Complete the Missing Half: Augmenting Aggregation Filtering with Diversification for Graph Convolutional Networks

Sitao Luan[1,2,*], **Mingde Zhao**[1,2,*], **Chenqing Hua**[1,2*], **Xiao-Wen Chang**[1], **Doina Precup**[1,2,3]
{sitao.luan@mail, mingde.zhao@mail, chenqing.hua@mail, chang@cs, dprecup@cs}.mcgill.ca
[1]McGill University; [2]Mila; [3]DeepMind
*Equal Contribution

Abstract

The core operation of current Graph Neural Networks (GNNs) is the *aggregation* enabled by the graph Laplacian or message passing, which filters the neighborhood node information. Though effective for various tasks, in this paper, we show that they are potentially a problematic factor underlying all GNN methods for learning on certain datasets, as they force the node representations similar, making the nodes gradually lose their identity and become indistinguishable. Hence, we augment the aggregation operations with their dual, *i.e.,* diversification operators that make the node more distinct and preserve the identity. Such augmentation replaces the aggregation with a two-channel filtering process that, in theory, is beneficial for enriching the node representations. In practice, the proposed two-channel filters can be easily patched on existing GNN methods with diverse training strategies, including spectral and spatial (message passing) methods. In the experiments, we observe desired characteristics of the models and significant performance boost upon the baselines on 9 node classification tasks. [1]

1 Introduction

As a generic data structure, graph is capable of modeling complex relations among objects in many real-world problems [32, 41, 13]. Motivated by the success of Convolutional Neural Networks (CNNs) [31] on images, graph convolution [54] is defined on the graph Fourier domain and the node spatial neighborhood domain [58], respectively, in the form of spectral- and spatial-based methods. Based on the 2 methodologies, different (linear) graph filters and (non-linear) deep learning techniques [30] are combined, giving rise to Graph Neural Networks (GNNs), achieving remarkable progress [6, 22, 19, 28, 52, 38, 37].

Most existing graph filters can be viewed as operators that aggregate node information from its direct neighbors. Different graph filters yield different spectral GNNs or spatial aggregation functions. Among them, the most commonly used is the *renormalized affinity matrix* [28]. By adding an identity matrix to the adjacency matrix, *i.e.,* a self-loop in the graph topology, renormalized affinity matrix is created as a low-pass (LP) filter [39] mainly capturing low-frequency signals, which are locally smooth features across the whole graph [53]. Aggregation processes, in the form of message passing used in spatial-based methods, as in *e.g.,* GraphSAGE [22] and GraphSAINT [57], are also node-level LP filters which make nodes become similar to their neighbors.

The main idea of neighborhood feature aggregation is to exploit the intrinsic geometry of the data distribution: if two data points are close (or connected) to each other on the manifold, they should

[1]See the follow-up work in [36, 37]

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

also be close to each other in the representation space. This assumption is usually referred to as manifold (local invariance) [1, 24, 8, 7], assortative mixing (assortativity) [42], homophily [40, 36, 33] or smoothness [27, 35] assumption, which plays an essential role in the development of various kinds of algorithms including dimensionality reduction [1] and semi-supervised learning [61][2]. This assumption naturally holds in many real world networks [40, 26], *e.g.,* social networks, citation networks, evolutionary biology *etc.*. However, in contrast to homophily, there also exists a large number of heterophily networks where individuals with diverse characteristics tend to gather in the same group [45], *e.g.,* dating networks [63] and fraudsters in online purchasing networks [43]. On these networks, there is no strong reason to impose smoothness assumption and on the contrary, non-smoothness pattern between nodes turns out to be important.

With the above in mind, in this paper we first propose a method to measure the smoothness of the input features and output labels of an attributed graph based on Dirichlet energy and graph signal energy. With the proposed method, we measure the smoothness of 9 real world datasets, which shows that signal defined on graph is generally a mixture of smooth and non-smooth graph signals and each part plays an indispensable role. Motivated by this discovery, we argue that, to learn richer representations, the distinctive information between nodes should also be extracted. Hence, we design a two-channel filterbank (FB) [14] GNN framework which use low-pass (LP) and high-pass (HP) filters together to learn the smooth and non-smooth components, respectively. FB-GNN framework can easily be plugged into spatial methods, with LP filter for aggregation operation and HP filter for diversification operation. With experiments on 9 real world datasets, we find that the the HP channel indeed plays and important role in the representation learning, and one-channel baseline methods can gain significant performance boost after being augmented by two-channel methods.

2 Preliminaries

After introducing the prerequisites, in this section, we formalize the idea behind graph signal filtering. We use bold fonts for vectors (*e.g.,* $\boldsymbol{v}$), vector blocks (*e.g.,* $\boldsymbol{V}$) and matrix blocks (*e.g.,* $\boldsymbol{V}_i$). Suppose we have an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ without bipartite component, where $\mathcal{V}$ is the node set with $|\mathcal{V}| = N$, $\mathcal{E}$ is the edge set, $A \in \mathbb{R}^{N\times N}$ is a symmetric adjacency matrix with $A_{ij} = 1$ if and only if $e_{ij} \in \mathcal{E}$ otherwise $A_{ij} = 0$, D is the diagonal degree matrix, *i.e.,* $D_{ii} = \sum_j A_{ij}$ and $\mathcal{N}_i = \{j : e_{ij} \in \mathcal{E}\}$ is the neighborhood set of node i. A graph signal is a vector $\boldsymbol{x} \in \mathbb{R}^N$ defined on $\mathcal{V}$, where x_i is defined on the node i. We also have a feature matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ whose columns are graph signals and each node i has a feature vector $\boldsymbol{X}_{i,:}$ with dimension F, which is the i-th row of $\boldsymbol{X}$.

