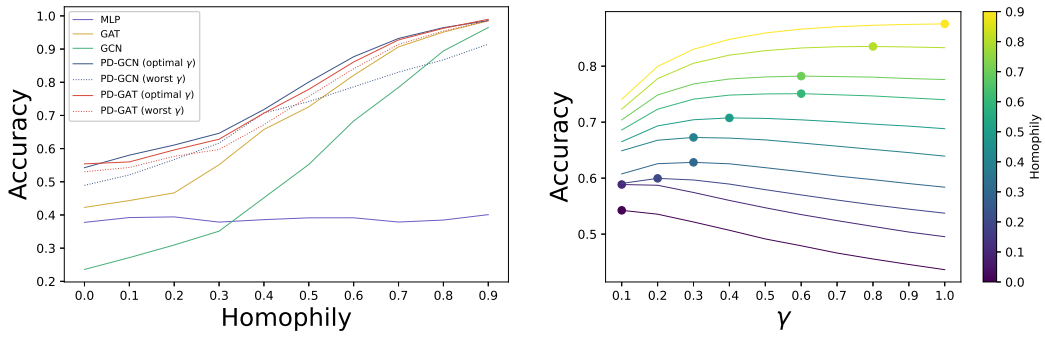
diffusion can individually improve model performance; (3) PD-GCN is more sensitive to the choice of Laplacian parameters. PD-GCN with the default parameters achieves performance gain over GCN, except on the *questions* and *squirrel-filtered*. However, with optimal parameters, PD-GCN even outperforms PD-GAT on these two datasets; (4) After incorporating $\mathbf{L}^{(\alpha,\gamma)}$, *i.e.,* perform a grid-search for the optimal α, γ for each graph to find the most suitable relative node positions, the effectiveness of the parameterized diffusion and the rewiring strategy is further enhanced; (5) Each component is indispensable for the success of PD-GCN and PD-GAT, except for the *amazon-ratings* dataset, where the rewiring does not provide a benefit[3].

4.2 Synthetic Experiments

In this subsection, we investigate the behavior of parameterized diffusion under different homophily levels, focusing on its performance and the optimal choice of γ. We follows [20, 52] to generate synthetic graphs characterized by a homophily coefficient $\mu \in \{0.0, 0.1, \ldots, 0.9\}$, representing the chance that a node forms a connection to another node with the same label. We say the graph is more heterophilic for a smaller μ. A detailed explanation about how the synthetic graph are generated is given in Appendix D.2. We generate 5 synthetic graphs under each homophily coefficient μ. Node features are sampled from overlapping multi-Gaussian distributions. And for each generated graph, nodes are randomly partitioned into train/validation/test sets with a ratio of 60%/20%/20%. Each model (MLP, GAT, GCN, PD-GCN, PD-GAT) is trained under the same hyperparameter setting with the learning rate 0.01, weight decay 0.001 and dropout rate 0.1. The number of layers is set to be 2 for each model. Each model use 64 hidden units and the attention-based model use 8 heads with 8 hidden states per head. For simplicity in comparison, the rewiring strategy is not applied to the proposed methods. For PD-GCN and PD-GAT, we fix $\alpha = 1.0$ and only vary $\gamma \in \{0.0, 0.1, \ldots, 0.9\}$.

As shown in Figure 1 (a), both PD-GCN and PD-GAT outperform MLP, GCN and GAT across all homophily levels, particularly in heterophilic cases. PD-GCN with the optimal γ achieves the best overall performance. However, the effectiveness of PD-GCN relies on a well-chosen γ, as its performance drops considerably with the worst γ, becoming worse than the baseline GNNs on more homophilic graphs. On the other hand, the performance of PD-GAT is more robust to γ. Even with the worst γ, PD-GAT still performs better results than the baselines. This difference arises because PD-GCN encodes the parameterized topology knowledge in the aggregation weights, whereas PD-

[3]It can be explained by the results in [50], which show that the GNNs yield negligible improvement over the graph-free models on *amazon-ratings*.



(a) **(b)**

Figure 1: Experiments on synthetic graphs with varying levels of homophily. The y-axis represents averaged test accuracy. (a) Comparison of PD-GCN and PD-GAT with baseline models. The x-axis denotes the graph homophily level, where a larger value indicates a more homophilic graph. The solid lines for the proposed models represent performance with the optimal γ, while the dashed lines show performance with the worst γ. (b) Each line corresponds to synthetic graphs with a specific homophily level, illustrating the relationship between the performance of a one-layered PD-GCN and γ. The line color indicates the homophily coefficient, with blue representing low homophily and yellow indicating high homophily. The dot markers denote the optimal γ for each homophily level.

GAT incorporates it into the edge features. As a result, the choice of parameter has a more direct impact on the performance of PD-GCN than PD-GAT.

Figure 1 (b) shows the performance of a one-layered PD-GCN to demonstrate the strong correlation between the parameter γ and the model performance across different graph homophily levels. We find that: **(1)** As the graph becomes more heterophilic, the optimal γ (the dot marker on each curve) for PD-GCN decreases, and vice versa. This suggests that the reduction of the diffusion distance between distant nodes, according to Theorem 3.3, is favored on heterophilic graphs. This also verifies that PD-GCN can indeed achieve flexible diffusion scopes by adjusting γ and γ is an interpretable parameter. **(2)** The performance changes smoothly with the value of γ. The curves corresponding to larger homophily levels are always higher than those with smaller homophily levels, which implies that, given a fixed γ, PD-GCN consistently performs better on more homophilic graphs. **(3)** Furthermore, the more heterophilic the graph is, the less satisfactory the performance with the default $\gamma = 1$. This phenomenon highlights that the limitations of the conventional Laplacian are significant in heterophilic cases, and the proposed new Laplacian effectively addresses this issue through parameterized diffusion.

4.3 Experiments on Real-world Datasets

In this subsection, we compare PD-GCN and PD-GAT with 6 baseline models[4]: GAT [5], GAT-sep [5], GCN [3], SAGE [4], Graph Transformer (GT) [53] and GT-sep, and 11 heterophily-specific SOTA models: H_2GCN [18], CPGNN [54], GPR-GNN [55], FSGNN [56], GloGNN [57], FAGCN [58], GBK-GNN [59], JacobiCov [60], BernNet [22], LINKX [61] and APPNP [62]. The comparison also contains GNNs with novel aggregation including DGN [8], ωGCN [14] and PGSO-GCN [12]. The experiments are conducted on 7 recently proposed heterophily benchmark datasets [50]: *roman-empire*, *amazon-ratings*, *minesweeper*, *tolokers*, *questions*, *Chameleon-filtered* and *Squirrel-filtered*[6].

Experimental Setups. For the 7 new heterophily datasets, we evaluate PD-GCN and PD-GAT and their variants under the same experiment settings for baseline models with fixed splits used in [50], with learning rate of $3 \cdot 10^{-5}$, weight decay of 0, dropout rate of 0.2, hidden dimension of 512 and

[4]Baseline models adopt residual connections at each layer, following the implementation provided by [50].

[5]Here "sep" means to concatenate the ego feature of node and the aggregated neighborhood information, which is the trick used in [50] and we follow this setting.

[6]The overall statistics of these real-word datasets are given in Appendix D.1. And additional results on homophilic datasets are provided in Appendix G

	roman-empire	amazon-ratings	minesweeper	tolokers	questions	squirrel-filtered	chameleon-filtered	Rank
GCN	73.69 ± 0.74	48.70 ± 0.63	89.75 ± 0.52	83.64 ± 0.67	76.09 ± 1.27	39.47 ± 1.47	40.89 ± 4.12	12.14
SAGE	85.74 ± 0.67	53.63 ± 0.39	93.51 ± 0.57	82.43 ± 0.44	76.44 ± 0.62	36.09 ± 1.99	37.77 ± 4.14	10.29
GAT	80.87 ± 0.30	49.09 ± 0.63	92.01 ± 0.68	83.70 ± 0.47	77.43 ± 1.20	35.62 ± 2.06	39.21 ± 3.08	11.57
GAT-sep	88.75 ± 0.41	52.70 ± 0.62	93.91 ± 0.35	83.78 ± 0.43	76.79 ± 0.71	35.46 ± 3.10	39.26 ± 2.50	8.71
GT	86.51 ± 0.73	51.17 ± 0.66	91.85 ± 0.76	83.23 ± 0.64	77.95 ± 0.68	36.30 ± 1.98	38.87 ± 3.66	10.43
GT-sep	87.32 ± 0.39	52.18 ± 0.80	92.29 ± 0.47	82.52 ± 0.92	78.05 ± 0.93	36.66 ± 1.63	40.31 ± 3.01	8.43
H_2GCN	60.11 ± 0.52	36.47 ± 0.23	89.71 ± 0.31	73.35 ± 1.01	63.59 ± 1.46	35.10 ± 1.15	26.75 ± 3.64	22.14
CPGNN	63.96 ± 0.62	39.79 ± 0.77	52.03 ± 5.46	73.36 ± 1.01	65.96 ± 1.95	30.04 ± 2.03	33.00 ± 3.15	22.29
GPR-GNN	64.85 ± 0.27	44.88 ± 0.34	86.24 ± 0.61	72.94 ± 0.97	55.48 ± 0.91	38.95 ± 1.99	39.93 ± 3.30	18.57
FSGNN	79.92 ± 0.56	52.74 ± 0.83	90.08 ± 0.70	82.76 ± 0.61	78.86 ± 0.92	35.92 ± 1.32	40.61 ± 2.97	9.86
GloGNN	59.63 ± 0.69	36.89 ± 0.14	51.08 ± 1.23	73.39 ± 1.17	65.74 ± 1.19	35.11 ± 1.24	25.90 ± 3.58	22.86
FAGCN	65.22 ± 0.56	44.12 ± 0.30	88.17 ± 0.73	77.75 ± 1.05	77.24 ± 1.26	41.08 ± 2.27	41.90 ± 2.72	13.71
GBK-GNN	74.57 ± 0.47	45.98 ± 0.71	90.85 ± 0.58	81.01 ± 0.67	74.47 ± 0.86	35.51 ± 1.65	39.61 ± 2.60	15.29
JacobiCov	71.14 ± 0.42	43.55 ± 0.48	89.66 ± 0.40	68.66 ± 0.65	73.88 ± 1.16	29.71 ± 1.66	39.00 ± 4.20	19.86
BernNet	65.56 ± 1.34	44.64 ± 0.56	77.99 ± 0.95	77.00 ± 0.65	70.43 ± 1.38	41.18 ± 1.77	40.90 ± 4.06	15.71
LINKX	56.15 ± 0.93	52.66 ± 0.64	56.78 ± 2.47	81.15 ± 1.23	71.96 ± 2.07	40.10 ± 2.21	42.34 ± 4.13	13.14
APPNP	65.87 ± 0.53	46.02 ± 0.73	69.62 ± 2.11	76.98 ± 1.03	64.77 ± 1.32	35.12 ± 1.12	37.50 ± 3.69	20.00
DGN	83.12 ± 0.47	47.65 ± 0.71	90.64 ± 0.61	OOM	OOM	38.56 ± 1.84	41.24 ± 3.62	11.20
ωGCN	74.79 ± 0.46	51.79 ± 0.74	90.71 ± 0.67	80.96 ± 1.04	71.10 ± 1.45	35.65 ± 2.06	40.93 ± 3.48	13.00
PGSO-GCN	74.72 ± 0.59	48.43 ± 0.41	87.84 ± 1.95	78.58 ± 1.73	OOM	41.06 ± 3.05	43.91 ± 3.04	11.83
PD-GCN	78.05 ± 0.49	49.49 ± 0.67	91.60 ± 0.62	83.83 ± 0.86	78.04 ± 0.91	43.31 ± 1.92	46.67 ± 3.56	6.57
PD-GAT	83.46 ± 0.39	49.69 ± 0.51	92.15 ± 0.71	84.11 ± 0.41	78.66 ± 0.97	37.83 ± 1.54	43.37 ± 3.01	6.71
PD-GAT ($\mathcal{R}(\mathcal{G})$)	87.27 ± 0.64	48.03 ± 0.58	93.27 ± 0.56	84.74 ± 0.59	79.55 ± 0.81	42.09 ± 2.65	44.16 ± 4.20	4.57
PD-GAT-sep	88.46 ± 0.58	52.57 ± 0.95	93.81 ± 0.42	84.04 ± 0.51	77.29 ± 0.73	37.01 ± 2.97	41.40 ± 4.74	6.14
PD-GAT-sep ($\mathcal{R}(\mathcal{G})$)	89.23 ± 0.56	50.96 ± 0.43	94.03 ± 0.45	84.83 ± 0.40	78.88 ± 0.94	39.69 ± 2.28	41.15 ± 4.66	4.29

Table 2: Experiment results on heterophily datasets proposed by [50]. Values stand for mean and standard deviation of evaluation metrics on the test datasets. Here *roman-empire*, *amazon-ratings*, *squirrel-filtered* and *chameleon-filtered* use accuracy for evaluation, while *minsweeper*, *tolokers* and *questions* use ROC AUC. The "sep" refers to the trick proposed in [18] which concatenates node's and the mean of neighbours' embedding in each aggregation step, instead of adding them together. For fair comparison, we use the same experimental setup in [50] for proposed models. Results for heterophily GNNs, except for BernNet, LINKX and APPNP, are reported by [50]. The top three results are highlighted in red, blue, and violet, respectively.

attention head of 8, and number of layers from 1 to 5. Following [50], the proposed models are also trained for 1000 steps with Adam optimizer and select the best step based on the performance on the validation set. For other GNNs, we perform a grid search for the learning rate, weight decay and dropout rate, and model-specific hyperparameters, with details provided in Appendix E.