2.1 Graph Laplacian and Affinity Matrix

The (Combinatorial) graph Laplacian is defined as $L = D - A$, which is a Symmetric Positive Semi-Definite (SPSD) matrix[11]. Its eigendecomposition gives $L = U\Lambda U^T$, where the columns of $U \in \mathbb{R}^{N\times N}$ are orthonormal eigenvectors, namely the *graph Fourier basis*, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ with $\lambda_1 \leq \cdots \leq \lambda_N$ and these eigenvalues are also called *frequencies*. The graph Fourier transform of the graph signal $\boldsymbol{x}$ is defined as $\boldsymbol{x}_{\mathcal{F}} = U^{-1}\boldsymbol{x} = U^T\boldsymbol{x} = [\boldsymbol{u}_1^T\boldsymbol{x}, \ldots, \boldsymbol{u}_N^T\boldsymbol{x}]^T$, where $\boldsymbol{u}_i^T\boldsymbol{x}$ is the component of $\boldsymbol{x}$ in the direction of $\boldsymbol{u_i}$.

A smaller λ_i indicates a smoother basis function $\boldsymbol{u_i}$ defined on $\mathcal{G}$ [12], which means any two elements of $\boldsymbol{u_i}$ corresponding to two connected nodes will have more similar values. This is because finding the eigenvalues and eigenvectors of graph Laplacian is actually solving a series of conditioned minimization problems relevant to the smoothness of the function defined on $\mathcal{G}$.

Some variants of graph Laplacians are commonly used in practice, *e.g.,* the symmetric normalized Laplacian $L_{\text{sym}} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}AD^{-1/2}$, the random walk normalized Laplacian $L_{\text{rw}} = D^{-1}L = I - D^{-1}A$. L_{rw} and L_{sym} share the same eigenvalues, which are inside $[0, 2)$, and their corresponding eigenvectors satisfy $\boldsymbol{u}^i_{\text{rw}} = D^{-1/2}\boldsymbol{u}^i_{\text{sym}}$.

The affinity (transition) matrix derived from L_{rw} is defined as $A_{\text{rw}} = I - L_{\text{rw}} = D^{-1}A$ and its eigenvalues $\lambda_i(A_{\text{rw}}) = 1 - \lambda_i(L_{\text{rw}}) \in (-1, 1]$. Similarly, $A_{\text{sym}} = I - L_{\text{sym}} = D^{-1/2}AD^{-1/2}$ is an affinity matrix as well. Renormalized affinity matrix is introduced in [28] and defined as

[2] In this paper, we do not distinguish the name of this assumption.

$\hat{A}_{\text{rw}} = \tilde{D}^{-1}\tilde{A}$, where $\tilde{A} \equiv A + I$, $\tilde{D} \equiv D + I$ and $\lambda(\hat{A}_{\text{rw}}) \in (-1, 1]$. It essentially defines a random walk matrix on $\mathcal{G}$ with a self-loop added to each node in $\mathcal{V}$ and is widely used in GCN as follows,

$$\boldsymbol{Y} = \text{softmax}(\hat{A}_{\text{rw}}\ \text{ReLU}(\hat{A}_{\text{rw}}\boldsymbol{X}W_0)\ W_1) \tag{1}$$

where $W_0 \in \mathbb{R}^{F \times F_1}$ and $W_1 \in \mathbb{R}^{F_1 \times O}$ are parameter matrices. $\hat{L}_{\text{rw}}$ can be defined as $I - \hat{A}_{\text{rw}}$. $\hat{A}_{\text{sym}} \equiv \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ can also be applied in GCN and $\hat{L}_{\text{sym}} = I - \hat{A}_{\text{sym}}$. Specifically, the nature of transition matrix makes $\hat{A}_{\text{rw}}$ behave as a mean aggregator $(\hat{A}_{\text{rw}}\boldsymbol{x})_i = \sum_{j \in \{\mathcal{N}_i \cup i\}} x_j/(D_{ii} + 1)$ which is applied in [22] and is important to bridge the gap between spatial- and spectral-based graph convolution methods.

2.2 Measure of Smoothness and (Dirichlet) Energy

Dirichlet Energy is often used to measure how variable a function is [17] and for signal defined on graph, it can measure the global smoothness of the signal [48, 5, 49] and is defined as follows.

Definition 1. *(Dirichlet Energy) The Dirichlet energy of vector block $\boldsymbol{X}$ and column vector $\boldsymbol{x}$ defined on $\mathcal{G}$ are separately defined as*

$$E_S^{\mathcal{G}}(\boldsymbol{X}) = tr(\boldsymbol{X}^T L \boldsymbol{X}),\ \ E_S^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^T L \boldsymbol{x} \tag{2}$$

Note that $E_S^{\mathcal{G}}$ is always non-negative since L is SPSD. The graph signal energy is defined as follows.

Definition 2. *(Graph Signal Energy [18, 51]) The signal energy of block vector $\boldsymbol{X}$ and column vector $\boldsymbol{x}$ defined on undirected graph $\mathcal{G}$ are separately defined as*

$$E^{\mathcal{G}}(\boldsymbol{X}) = tr(\boldsymbol{X}^T \boldsymbol{X}),\ \ E^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{x} \tag{3}$$

The signal energy represents the amount of contents in a graph signal and we will draw the correlation between $E_S^{\mathcal{G}}$ and $E^{\mathcal{G}}$ and explain how they can be used to measure the smoothness and non-smoothness of a graph (block) signal.