The results are summarized in Table 2, where GNN baselines are organized in the top block, heterophily-specific GNNs are placed in the second block, models with novel aggregation operators are in the third block, and our proposed methods are listed in the bottom block.

Results and Comparisons. **(1)** In most cases, the proposed parameterized diffusion augmented GNNs obtain significant performance improvement against the baseline models, *i.e.,* PD-GCN, PD-GAT, PD-GAT-sep outperform GCN, GAT and GAT-sep, respectively. **(2)** The proposed models consistently surpass SOTA GNNs specifically designed to address heterophily, as well as GNNs utilizing various aggregation paradigms. The "sep" trick further enhances PD-GAT's performance on *roman-empire*, *minesweeper* and *tolokers*. The "sep" trick allows the aggregation step to assign negative weights to the propagated message, enabling node-wise diversification, which has been shown to be useful for heterophily data [19, 63]. **(3)** The superior performance of PD-GAT over GT-based methods indicates that finding the useful graph for message propagation with sparse attention is more effective than considering message passing between all pairs of nodes. **(4)** The topology-guided rewiring mechanism is effective for GAT and GAT-sep on most real-world benchmark datasets.

5 Conclusion

In this paper, we address the limitations of GNNs in capturing long-range and global topology information, particularly in heterophilic graphs, by proposing a novel class of parameterized normalized Laplacian matrices. The new Laplacian provides greater flexibility in controlling the diffusion distance between nodes, enabling adaptive diffusion scopes to accommodate varying levels of graph homophily. Then, we prove that the order-preserving relationship between the diffusion distance and spectral distance. With this result and the new Laplacian, we propose two models with flexible diffusion scopes, PD-GCN and PD-GAT, along with a topology-guided rewiring strategy that further enhances performance. The effectiveness of the proposed methods is justified both theoretically and empirically.

References

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[2] 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. 1

[3] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1, 5, 8

[4] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017. 1, 8

[5] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 1, 8

[6] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 1

[7] Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Break the ceiling: Stronger multi-scale deep graph convolutional networks. *Advances in neural information processing systems*, 32, 2019.

[8] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *Proceedings of the International Conference on Machine Learning*, pages 748–758. PMLR, 2021. 2, 3, 4, 5, 8, 15, 16, 18

[9] Mingde Zhao, Zhen Liu, Sitao Luan, Shuyuan Zhang, Doina Precup, and Yoshua Bengio. A consciousness-inspired planning agent for model-based reinforcement learning. *Advances in neural information processing systems*, 34:1569–1581, 2021.

[10] Chenqing Hua, Sitao Luan, Minkai Xu, Rex Ying, Jie Fu, Stefano Ermon, and Doina Precup. Mudiff: Unified diffusion for complete molecule generation. In *The Second Learning on Graphs Conference*, 2023.

[11] Qincheng Lu, Sitao Luan, and Xiao-Wen Chang. GCEPNet: Graph convolution-enhanced expectation propagation for massive mimo detection. *arXiv preprint arXiv:2404.14886*, 2024. 1

[12] George Dasoulas, Johannes F. Lutzeyer, and Michalis Vazirgiannis. Learning parametrised graph shift operators. In *International Conference on Learning Representations*, 2021. 1, 3, 8, 18

[13] Lorenzo Dall'Amico, Romain Couillet, and Nicolas Tremblay. Optimal laplacian regularization for sparse spectral community detection. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3237–3241. IEEE, 2020. 1

[14] Moshe Eliasof, Lars Ruthotto, and Eran Treister. Improving graph neural networks with learnable propagation operators. In *International Conference on Machine Learning*, pages 9224–9245. PMLR, 2023. 1, 3, 8

[15] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019. 1

[16] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, and Doina Precup. When do we need graph neural networks for node classification? *International Conference on Complex Networks and Their Applications*, 2023. 1

[17] Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[18] 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. 2, 8, 9

[19] Sitao Luan, Mingde Zhao, Chenqing Hua, Xiao-Wen Chang, and Doina Precup. Complete the missing half: Augmenting aggregation filtering with diversification for graph convolutional networks. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022. 2, 9

[20] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29. PMLR, 2019. 2, 3, 7, 18

[21] Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612*, 2020.

[22] Mingguo He, Zhewei Wei, Hongteng Xu, et al. BernNet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34, 2021. 8

[23] Sitao Luan, Chenqing Hua, Qincheng Lu, Liheng Ma, Lirong Wu, Xinyu Wang, Minkai Xu, Xiao-Wen Chang, Doina Precup, Rex Ying, et al. The heterophilic graph learning handbook: Benchmarks, models, theoretical analysis, applications and challenges. *arXiv preprint arXiv:2407.09618*, 2024. 3, 14

[24] Jianfei Li, Ruigang Zheng, Han Feng, Ming Li, and Xiaosheng Zhuang. Permutation equivariant graph framelets for heterophilous graph learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[25] Keke Huang, Yu Guang Wang, Ming Li, et al. How universal polynomial bases enhance spectral graph neural networks: Heterophily, over-smoothing, and over-squashing. *arXiv preprint arXiv:2405.12474*, 2024. 2

[26] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020. 2, 20

[27] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6):1–25, 2019. 2

[28] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021. 3, 4, 16

[29] Boaz Nadler, Stephane Lafon, Ioannis Kevrekidis, and Ronald Coifman. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Advances in neural information processing systems*, 18, 2005. 3

[30] Huaijun Qiu and Edwin R. Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007.

[31] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 3, 15

[32] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006. 3, 15

[33] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020. 3

[34] Luis Muller, Mikhail Galkin, Christopher Morris, and Ladislav Rampasek. Attending to graph transformers. *arXiv preprint arXiv:2302.04181*, 2023. 3

[35] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. 3

[36] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[37] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[38] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Direct multi-hop attention based graph neural network. *arXiv preprint arXiv:2009.14332*, 2020.