Take column vector $\boldsymbol{x}$ for example, $E_S^{\mathcal{G}}(\boldsymbol{x})$ can be written as,

$$\boldsymbol{x}^T L \boldsymbol{x} = \sum_i \lambda_i (u_i^T \boldsymbol{x})^T u_i^T \boldsymbol{x} = \sum_i \lambda_i \left\| u_i^T \boldsymbol{x} \right\|_2^2$$

The frequency λ_i before $\left\| u_i^T \boldsymbol{x} \right\|_2^2$ can be considered as a scalar weight and $E_S^{\mathcal{G}}(\boldsymbol{x})$ focuses on measuring the component of $\boldsymbol{x}$ in the direction of non-smooth $\boldsymbol{u_i}$, who has a large weight λ_i. A small $E_S^{\mathcal{G}}(\boldsymbol{x})$ means $\boldsymbol{x}$ does not contain much non-smooth components. $E^{\mathcal{G}}(\boldsymbol{x})$ can be written as

$$\boldsymbol{x}^T \boldsymbol{x} = \sum_i (u_i^T \boldsymbol{x})^T u_i^T \boldsymbol{x} = \sum_i \left\| u_i^T \boldsymbol{x} \right\|_2^2$$

Signal $\boldsymbol{x}$ can be decomposed into smooth and non-smooth components, and the amount the non-smooth component can be measured by

$$E_{NS}^{\mathcal{G}}(\boldsymbol{x}) = E^{\mathcal{G}}(\boldsymbol{x}) - E_S^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^T (I - L) \boldsymbol{x} = \sum_i (1 - \lambda_i) \left\| u_i^T \boldsymbol{x} \right\|_2$$

Note that $E_{NS}^{\mathcal{G}}(\boldsymbol{x})$ can be negative and a small $E_{NS}^{\mathcal{G}}(\boldsymbol{x})$ indicates that $\boldsymbol{x}$ is highly non-smooth.

Upon the above analysis, we define $S(\boldsymbol{x})$ to measure the smoothness of a signal as follows

$$S(\boldsymbol{x}) = \frac{E_S^{\mathcal{G}}(\boldsymbol{x})}{E^{\mathcal{G}}(\boldsymbol{x})},\ \ S(\boldsymbol{X}) = \frac{E_S^{\mathcal{G}}(\boldsymbol{X})}{E^{\mathcal{G}}(\boldsymbol{X})} \tag{4}$$

Graph signal with a small S-value means it is a smooth function define on $\mathcal{G}$. S can be different depends on the Laplacian we use to train GNN and S can be larger than 1. In this paper, we use L_{sym} and $\hat{L}_{\text{sym}}$ to measure the smoothness of input features $\boldsymbol{X}$ and labels $\boldsymbol{y}$ for different GNNs.

3 Filterbanks in GNNs: From One Channel to Two

In this section, we state why it is necessary to switch to the two-channel filtering process from only one-channel. Then, we propose the filterbank-GNN framework which can learn a mixture of smooth and non-smooth graph signals.

3.1 Motivation

Table 1: Dataset Overview: Network Characteristics and S-values measured by L_{sym}

datasets		Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
Network Info	#nodes	183	251	183	7600	2277	5201	2708	3327	19717
	#edges	295	499	309	33544	36101	217073	5429	4732	44338
	#features	1703	1703	1703	931	2325	2089	1433	3703	500
	#classes	5	5	5	5	5	5	7	6	3
S-values	input feature	0.904	0.873	0.854	0.901	0.99	0.987	0.862	0.799	0.832
	label	0.883	0.877	0.909	0.836	0.747	0.782	0.288	0.35	0.501
	diff (label - feature)	-0.021	0.004	0.055	-0.065	-0.243	-0.205	-0.574	-0.449	-0.331

We use blue and red shades to demonstrate the relation between label and feature: the label of the blue shaded datasets is smoother than its feature and red datasets are less smooth.

We measure the smoothness of 9 frequently used benchmark datasets and present the results with the network characteristics for each task in Table 1). It shows that the input features and ground truth labels of different datasets are all mixtures of smooth and non-smooth graph signals but in different proportions. Besides, it illustrates that different tasks have different demands of learning how to smoothen the input signals. For example, in *Cora*, *Citeseer* and *Pubmed*, the ground truth labels are much smoother than the input features, such pattern motivates us to learn how to smoothen the input signals; while in *Wisconsin* and *Texas*, the labels are less smooth than the input features, thus there is no reason that we still learn how to smoothen the input signals. Therefore, to accommodate different situations, we propose that we should learn both smooth and non-smooth components of the input features adaptively instead of merely extracting the smooth part. This motivates us to use filterbanks (LP and HP filters) to filter the signals in GNNs.

LP, HP Graph Filters and Filter Banks The multiplication of L and $\boldsymbol{x}$ acts as a filtering operation over $\boldsymbol{x}$, adjusting the scale of the components of $\boldsymbol{x}$ in frequency domain. To see this, consider

$$\boldsymbol{x} = \sum_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}, \;\; L\boldsymbol{x} = \sum_i \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x} \tag{5}$$

The projection $\boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}$ corresponding to a large $|\lambda_i|$ will be amplified, while the one corresponding to a small $|\lambda_i|$ will be suppressed. More specifically, a graph filter that filters out smooth (non-smooth) components is called HP (LP) filter. Generally, the Laplacian matrices (L_{sym}, L_{rw}, $\hat{L}_{sym}$, $\hat{L}_{rw}$) can be regarded as HP filters [14] and affinity matrices (A_{sym}, A_{rw}, $\hat{A}_{\text{sym}}$, $\hat{A}_{\text{rw}}$) can be treated as LP filters [39]. In general, we denote HP and LP filters as L_{HP} and L_{LP} respectively.

On the node level, left multiplying HP and LP filters on $\boldsymbol{x}$ can be understood as diversification and aggregation operations, respectively. For example, if we implement L_{rw} and A_{rw} on the i-th node, we have

$$(L_{\text{rw}}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_i} \frac{1}{D_{ii}}(x_i - x_j), \;\; (A_{\text{rw}}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_i} \frac{1}{D_{ii}} x_j \tag{6}$$

Intuitively, HP filters depict the differences between one node and its neighbors; While LP filters focus on the similarity within a neighborhood, from which we can obtain missing or "hidden" features of one node. We believe that these two conjugate components are both indispensable to portray a node.

Mathematically, multiplying with LP filter (aggregation) is a linear projection, which will project the features to a fixed subspace. We will lose the expressive power by only using LP filter, and the missing half is the HP component of the learned signals, as $L_{\text{LP}} + L_{\text{HP}} = I$, which satisfies the perfect reconstruction property [14].

The two-channel linear filterbank which contains a set of filters L_{LP} and L_{HP} is widely used in graph signal processing [15, 14], but are rarely used in graph neural networks. Inspired by this technique,

we propose the two-channel filterbank GNNs in section 3.2, which can extract both smooth and non-smooth components from input features.