[39] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022. 3, 14

[40] Shuai Zheng, Zhenfeng Zhu, Zhizhe Liu, Youru Li, and Yao Zhao. Node-oriented spectral filtering for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 3

[41] Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. Less is more: on the over-globalizing problem in graph transformers. *arXiv preprint arXiv:2405.01102*, 2024.

[42] Yao Cheng, Caihua Shan, Yifei Shen, Xiang Li, Siqiang Luo, and Dongsheng Li. Resurrecting label propagation for graphs with heterophily and label noise. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 433–444, 2024. 3

[43] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020. 3

[44] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. *arXiv Preprint arXiv:1907.10903*, 2019.

[45] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. DropGNN: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021.

[46] Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. Diffwire: Inductive graph rewiring via the lovász bound. *arXiv preprint arXiv:2206.07369*, 2022. 3

[47] Yaron Lipman, Raif M Rustamov, and Thomas A Funkhouser. Biharmonic distance. *ACM Transactions on Graphics (TOG)*, 29(3):1–11, 2010. 4

[48] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021. 5

[49] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29, 2016. 5

[50] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023. 7, 8, 9, 14, 19

[51] Katsuhiko Ishiguro, Shin-ichi Maeda, and Masanori Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. *arXiv preprint arXiv:1902.01020*, 2019. 6, 19

[52] Fariba Karimi, Mathieu Génois, Claudia Wagner, Philipp Singer, and Markus Strohmaier. Homophily influences ranking of minorities in social networks. *Scientific Reports*, 8(1), jul 2018. doi: 10.1038/s41598-018-29405-7. 7

[53] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 8

[54] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11168–11176, 2021. 8

[55] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020. 8

[56] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science*, 62:101695, 2022. 8

[57] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, pages 13242–13256. PMLR, 2022. 8

[58] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3950–3957, 2021. 8

[59] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. GBK-GNN: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, pages 1550–1558, 2022. 8

[60] Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International Conference on Machine Learning*, pages 23341–23362. PMLR, 2022. 8

[61] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021. 8

[62] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018. 8

[63] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint arXiv:2109.05641*, 2021. 9

[64] Yilun Zheng, Sitao Luan, and Lihui Chen. What is missing in homophily? disentangling graph homophily for graph neural networks. *arXiv preprint arXiv:2406.18854*, 2024. 14

[65] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. New benchmarks for learning on non-homophilous graphs. *arXiv preprint arXiv:2104.01404*, 2021. 14

[66] 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. 19

[67] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020. 19

A Homophily Metrics

Here we review some commonly used metrics to measure homophily [23, 64]. We denote $\mathbf{z} \in \mathbb{R}^N$ as labels of nodes, and $\mathbf{Z} \in \mathbb{R}^{N \times C}$ as the one-hot encoding of labels, where C is the number of classes. Edge homophily H_{edge} and node homophily H_{node} are defined as follows:

$$H_{\text{edge}} = \frac{\{e_{ij} | e_{ij} \in \mathcal{E}, \mathbf{z}_i = \mathbf{z}_j\}}{|\mathcal{E}|}, \; H_{\text{node}} = \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \frac{|y_u = y_v : v \in \mathcal{N}(v)|}{|\mathcal{N}(u)|}. \tag{18}$$

Adjusted edge homophily H^*_{edge} considers classes imbalance and is defined as [50] :

$$H^*_{\text{edge}} = \frac{H_{\text{edge}} - \sum_c p^2(c)}{1 - \sum_c p^2(c)}. \tag{19}$$

Here $p(c) = \sum_{i:\mathbf{z}_i=c} \mathbf{D}_{ii}/(2|\mathcal{E}|)$, $c = 1 : C$, defines the degree-weighted distribution of class labels. The class homophily is also proposed to take class imbalance into account [65]:

$$H_{\text{class}} = \frac{1}{C-1} \sum_c \left[h_c - \frac{|\{v_i | \mathbf{z}_i = c\}|}{N} \right]_+ \tag{20}$$

$$h_c = \frac{\sum_{v_i:\mathbf{z}_i=c} |\{e_{ij} | e_{ij} \in \mathcal{E}, \mathbf{z}_i = \mathbf{z}_j\}|}{\sum_{v_i:\mathbf{z}_i=c} \mathbf{D}_{ii}}. \tag{21}$$

Label informativeness, which indicates the amount of information a neighbor's label provides about node's label, is defined as follows [50]:

$$LI = 2 - \frac{\sum_{c_1,c_2} p(c_1, c_2) \log p(c_1, c_2)}{\sum_c p(c) \log p(c)}, \tag{22}$$

where $p(c_1, c_2) = |\{e_{ij} | e_{ij} \in \mathcal{E}, \mathbf{z}_i = c_1, \mathbf{z}_j = c_2\}|/(2|\mathcal{E}|)$. The aggregation homophily $H^{\text{M}}_{\text{agg}}(\mathcal{G})$ measures the proportion of nodes that assign greater average weights to intra-class nodes than inter-class nodes. It is defined as follows [39]:

$$\begin{aligned} H^{\text{M}}_{\text{agg}}(\mathcal{G}) = \frac{1}{|\mathcal{V}|} \big| \{ v_i | \text{Mean}_j(\{S(\hat{\mathbf{A}}, \mathbf{Z})_{ij} | \mathbf{z}_i = \mathbf{z}_j\}) \\ \geq \text{Mean}_j(\{S(\hat{\mathbf{A}}, \mathbf{Z})_{ij} | \mathbf{z}_i \neq \mathbf{z}_j\}) \} \big|, \end{aligned} \tag{23}$$

where $S(\hat{\mathbf{A}}, \mathbf{Z}) = \hat{\mathbf{A}}\mathbf{Z}(\hat{\mathbf{A}}\mathbf{Z})^\intercal$ defines the post-aggregation node similarity, with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\text{Mean}_j(\{\cdot\})$ takes the average over node v_j of a given multiset of values. Under the homophily metrics mentioned above, a smaller value indicates a higher degree of heterophily. While H^*_{edge} can assume negative values, the other metrics fall within the range $[0, 1]$.

B Proof of Theorem

B.1 Proof of theorem 3.1

Proof. By Definition 3.1, we have

$$\begin{aligned} \mathbf{P}^{(\alpha,\gamma)} &= \mathbf{I} - \mathbf{L}^{(\alpha,\gamma)} = \mathbf{I} - \gamma[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-\alpha}\mathbf{L}[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\alpha-1} \\ &= [\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-\alpha}[\gamma\mathbf{D} + (1-\gamma)\mathbf{I} - \gamma\mathbf{L}][\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\alpha-1} \\ &= [\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-\alpha}[\gamma\mathbf{A} + (1-\gamma)\mathbf{I}][\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\alpha-1} \end{aligned}$$

It is easy to see that all elements in $\mathbf{P}^{(\alpha,\gamma)}$ are non-negative. Since $\mathbf{A}\mathbf{1} = \mathbf{D}\mathbf{1}$, we have

$$\mathbf{P}^{(1,\gamma)}\mathbf{1} = (\gamma\mathbf{D} + (1-\gamma)\mathbf{I})^{-1} (\gamma\mathbf{A} + (1-\gamma)\mathbf{I}) \mathbf{1} = \mathbf{1},$$

completing the proof. □

B.2 Proof of theorem 3.2

Proof. For any nonzero $\mathbf{x} \in \mathbb{R}^N$, write $\mathbf{y} := [\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-1/2}\mathbf{x}$. Then we have

$$\frac{\mathbf{x}^\mathsf{T}\mathbf{L}^{(1/2,\gamma)}\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{x}} = \frac{\mathbf{x}^\mathsf{T}\gamma[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-1/2}\mathbf{L}[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{-1/2}\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{x}}$$
$$= \frac{\gamma\mathbf{y}^\mathsf{T}\mathbf{L}\mathbf{y}}{\mathbf{y}^\mathsf{T}[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]\mathbf{y}} = \frac{\frac{\gamma}{2}\sum_{ij} a_{ij}(y_i - y_j)^2}{\gamma\sum_{ij} a_{ij}y_i^2 + (1-\gamma)\sum_i y_i^2}$$

By the Rayleigh quotient theorem,

$$\lambda^{(0)}(\gamma) = \min_{\mathbf{y}\neq\mathbf{0}} \frac{\frac{\gamma}{2}\sum_{ij} a_{ij}(y_i - y_j)^2}{\gamma\sum_{ij} a_{ij}y_i^2 + (1-\gamma)\sum_i y_i^2} = 0, \tag{24}$$

where the minimum is reached when $\mathbf{y}$ is a multiple of $\mathbf{1}$ and

$$\lambda^{(N-1)}(\gamma) = \max_{\mathbf{y}\neq\mathbf{0}} \frac{\frac{\gamma}{2}\sum_{ij} a_{ij}(y_i - y_j)^2}{\gamma\sum_{ij} a_{ij}y_i^2 + (1-\gamma)\sum_i y_i^2}$$
$$\leq \max_{\mathbf{y}\neq\mathbf{0}} \frac{\sum_{ij} a_{ij}(y_i^2 + y_j^2)}{\sum_{ij} a_{ij}y_i^2} \leq 2,$$

leading to Eq. (11). The proof of showing $\lambda^{(1)}(\gamma) \neq 0$ if and only if $\mathcal{G}$ is connected is similar to [31], thus we omit the details here.

By the Courant-Fischer min-max theorem, for $\gamma \neq 0$,

$$\lambda^{(i)}(\gamma) = \min_{\{S:\dim(S)=i+1\}} \max_{\{\mathbf{x}:\mathbf{0}\neq\mathbf{x}\in S\}} \frac{\mathbf{x}^\mathsf{T}\mathbf{L}^{(1/2,\gamma)}\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{x}}$$
$$= \min_{\{S:\dim(S)=i+1\}} \max_{\{\mathbf{y}:\mathbf{0}\neq\mathbf{y}\in S\}} \frac{\mathbf{y}^\mathsf{T}\mathbf{L}\mathbf{y}}{\mathbf{y}^\mathsf{T}[\mathbf{D} - \mathbf{I} + (1/\gamma)\mathbf{I}]\mathbf{y}}.$$

It is obvious that the Rayleigh quotient $\frac{\mathbf{y}^\mathsf{T}\mathbf{L}\mathbf{y}}{\mathbf{y}^\mathsf{T}[\mathbf{D}-\mathbf{I}+(1/\gamma)\mathbf{I}]\mathbf{y}}$ is strictly increasing with respect to $\gamma \in (0, 1]$ if $\mathbf{L}\mathbf{y} \neq 0$, *i.e.,* $\mathbf{y}$ not a multiple of $\mathbf{1}$. Note that $\lambda^{(0)}(\gamma) = 0$ and it is reached when $\mathbf{L}\mathbf{y} = \mathbf{0}$, or equivalently $\mathbf{y}$ is a multiple of $\mathbf{1}$. Thus, $\lambda^{(i)}(\gamma)$ is strictly increasing with respect to γ for $i = 1 : N-1$.

From the eigendecomposition of the symmetric $\mathbf{L}^{(1/2,\gamma)}$ in (10), we can find the eigendecomposition of $\mathbf{L}^{(\alpha,\gamma)}$ as follows:

$$\mathbf{L}^{(\alpha,\gamma)} = [\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{1/2-\alpha}\mathbf{L}^{(1/2,\gamma)}[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\alpha-1/2}$$
$$= [\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{1/2-\alpha}(\mathbf{U}\mathbf{\Lambda}^{(\gamma)}\mathbf{U}^\mathsf{T})[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{\alpha-1/2}$$
$$= \left([\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{1/2-\alpha}\mathbf{U}\right)\mathbf{\Lambda}^{(\gamma)}\left([\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{1/2-\alpha}\mathbf{U}\right)^{-1}$$

Thus, $\lambda^{(i)}(\gamma)$ is also an eigenvalue of $\mathbf{L}^{(\alpha,\gamma)}$ for $i = 0 : N-1$, and the i-th column of $[\gamma\mathbf{D} + (1-\gamma)\mathbf{I}]^{1/2-\alpha}\mathbf{U}$ is a corresponding eigenvector. □

B.3 Proof of Theorem 3.3

Proof. The proof is similar to the proof of [8]. By [32], the diffusion distance at time t between node v_i and v_j can be expressed as:

$$d_t(v_i, v_j) = \left(\sum_{k=1}^{n-1} e^{-2t\lambda^{(k)}(\gamma)}(\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right)^{\frac{1}{2}}, \tag{25}$$

where $\lambda^{(1)}(\gamma) \leq \lambda^{(2)}(\gamma) \leq \cdots \leq \lambda^{(n-1)}(\gamma)$ are eigenvalues of $\mathbf{L}^{(1,\gamma)}$, and $\{\boldsymbol{\phi}^{(1)}(\gamma), \boldsymbol{\phi}^{(2)}(\gamma), \ldots, \boldsymbol{\phi}^{(n-1)}(\gamma)\}$ are the corresponding eigenvectors. We omit the zero $\lambda^{(0)}(\gamma)$.