3.2 Filter Bank assisted GNNs (FB-GNNs)

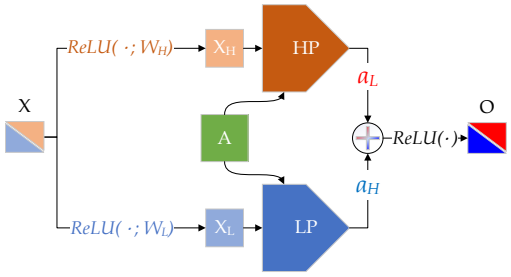


Figure 1: Two-Channel Learning: Information needed for HP and LP filters, X_H and X_L, are separately extracted from the input signal X by nonlinear transformations. After being filtered by HP and LP, which are both derived upon adjacency matrix A, the filtered signals are again recombined adaptively to form the output O.

Spectral-based FB-GNNs We use previously defined L_{LP} and L_{HP} (see section 3.1) to construct the two-channel FB-GNNs as follows (learning framework is provided in figure 1)

$$\begin{aligned} \boldsymbol{H}_L^l &= L_{\text{LP}}\text{ReLU}(\boldsymbol{H}^{l-1}W_L^{l-1}),\ \boldsymbol{H}_H^l = L_{\text{HP}}\text{ReLU}(\boldsymbol{H}^{l-1}W_H^{l-1}) \\ \boldsymbol{H}^l &= \text{ReLU}\left(\alpha_L^l \cdot \boldsymbol{H}_L^l + \alpha_H^l \cdot \boldsymbol{H}_H^l\right),\ \ l = 1, \ldots, n \end{aligned} \tag{7}$$

where $\boldsymbol{H}^0 = \boldsymbol{X}$; $W_L^{l-1}, W_H^{l-1} \in \mathbb{R}^{F_{l-1} \times F_l}$ are learnable parameter matrices for the non-linear feature extractor focusing on disentangling the smooth and non-smooth information from input $\boldsymbol{H}^{l-1}$, separately; $\alpha_L^l, \alpha_H^l \in [0, 1]$ are learnable scalar parameters which can learn the relative importance of $\boldsymbol{H}_L^l$ and $\boldsymbol{H}_H^l$ and keep a balance between them; l is the layer number and suppose the FB-GNN has n layers. In this way, the hidden output $\boldsymbol{H}^l$ is able to learn a mixture of smooth and non-smooth signals.

Spatial-based FB-GNNs Inspired by (6) and (7), the two-channel spatial-based method can be implemented by designing different aggregator (LP filter) and diversification operator (HP filter) as follows,

$$\begin{aligned} (\hat{\boldsymbol{h}}_i^l)_L &= \text{ReLU}(W_L^{l-1}\boldsymbol{h}_i^{l-1}),\ \ (\hat{\boldsymbol{h}}_i^l)_H = \text{ReLU}(W_H^{l-1}\boldsymbol{h}_i^{l-1}) \\ (\boldsymbol{h}_i^l)_L &= \sum_{j \in \{\mathcal{N}_i \cup i\}} \boldsymbol{w}_{ij} \left((\hat{\boldsymbol{h}}_i^l)_L + (\hat{\boldsymbol{h}}_j^l)_L\right),\ (\boldsymbol{h}_i^l)_H = \sum_{j \in \{\mathcal{N}_i \cup i\}} \boldsymbol{w}_{ij} \left((\hat{\boldsymbol{h}}_i^l)_H - (\hat{\boldsymbol{h}}_j^l)_H\right), \\ \boldsymbol{h}_i^l &= \text{ReLU}\left(\alpha_L^l \cdot (\boldsymbol{h}_i^l)_L + \alpha_H^l \cdot (\boldsymbol{h}_i^l)_H\right),\ , i \in \mathcal{V},\ l = 1, \ldots, n \end{aligned} \tag{8}$$

where $W_L^{l-1}, W_H^{l-1} \in \mathbb{R}^{F_l \times F_{l-1}}$ are learnable parameter matrices to extract LP and HP features for two channels; $\boldsymbol{w}_{ij}$ is the connection weight between node i and node j derived from adjacency matrix, it can be a fixed value or a learnable attention coefficient such as [52]; $\alpha_L^l, \alpha_H^l \in [0, 1]$ are learnable scalar parameters; l is the layer number.

Computational Cost: Parameters and Runtime The spectral two-channel learning introduces additionally one GCN operation and one weighted sum (with negligible costs introduced with non-linearity and weighted sum before output); For spatial methods, similarly, the two-channel learning introduces one additional node-wise subtraction and one additional weighted sum for training on each pair of nodes. Thus, the computational cost and the number of parameters are approximately doubled;

For runtime, overlooking the minor overhead of synchronization, the computations introduced by the additional pass are naturally parallelizable with the original pass (for their independently

associated parameters) both in the forward and backward passes. Therefore, no significant additional computational time will be incurred on modern GPU architectures.

4 Related Works

Dirichlet energy Dirichlet energy (more generally in p-Dirichlet form) is usually used as a regularizer or objective function to impose local neighborhood smoothness in various machine learning tasks, *e.g.,* spectral clustering [1], image processing [16, 4, 60], non-negative matrix factorization [7], matrix completion, principal component analysis (PCA), semi-supervised learning [64, 65, 2]. It has different names in different literature, *e.g.,* Laplacian regularizer [60], manifold regularizer [7], quadratic energy function [64], *etc.*. The definition of smoothness derived from Dirichlet energy in (4) can be considered as a continuous relaxed form of normalized cut (ratio cut) [20, 50], which is closely related to graph partition problems. Instead of using Dirichlet energy as a part of loss function during training process, we point out that combining with graph signal energy, it can be used to measure the smoothness of the input features and output labels for a given learning task. With this, the necessity of learning the non-smoothness component can be confirmed.