The inequality $d_t(v_m, v_j) < d_t(v_i, v_j)$ is then equivalent as

$$\begin{aligned}&\left(\sum_{k=1}^{n-1} e^{-2t\lambda^{(k)}(\gamma)}(\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right)^{\frac{1}{2}} \\ &< \left(\sum_{k=1}^{n-1} e^{-2t\lambda^{(k)}(\gamma)}(\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right)^{\frac{1}{2}}.\end{aligned} \tag{26}$$

We can take out $\lambda^{(1)}(\gamma)$ and $\boldsymbol{\phi}^{(1)}(\gamma)$ and rearrange the above inequality as:

$$\begin{aligned}&\sum_{k=2}^{n-1} e^{-2t\lambda^{(k)}(\gamma)}\left((\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2 - (\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right) \\ &< e^{-2t\lambda^{(1)}(\gamma)}\left((\phi_i^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2 - (\phi_m^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2\right).\end{aligned} \tag{27}$$

The left-hand side of Eq. (27) has an upper bound:

$$\begin{aligned}&\sum_{k=2}^{n-1} e^{-2t\lambda^{(k)}(\gamma)}\left|(\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2 - (\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right| \\ &\leq e^{-2t\lambda^{(2)}(\gamma)}\sum_{k=2}^{n-1}\left|(\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2 - (\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right|.\end{aligned} \tag{28}$$

Then Eq. (27) holds if:

$$\begin{aligned}&e^{-2t\lambda^{(2)}(\gamma)}\sum_{k=2}^{n-1}\left|(\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2 - (\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right| \\ &\leq e^{-2t\lambda^{(1)}(\gamma)}\left((\phi_i^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2 - (\phi_m^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2\right),\end{aligned} \tag{29}$$

which is equivalent to

$$\begin{aligned}&\log\left(\frac{(\phi_i^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2 - (\phi_m^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2}{\sum_{k=2}^{n-1}\left|(\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2 - (\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right|}\right) \\ &\times \frac{1}{2(\lambda^{(1)}(\gamma) - \lambda^{(2)}(\gamma))} < t.\end{aligned} \tag{30}$$

Let the constant C be the left-hand side of Eq. (30), then if we take $t \geq \lfloor C \rfloor + 1$, we have $d_t(v_m, v_j) < d_t(v_i, v_j)$. Note that C exits if

$$\frac{(\phi_i^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2 - (\phi_m^{(1)}(\gamma) - \phi_j^{(1)}(\gamma))^2}{\sum_{k=2}^{n-1}\left|(\phi_m^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2 - (\phi_i^{(k)}(\gamma) - \phi_j^{(k)}(\gamma))^2\right|} > 0, \tag{31}$$

which is satisfied since we assume $|\phi_i^{(1)}(\gamma) - \phi_j^{(1)}(\gamma)| > |\phi_m^{(1)}(\gamma) - \phi_j^{(1)}(\gamma)|$. The original theorem [8] is only based on $\boldsymbol{\phi}^{(1)}$ and does not assume that v_m must satisfy $|\phi_i^{(1)} - \phi_j^{(1)}| > |\phi_m^{(1)} - \phi_j^{(1)}|$, which is necessary for the existence of C. In addition, the original theorem [8] assumes that v_m is obtained by taking a gradient step from v_i, *i.e.,* $\phi_m - \phi_i = \max_{j:v_j \in \mathcal{N}(v_i)}(\phi_j - \phi_i)$, while this property is not needed for the proof. Therefore, Theorem 3.3 both extends and overcomes the shortcomings of [8]. □

C Computational Complexity Analysis

The complexity of obtaining the first non-trivial eigenvector of the Laplacian is $O(|E|)$, aligning with previous research [8, 28]. Here, we further explain the algorithm, where full eigendecomposition is not required since we only need the first non-trivial eigenvector.

Suppose the symmetrically normalized Laplacian $\mathbf{L}_s$ has eigen-pairs $\{(\mathbf{u}_i, \lambda_i)\}_{i=1:N}$ with $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N \leq 2$. The eigenvector corresponding to the smallest eigenvalue 0 is $\mathbf{u}_1 = \mathbf{D}^{\frac{1}{2}}\mathbf{e}$, where $\mathbf{D}$ is the degree matrix and $\mathbf{e}$ is a all-ones vector. To compute the first non-trivial eigenvector $\mathbf{u}_2$ of $\mathbf{L}_s$, we define $\bar{\mathbf{L}}_s = 2\mathbf{I} - \mathbf{L}_s$ and denote the eigenvalues of $\bar{\mathbf{L}}_s$ as $\bar{\lambda}_i = 2 - \lambda_i$. We know that $\bar{\mathbf{L}}_s$ and $\mathbf{L}_s$ shares eigenvectors, and $\bar{\lambda}_1 > \bar{\lambda}_2 \geq \cdots \geq \bar{\lambda}_N$. Thus, we want to get the eigenvector associated with $\bar{\lambda}_2$.

Next, we define the deflated matrix $\widetilde{\mathbf{L}}_s = \bar{\mathbf{L}}_s - 2\mathbf{u}_1\mathbf{u}_1^\mathsf{T}/\|\mathbf{u}_1\|_2^2$. The largest eigenvalue of $\widetilde{\mathbf{L}}_s$ is $\bar{\lambda}_2$, and the corresponding eigenvector is $\mathbf{u}_2$. This eigenvector can be obtained using power method on $\widetilde{\mathbf{L}}_s$. Note that during power iteration, we compute the multiplication of $\widetilde{\mathbf{L}}_s$ with a vector $\mathbf{b}$, where we do not need to form the $\widetilde{\mathbf{L}}_s$ explicitly and the complexity is $O(|E|)$ with a reasonable precision of stopping criterion: (1) $\bar{\mathbf{L}}_s\mathbf{b}$ would take $O(|E|)$, considering the sparsity of the graph; (2) The multiplication of $2\mathbf{u}_1\mathbf{u}_1^\mathsf{T}/\|\mathbf{u}_1\|_2^2$ with $\mathbf{b}$ takes $O(N)$ as we compute $\mathbf{u}_1^\mathsf{T}\mathbf{b}$ first. Thus, the overall complexity of computing $\mathbf{u}_2$ is in $O(|E|)$ given that $N < |E|$.

D Datasets

D.1 Real-world Datasets

The overall statistics of the real-world datasets are presented in Table D.1 and Table D.1 provides their heterophily levels calculated using various homophily metrics.