Measuring Smoothness The authors of [44] propose a node homophily to measure the smoothness of ground truth labels of dataset as follows,

$$\frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\frac{\#v \text{ 's neighbors who have the same label as } v}{\#v \text{ 's neighbors}}$$

[63] proposes edge homophily ratio, which is the fraction of edges that the connected nodes share the same label (*i.e.,*, intra-class edges). Both of these methods do not provide an extension definition on block vector. Thus, they fail to measure the smoothness of the input features and cannot be used to compare the difference of smoothness between the input features and labels. [59] proposes row-diff and col-diff to measure the average of all pairwise distances between the node features and the average of pairwise distances between columns of the representation matrix. But quantifying pairwise distance is inconsistent with the definition of smoothness introduced in section 1 which focuses on measuring the distance between connected nodes. [37] studies homophily from post-aggregation node similarity perspective. [35] uses statistical hypothesis testing to detect the effect the edge bias.

On Addressing Heterophily Geom-GCN [44] uses a geometric aggregation scheme and a bi-level aggregator to capture the information of structural neighborhoods, which can be distant nodes. These can efficiently take use of the geometric relationships defined in the latent space. H_2GCN [63] designs ego- and neighbor-embedding separation, aggregation of higher-order neighborhoods, and combination of intermediate representations to generalize the limitation of existing GNNs beyond homophily setting. Non-local GNNs [34] propose a simple and effective non-local aggregation framework with an efficient attention-guided sorting for GNNs. CPGNN [62] models label correlations through a compatibility matrix, which is beneficial for heterophilic graphs, and propagates a prior belief estimation into the GNN by using the compatibility matrix. FAGCN [3] learns edge-level aggregation weights as GAT [52] but allows the weights to be negative, which enables the network to capture high-frequency components in the graph signals. GPRGNN [10] uses learnable weights that can be both positive and negative for feature propagation. This allows GPRGNN to adapt to heterophilic graphs and to handle both high- and low-frequency parts of the graph signals. BernNet [23] designs a scheme to learn arbitrary graph spectral filters with Bernstein polynomial to address heterophily.

The aforementioned works design various tricks, trying to take use of multi-hop neighborhood information and capture long-range dependencies with the belief that heterophily problem could be alleviated with the help of the distant nodes. Although these methods show some promising results, the effectiveness is limited and do not jump out of the scope of neighborhood aggregation. In this paper, We target directly its cause, handling heterophily problem by seeking the distinctiveness between nodes with an additional channel to learn the non-smoothness components.

5 Experiments

In this section, we first validate whether the two-channel filtering and learning procedure lead to better representation learning when patched on popular shallow GNN baselines[3]: GraphSAINT [57], GraphSAGE [22], Graph Attention Network (GAT) [52], GCN [28], Geom-GCN-P (-S and -I) [44] and Graph Wavelet Neural Network (GWNN) [55]. Deeper GNNs are shown to have the potentials of mitigating the heterophily problem by extracting multi-hop neighborhood information. For them, we test the two-channel framework on two state-of-the-art methods GCNII and GCNII* [9], with varied model depths. After these, we validate the effectiveness of each proposed component with a detailed ablation test.

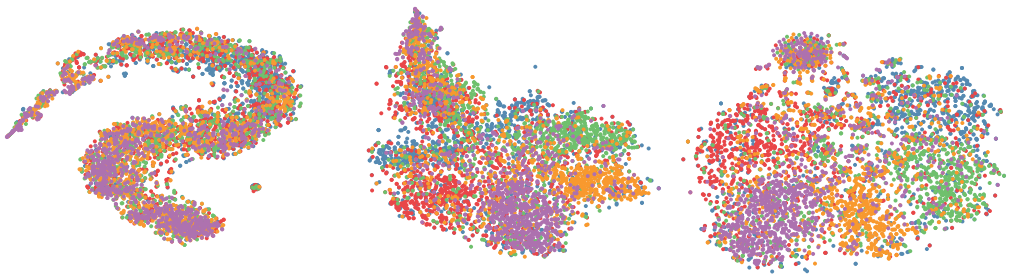
The experiments are conducted in the form of node classification[4] under supervised learning setting and performed on 9 datasets including *Cornell*, *Wisconsin*, *Texas*, *Actor*, *Chameleon*, *Squirrel*, *Cora*, *Citeseer*, and *Pubmed* (details to be found in the appendix). Their rough characteristics are shown in Table 1.

5.1 Experimental Setup

In supervised learning of shallow GNNs, we keep the same training configurations for GraphSAINT and FB-GraphSAINT on the 9 datasets, which are the random walk sampler with length 2 (RW) setting[5] in GraphSAINT [57]; for GWNN [55] and FB-GWNN, we use the same hyperparameters $s = 1.0, t = 10^{-4}$ on the 9 datasets[6]. Other GNNs and their two-channel variants are under the same experiment settings as [22] and [44]. We use $\hat{A}_{\text{sym}}$ as low-pass spectral filter and $\hat{L}_{\text{sym}}$ as high-pass spectral filter [7].

For supervised learning on deep GNNs, GCNII, GCNII*, FB-GCNII, and FB-GCNII* use $\lambda = 1.5$ and $\alpha = 0.2$ on $Actor$ and $Squirrel$. Other deep models use the same training configurations as GCNII [9] on the remaining 7 datasets.

For all experiments, we use the same 48%/ 32%/ 20% splits for training, validation and testing as in [44]. We report the average performance of all models on the test sets over 10 splits[8]. We tune the learning rate in {0.01, 0.05, 0.1}, weight decay in {0, 5e-6, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2}, and dropout in {0, 0.1, 0.2, . . . , 0.9}.



(a) Low-Pass Channel (b) High-Pass Channel (c) Two Channels Combined

Figure 2: t-SNE Visualization of the Learned Node Embeddings in Different Channels of FB-GCN for Squirrel Dataset.

[3] Source code submitted within supplementary materials and to be published after the review.

[4] See Appendix B for experimental results on graph classification tasks.

[5] The name "random walk sampler with length 2 setting" is what the authors used in their paper and they use the name PPI-large-2 in their code.

[6] This set of hyperparameters is the same as that of the original paper when training GWNN on Cora.

[7] We will discuss other filters in Appendix E

[8] We obtain the performance of GAT, GCN, and GEOM-GCN-P(-S and -I) directly from [44]; and we reproduce other baseline models and implement all the filterbank GNNs.

Table 2: Supervised Learning of Shallow GNNs

Models/Datasets	Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
Diff of S-values	-0.021	0.004	0.055	-0.065	-0.243	-0.205	-0.574	-0.449	-0.330
Spatial Methods(%)									
GraphSAINT	70.27	71.35	72.97	17.89	43.86	33.27	84.69	73.2	89.42
FB-GraphSAINT	78.38(8.11)	80(8.65)	75.68(2.71)	19.08(1.19)	46.05(2.19)	36.06(2.79)	87.5(2.81)	74.76(1.56)	89.88(0.46)
GraphSAGE	54.05	66	56.76	14.67	40.13	24.14	80.64	72.36	85.49
FB-GraphSAGE	63.14(9.09)	70(4)	58.05(1.29)	23.27(8.6)	39.74(-0.39)	24.6(0.46)	83.7(3.06)	72.58(0.22)	86.31(0.82)
Spectral Methods(%)									
GAT	54.32	49.41	58.38	28.45	42.93	30.03	86.37	74.32	87.62
FB-GAT	64.86(10.54)	60.78(11.37)	64.86(6.48)	30.66(2.21)	47.37(4.44)	31.8(1.77)	**88.73(2.36)**	77.12(2.8)	88.16(0.54)
GCN	52.7	45.88	52.16	26.86	28.18	23.96	85.77	73.68	88.13
FB-GCN	62.16(9.46)	56.86(10.98)	62.16(10.00)	31.21(4.35)	32.89(4.71)	24.73(0.77)	85.92(0.15)	75.24(1.56)	88.54(0.41)
Geom-GCN-P	60.81	64.12	67.57	**31.63**	60.9	38.14	84.93	75.14	88.09
FB-Geom-GCN-P	64.86(4.05)	72.55(8.43)	70.27(2.70)	31.02(-0.61)	**67.20(6.30)**	**49.66(11.52)**	85.17(0.24)	76.23(1.09)	88.25(0.16)
Geom-GCN-S	55.68	56.67	59.73	30.3	59.96	36.24	85.27	74.71	84.75
FB-Geom-GCN-S	56.54(0.86)	56.94(0.27)	62.16(2.43)	31.25(0.95)	61.49(1.53)	37.27(1.03)	85.43(0.16)	75.21(0.5)	85.88(1.13)
Geom-GCN-I	56.76	58.24	57.58	29.09	60.31	33.32	85.19	**77.99**	90.05
FB-Geom-GCN-I	57.38(0.62)	60.68(2.44)	62.21(4.63)	31.45(2.36)	60.76(0.45)	35.27(1.95)	85.45(0.26)	77.69(-0.3)	**90.48(0.43)**
GWNN	70.67	72.22	69.44	20.92	33.63	29.13	84.49	72.47	83.6
FB-GWNN	**80.11(9.44)**	**84.67(12.45)**	**77.78(8.34)**	22.24(1.32)	37.36(3.73)	30.6(1.47)	85.6(1.11)	72.83(0.36)	85.92(2.32)
Baseline Average	59.41	60.49	62.32	26.2	46.46	31.44	85.15	74.33	87.3
FB-Baseline Average	65.93(6.52)	67.81(**7.32**)	66.65(4.33)	27.52(1.32)	49.11(2.65)	33.75(2.31)	85.94(1.27)	75.21(0.97)	87.93(0.63)

The results are averaged from 10 independent runs. The (values) represent the difference of performance brought by patching FB.

Table 3: Statistics of Datasets (measured by $\hat{L}_{\text{sym}}$ instead of L_{sym}) and Comparison of the Output Smoothness

	datasets	Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	Citeseer	Pubmed
S-values	input feature	0.172	0.385	0.205	0.567	0.831	0.87	0.617	0.515	0.529
	label	*0.139*	*0.328*	*0.301*	*0.511*	*0.638*	*0.681*	*0.188*	*0.209*	*0.272*
	diff (label - feature)	-0.033	-0.057	0.096	-0.056	-0.193	-0.189	-0.429	-0.306	-0.257
	GCN output	*0.037 (0.102)*	*0.124 (0.204)*	*0.139 (0.162)*	*0.397 (0.114)*	*0.595 (0.043)*	*0.578 (0.103)*	*0.156 (0.032)*	*0.112 (0.097)*	*0.234 (0.038)*
	FB-GCN output	***0.099 (0.040)***	***0.269 (0.059)***	***0.201 (0.100)***	***0.531 (0.020)***	***0.655 (0.017)***	***0.683 (0.002)***	***0.172 (0.016)***	***0.148 (0.061)***	***0.247 (0.025)***

These results are obtained from 10 independent runs. The stds are negligible so they are not presented (mostly < 0.002).This table shows how FB- patched baseline could better reconstruct the label smoothness, *i.e.,* we want the S-value of the output to be closer to that of the labels. The (values) stand for the absolute difference between the S-values of the output of the methods and those of the ground truths. Better reconstruction between the two methods on each task is marked **bold**.

5.2 Supervised Learning of Shallow GNNs

In Table 2, we summarize the mean accuracy of shallow baseline GNNs and their filterbank versions. The best performance is highlighted. Also, we record the performance differences between baselines and the two-channel augmented methods in the brackets. For better intuitive understanding of the effectiveness of incorporating the high-pass filter, we present in Figure 2 the t-SNE visualization of the learned embedding in different channels of FB-GCN for Squirrel dataset. Those of the other datasets will be provided in the appendix D.

From the results we can see that, our propose methods generally boost the performance of almost all cases, especially when the labels are not much smoother than the input features indicated, considering the S-values in Table 3.