	#Nodes	#Edges	#Features	#Classes	Metric
cora	2,708	5,278	1,433	7	ACC
citeseer	3,327	4,552	3,703	6	ACC
pubmed	19,717	44,324	500	3	ACC
roman-empire	22,662	32,927	300	18	ACC
amazon-ratings	24,492	93,050	300	5	ACC
minesweeper	10,000	39,402	7	2	ROC AUC
tolokers	11,758	519,000	10	2	ROC AUC
questions	48,921	153,540	301	2	ROC AUC
squirrel-filtered	2,223	46,998	2,089	5	ACC
chameleon-filtered	890	8,854	2,325	5	ACC

Table 3: Statistics of the benchmark dataset. Following pre-processing, the graph has been transformed into an undirected and simple form, without self-loops or multiple edges.

	H_{node}	H_{edge}	H_{class}	$H_{\text{agg}}^{\text{M}}(\mathcal{G})$	H_{edge}^{*}	LI
cora	0.83	0.81	0.77	0.99	0.77	0.59
citeseer	0.71	0.74	0.63	0.97	0.67	0.45
pubmed	0.79	0.80	0.66	0.94	0.69	0.41
roman-empire	0.05	0.05	0.02	1.00	-0.05	0.11
amazon-ratings	0.38	0.38	0.13	0.60	0.14	0.04
minesweeper	0.68	0.68	0.01	0.61	0.01	0.00
tolokers	0.63	0.59	0.18	0.00	0.09	0.01
questions	0.90	0.84	0.08	0.00	0.02	0.00
squirrel-filtered	0.19	0.21	0.04	0.00	0.01	0.00
chameleon-filtered	0.24	0.24	0.04	0.25	0.03	0.01

Table 4: Heterophily levels of benchmark datasets. The $H_{\text{agg}}^{\text{M}}(\mathcal{G})$ stands for the aggregation homophily, calculated using $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. The H_{edge}^{*} stands for the adjusted edge homophily, and the LI stands for the label informativeness. The definitions of these metrics can be found in Appendix A.

D.2 Synthetic Datasets

In addition to the real-world datasets, we also tested parameterized diffusion on synthetic graphs generated with different homophily levels ranging from 0 to 1 using the method proposed in [20]. Here we give a review of the generation process.

More specifically, when generating the output graph $\mathcal{G}$ with a desired total number of nodes N, a total of C classes, and a homophily coefficient μ, the process begins by dividing the N nodes into C equal-sized classes. Then the synthetic graph $\mathcal{G}$ (initially empty) is updated iteratively. At each step, a new node v_i is added, and its class z_i is randomly assigned from the set $\{1, \ldots, C\}$. Whenever a new node v_i is added to the graph, we establish a connection between it and an existing node v_j in $\mathcal{G}$ based on the probability p_{ij} determined by the following rules:

$$p_{ij} = \begin{cases} d_j \times \mu, & \text{if } z_i = z_j \\ d_j \times (1-\mu) \times w_{d(z_i,z_j)}, & \text{otherwise} \end{cases}. \tag{32}$$

where z_i and z_j are class labels of node i and j respectively, and $w_{d(z_i,z_j)}$ denotes the "cost" of connecting nodes from two distinct classes with a class distance of $d(z_i, z_j)$. For a larger μ, the chance of connecting with a node with the same label increases. The distance between two classes simply implies the shortest distance between the two classes on a circle where classes are numbered from 1 to C. For instance, if $C = 6$, $z_i = 1$ and $z_j = 5$, then the distance between z_i and z_j is 2. The weight exponentially decreases as the distance increases and is normalized such that $\sum_d w_d = 1$. In addition, the probability p_{ij} defined in Eq. (32) is also normalized over the exiting nodes:

$$\bar{p}_{ij} = \frac{p_{ij}}{\sum_{k:v_k \in \mathcal{N}(v_i)} p_{ik}}$$

Lastly, the features of each node in the output graph are sampled from overlapping 2D Gaussian distributions. Each class has its own distribution defined separately.

E Hyperparameters

The following table lists the optimal γ and α used in models with the proposed methods.

		roman-empire	amazon-ratings	minesweeper	tolokers	questions	squirrel-filtered	chameleon-iltered
GCN (R(G))	γ	0.1	0.3	0.1	0.9	0.1	0.1	0.1
	α	0.6	0.3	0.3	0.9	0.2	0	0
GAT (R(G))	γ	0.8	0.2	0.2	0.5	0.5	0.2	0.1
	α	0.7	0	0.3	0.9	1	0.9	1
PD-GCN	γ	1	0.9	1	0.6	0.7	1	0.9
	α	0	0.9	1	0.4	0.8	0.1	0
PD-GAT	γ	0	0.9	0.4	1	0.1	0.4	0.6
	α	0	0.9	0.2	1	0.4	0.3	0.2
PD-GAT (R(G))	γ	0.5	0.4	0.3	1	0.5	0.7	0.8
	α	0.9	1	0	0.7	1	0.4	0.5
PD-GAT-sep	γ	0.9	0.1	0.9	0.5	0.2	0.5	0.1
	α	0.4	0.9	0	1	0.2	0.3	0.9
PD-GAT-sep (R(G))	γ	0.6	0.6	0.3	0.5	0.3	0.5	0.2
	α	0.8	0.2	0	0.9	0.4	0.9	0.9

Table 5: Optimal γ and α for real-world datasets.

For BernNet, LINKX, APPNP and ωGCN in real-world benchmark, we perform a grid search for learning rate $\in \{0.01, 0.05, 0.1\}$, weight decay $\in \{0, 5e-7, 5e-6, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2\}$, dropout $\in \{0, 0.1, 0.3, 0.5, 0.7\}$. Model specific parameters are: (1) BernNet: the propagation steps $K = 10$; (2) LINKX: the numbder of layers of MLP_A and MLP_X in $\{1, 2\}$; (3) APPNP: $\alpha \in \{0.1, 0.2, 0.5, 0.9\}$ and up to 10^{th} power of the adjacency is used; (4) PGSO: initialization $\in$ { "GCN", "all zeros", "SymLaplacian", "RWLaplacian", "Adjacency"}. According to [12], the weight decay is $5e-4$, the learning rate is 0.005 for the exponential parameters and 0.01 for all other model parameters. We perform a grid search for dropout $\in \{0, 0.1, 0.3, 0.5, 0.7\}$; (5) DGN: We compute the first two non-trivial eigenvectors of $\mathbf{L}_\text{rw}$ for DGN. We perform a grid search for the hyperparameters of DGN according to [8] for the learning rate in $\{10^{-5}, 10^{-4}\}$, the weight decay in $\{10^{-6}, 10^{-5}\}$, the dropout rate in $\{0.3, 0.5\}$, the aggregator in {"mean-dir1-av", "mean-dir1-dx", "mean-dir1-av-dir1-dx" }, the net type in {"complex", "simple" }.

F Comparison with Message Passing with Virtual Nodes

This section further demonstrates the effectiveness of the parameterized diffusion with rewiring by comparing it to message passing with virtual nodes [51, 66], and the results are summarized in Table. The virtual node is an additional node attached to the original graph and connected to all nodes in the graph. While our rewiring strategy connects all nodes to the gradient node determined by $\mathbf{L}^{(\alpha,\gamma)}$. The difference between this and the virtual node approach is threefold: (1) the gradient node is selected from the original graph's nodes, while the virtual node an artificially added node; (2) the selection of the gradient node is determined by α and γ; and (3) the virtual node is incompatible with the proposed parameterized diffusion, as we cannot define the weights or features for edges connecting the virtual nodes based the first non-trivial eigenvector of the $\mathbf{L}^{(\alpha,\gamma)}$ from the original graph topology.

We apply the virtual nodes method to the baseline GNNs, with results provided in Table 6. The same experiment settings from Section 4.3 are used, and results for baselines and the proposed rewiring strategy are also reported in Table 2. In conclusion, the proposed rewiring approach yields a greater performance improvement over the baselines compared to the virtual node method.

	roman-empire	amazon-ratings	minesweeper	tolokers	questions	squirrel-filtered	chameleon-filtered
GCN	73.69 ± 0.74	48.70 ± 0.63	89.75 ± 0.52	83.64 ± 0.67	76.09 ± 1.27	39.47 ± 1.47	40.89 ± 4.12
GAT	80.87 ± 0.30	49.09 ± 0.63	92.01 ± 0.68	83.70 ± 0.47	77.43 ± 1.20	35.62 ± 2.06	39.21 ± 3.08
GAT-sep	88.75 ± 0.41	52.70 ± 0.62	93.91 ± 0.35	83.78 ± 0.43	76.79 ± 0.71	35.46 ± 3.10	39.26 ± 2.50
GCN-Virtual Nodes	74.47 ± 0.67	48.18 ± 0.64	89.92 ± 0.59	83.43 ± 0.92	77.26 ± 1.14	40.71 ± 2.67	43.94 ± 4.15
GAT-Virtual Nodes	79.35 ± 0.26	46.85 ± 0.56	92.62 ± 0.77	84.17 ± 0.70	78.47 ± 0.81	39.07 ± 1.81	43.47 ± 3.59
GAT-sep-Virtual Nodes	88.07 ± 0.61	48.92 ± 0.50	93.79 ± 0.46	84.11 ± 0.50	78.00 ± 0.96	37.76 ± 1.57	40.96 ± 3.54
PD-GAT ($\mathcal{R}(\mathcal{G})$)	87.27 ± 0.64	48.03 ± 0.58	93.27 ± 0.56	84.74 ± 0.59	79.55 ± 0.81	42.09 ± 2.65	44.16 ± 4.20
PD-GAT-sep ($\mathcal{R}(\mathcal{G})$)	89.23 ± 0.56	50.96 ± 0.43	94.03 ± 0.45	84.83 ± 0.40	78.88 ± 0.94	39.69 ± 2.28	41.15 ± 4.66

Table 6: Experiment results on heterophily datasets proposed by [50] for comparing baseline models using virtual nodes methods [51, 66] and the proposed parameterized diffusion with topology-guided rewiring.

G Comparison on Homophilic Datasets

The following table summarizes the results on homophilic datasets, where we perform a grid search for learning rate $\in \{0.01, 0.05, 0.1\}$, weight decay $\in \{0, 5e-7, 5e-6, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2\}$, dropout $\in \{0, 0.1, 0.3, 0.5, 0.7\}$, and 64 hidden states. Note that the implementation of baselines follows [50], where residual connections are adopted in each layer.

	Cora	Citeseer	Pubmed
ResNet	72.03 ± 0.24	70.77 ± 1.81	88.01 ± 0.41
GCN	86.15 ± 1.24	74.58 ± 0.97	89.63 ± 0.44
SAGE	85.12 ± 1.64	74.47 ± 1.93	89.69 ± 0.51
GAT	86.46 ± 1.02	74.13 ± 1.85	88.87 ± 0.65
GAT-sep	84.24 ± 1.72	73.93 ± 1.93	89.14 ± 0.57
GT	86.19 ± 0.99	74.23 ± 1.12	89.48 ± 0.52
GT-sep	86.3 ± 1.13	74.14 ± 0.91	89.63 ± 0.52
DGN	85.16 ± 1.17	72.70 ± 1.17	87.35 ± 0.53
ωGCN	86.13 ± 1.38	74.74 ± 1.39	88.65 ± 0.42
PGSO	88.66 ± 0.94	76.55 ± 1.12	89.44 ± 0.53
PD-GCN	86.91 ± 1.45	75.17 ± 1.24	89.70 ± 0.45
PD-GAT	85.40 ± 1.41	74.83 ± 1.75	89.48 ± 0.45

Table 7: Experiments on homophily datasets proposed in [67].

H Training

In training and evaluating a model using a node classification benchmark dataset with C distinct classes, each node $v_i \in \mathcal{V}$ has a label z_i associated with it. We denote $\mathbf{Z} \in \mathbb{R}^{N \times C}$ as the one-

hot encoding of labels. Moreover, nodes are divided into three sets: the training set $\mathcal{V}_{\text{train}}$, the validation set $\mathcal{V}_{\text{val}}$ and the test set $\mathcal{V}_{\text{test}}$. In the training phase, the model uses features of all nodes under transductive learning. The model only has access to labels of nodes in $\mathcal{V}_{\text{train}}$ and in $\mathcal{V}_{\text{val}}$ (for hyperparameter tuning), while labels of nodes in $\mathcal{V}_{\text{test}} = \mathcal{V} \setminus (\mathcal{V}_{\text{train}} \cup \mathcal{V}_{\text{val}})$ remain unknown to the model. The cost function used in node classification tasks is the standard categorical cross-entropy loss [26], which is commonly used for multi-class classification tasks:

$$\mathcal{L} = -\frac{1}{|\mathcal{V}_{\text{train}}|}\text{trace}(\mathbf{Z}^{\mathsf{T}} \log \mathbf{Y}), \tag{33}$$

where $\mathbf{Y}$ is the output from the model after softmax and $\log(\cdot)$ is applied element-wise.