Table 3 shows that the S-values of FB-GCN outputs are closer to the S-values of the ground truth labels (see the absolute differences in the bracket) compared with those of GCN outputs. This indicates that FB-GCN is able to learn better representations which can reconstruct both the smooth and non-smooth part of the ground truth labels. Note that we measure the smoothness by $\hat{L}_{\text{sym}}$ instead of L_{sym}, because GCN is train with renormalized affinity matrix $\hat{A}_{\text{sym}}$.

5.3 Supervised Learning of Deep GNNs

In this subsection, we build deep multi-hop filterbank models based on the architecture of GCNII and GCNII* [9] to see if the two-channel method is capable to assist deep GNN models. We report mean accuracy, highlight best performing depth, and record performance difference in brackets in Table 4. In general, FB-GCNII and FB-GCNII* achieve better results than the unpatched GCNII and GCNII* at different depths, especially on *Wisconsin* and *Texas*, where the non-smooth part of representations are desirable.

Table 4: Supervised Learning of Deep Multi-scale GNNs

Models\Datasets	Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
GCNII-8	70.54	73.88	71.08	33.7	60.61	37.49	85.69	75.54	88.62
FB-GCNII-8	75.95(5.41)	82.35(8.47)	74.59(3.51)	35.37(1.67)	60.43(-0.18)	39.69(2.2)	86.04(0.35)	75.51(-0.03)	89.97(1.35)
GCNII-16	74.86	74.12	69.46	33.62	55.48	35.98	87.3	76.54	88.28
FB-GCNII-16	**77.57(2.71)**	82.55(8.43)	77.03(7.57)	35.12(1.5)	56.78(1.3)	39.38(3.4)	87.5(0.2)	76.67(0.13)	89.39(1.11)
GCNII-32	72.7	70.2	69.46	31.61	53.71	35.92	88.13	76.08	87.89
FB-GCNII-32	72.81(0.11)	78.63(8.43)	80.27(10.81)	32.99(1.38)	54.98(1.27)	36.81(0.89)	88.33(0.2)	76.49(0.41)	89.1(1.21)
GCNII-64	71.89	68.84	66.49	28.76	54.14	36.1	**88.49**	77.08	89.57
FB-GCNII-64	76.49(4.6)	76.27(7.43)	76.22(9.73)	29.57(0.81)	54.39(0.25)	36.79(0.69)	87.92(-0.57)	77(-0.08)	89.65(0.08)
GCNII*-8	72.97	78.82	72.7	34.89	62.48	40.72	86.14	75.06	89.7
FB-GCNII*-8	76.76(3.79)	**82.94(4.12)**	78.11(5.41)	**35.87(0.98)**	**65.11(2.63)**	41.19(0.47)	86.94(0.8)	76.32(1.26)	90.2(0.5)
GCNII*-16	76.49	81.57	75.41	34.18	58.86	39.88	87.46	75.8	86.69
FB-GCNII*-16	76.95(0.46)	82.39(0.82)	76.76(1.35)	35.4(1.22)	59.98(1.12)	40.08(0.2)	87.48(0.02)	76.43(0.63)	89.95(3.26)
GCNII*-32	74.32	77.06	77.84	33.78	56.27	37.69	88.35	76.55	89.37
FB-GCNII*-32	74.51(0.19)	80.78(3.72)	**84.86(7.02)**	34.73(0.95)	57.65(1.38)	**41.24(3.55)**	88.16(-0.19)	76.89(0.34)	89.92(0.55)
GCNII*-64	72.43	73.53	75.41	32.72	53.82	36.83	88.01	**77.13**	**90.3**
FB-GCNII*-64	75.84(3.41)	81.57(8.04)	80.54(5.13)	34.89(2.17)	57.52(3.7)	39.81(2.98)	87.44(-0.57)	77.03(-0.1)	89.98(-0.32)
Baseline Average	73.28	74.75	72.23	32.91	56.92	37.58	87.45	76.22	88.8
FB-Baseline Average	75.86(2.58)	80.94(6.19)	78.55(**6.32**)	34.24(1.33)	58.36(1.44)	39.37(1.79)	87.48(0.03)	76.54(0.32)	89.77(0.97)

The results are averaged from 10 independent runs. The (values) represent the difference of performance brought by patching FB.

Table 5: Ablation Results: Accuracy (%)

#Channels	Transformation	Cora		Cornell		Texas	
		Mean	Std	Mean	Std	Mean	Std
1	linear	83.92	1.0	64.86	2.2	70.60	1.9
1	nonlinear	84.69	0.5	70.27	0.8	72.97	0.8
2	linear	85.02	2.1	75.64	2.0	74.15	2.1
2	***nonlinear***	***87.50***	*1.6*	***78.38***	*1.5*	***75.68***	*1.0*

Color indicators are added to differentiate the performance of each test case: the greener the better, the redder the worse.

5.4 Ablation Tests

In this subsection, we perform ablation tests by using FB-GraphSAINT [57] on *Cora*, *Cornell* and *Texas* accordingly, which are the three principally different datasets measured by $\hat{L}_{\text{sym}}$ in Table 3 . The ablation tests would examine the effectiveness of each proposed component. The results are summarized in Table 5. The results show that both the non-linear feature extractor and two-channel filtering architecture are able to help capture richer information under different S-value distributions of input features and output labels.(See more ablation results in Appendix C.3)

Moreover, to emphasize the importance of HP component, we also test the learnable coefficients α_L and α_H for two components on FB-GraphSAINT over the 9 datasets at the validation stage (see Table 7 in Appendix C.1). For most of the tasks, neither the coefficients for LP nor HP are negligible. Among 6 out of 9 tasks, the learned coefficients for the HP components are even greater, this indicates the necessity of the HP components in graph representation learning.

In addition, from the averaged real-time change of the learned coefficients α_L and α_H in the output layer of FB-GraphSAINT on *Cora*, *Cornell* and *Texas* during training (see Figure 3 in Appendix C.2), we can see that α_L and α_H will converge to a pair of values that explains how the smooth and non-smooth features will be mixed. The fact that the ratio α_H/α_L is close to 1 again shows that the importance of the non-smooth part in constructing the output signal. More specifically, the importance (red line) is higher when the demand of non-smooth outputs (diff values) is higher. See more ablation study of GCN on PPI in Appendix C.3.

6 Conclusion

This paper recognizes the role of high-frequency information in graph representation learning. The proposed HP filter completes the spectrum of graph filters and yield significantly better representations on several empirical tasks. The importance of the non-smooth component in graph signals can be revealed by the new defined S-value.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[3] D. Bo, X. Wang, C. Shi, and H. Shen. Beyond low-frequency information in graph convolutional networks. *arXiv preprint arXiv:2101.00797*, 2021.

[4] S. Bougleux, A. Elmoataz, and M. Melkemi. Local and nonlocal discrete regularization on weighted graphs for image and mesh processing. *International journal of computer vision*, 84(2):220–236, 2009.

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *arXiv*, abs/1611.08097, 2016.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv*, abs/1312.6203, 2014.

[7] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010.

[8] D. Cai, X. Wang, and X. He. Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th annual international conference on machine learning*, pages 105–112, 2009.

[9] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

[10] E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations. https://openreview.net/forum*, 2021.

[11] F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[12] M. Daković, L. Stanković, and E. Sejdić. Local smoothness of graph signals. *Mathematical Problems in Engineering*, 2019, 2019.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv*, abs/1606.09375, 2016.

[14] V. N. Ekambaram. *Graph structured data viewed through a fourier lens*. University of California, Berkeley, 2014.

[15] V. N. Ekambaram, G. Fanti, B. Ayazifar, and K. Ramchandran. Critically-sampled perfect-reconstruction spline-wavelet filterbanks for graph signals. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 475–478. IEEE, 2013.

[16] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE transactions on Image Processing*, 17(7):1047–1060, 2008.

[17] L. C. Evans. Partial differential equations. graduate studies in mathematics. *American mathematical society*, 2:1998, 1998.

[18] A. Gavili and X.-P. Zhang. On the shift operator, graph frequency, and optimal filtering in graph signal processing. *IEEE Transactions on Signal Processing*, 65(23):6303–6318, 2017.

[19] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[20] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

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

[22] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *arXiv*, abs/1706.02216, 2017.

[23] M. He, Z. Wei, H. Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34, 2021.

[24] X. He and P. Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.

[25] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2013.

[26] Y. Jiang, D. I. Bolnick, and M. Kirkpatrick. Assortative mating in animals. *The American Naturalist*, 181(6):E125–E138, 2013.

[27] V. Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929, 2016.

[28] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv*, abs/1609.02907, 2016.

[29] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[30] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[31] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *arXiv*, abs/1901.01484, 2019.

[33] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021.

[34] M. Liu, Z. Wang, and S. Ji. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612*, 2020.

[35] S. Luan, C. Hua, Q. Lu, J. Zhu, X.-W. Chang, and D. Precup. When do we need gnn for node classification? *arXiv preprint arXiv:2210.16979*, 2022.

[36] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint arXiv:2109.05641*, 2021.

[37] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup. Revisiting heterophily for graph neural networks. *arXiv preprint arXiv:2210.07606*, 2022.

[38] S. Luan, M. Zhao, X.-W. Chang, and D. Precup. Break the ceiling: Stronger multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1906.02174*, 2019.

[39] T. Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[40] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[41] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[42] M. E. Newman. Mixing patterns in networks. *Physical review E*, 67(2):026126, 2003.

[43] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210, 2007.

[44] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[45] E. M. Rogers. *Diffusion of innovations*. Simon and Schuster, 2010.

[46] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding, 2019.

[47] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[48] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[49] K. Smith, L. Spyrou, and J. Escudero. Graph-variate signal analysis. *IEEE Transactions on Signal Processing*, 67(2):293–305, 2018.

[50] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, and T. Constantinides. Graph signal processing–part i: Graphs, graph spectra, and spectral clustering. *arXiv preprint arXiv:1907.03467*, 2019.

[51] L. Stanković, E. Sejdić, and M. Daković. Reduced interference vertex-frequency distributions. *IEEE Signal Processing Letters*, 25(9):1393–1397, 2018.

[52] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv*, abs/1710.10903, 2017.

[53] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[54] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv*, abs/1901.00596, 2019.

[55] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng. Graph wavelet neural network. *arXiv preprint arXiv:1904.07785*, 2019.

[56] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[57] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

[58] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, 2019.

[59] L. Zhao and L. Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

[60] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. Graph regularized sparse coding for image representation. *IEEE transactions on image processing*, 20(5):1327–1336, 2010.

[61] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[62] J. Zhu, R. A. Rossi, A. Rao, T. Mai, N. Lipka, N. K. Ahmed, and D. Koutra. Graph neural networks with heterophily. *arXiv preprint arXiv:2009.13566*, 2020.

[63] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Generalizing graph neural networks beyond homophily. *arXiv preprint arXiv:2006.11468*, 2020.

[64] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[65] X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1052–1059, 2005.

A Dataset Descriptions

For node classification. there are 4 main categories:

Cora, *Citeseer*, and *Pubmed* are 3 benchmark datasets [47] in the category of *Citation network*. Such networks use nodes to represent papers and edges to denote citations. Node features are the bag-of-words representation and node labels are classified into different academic topics.

Cornell, *Texas*, and *Wisconsin* belong to the webpage dataset *WebKB* [44] created by Carnegie Mellon University. Each node represents a web page, and the edges are hyperlinks between nodes. Node features are the bag-of-words representation and node labels are in five classes.

Chameleon and *Squirrel* are twi page-to-page networks in the *Wikipedia network* [46]. Nodes represent web pages and edges show mutual links between pages. Node features are informative nouns in the Wikipedia pages and nodes are classified into 5 groups based on monthly views.

Actor refers to the *Actor co-occurrence network*. A node correspond to an actor, and an edge exists if two actors occur on the same Wikipedia page. Node features correspond to some keywords in the Wikipedia pages, and nodes are categorized into five classes of words of actors actor's Wikipedia.

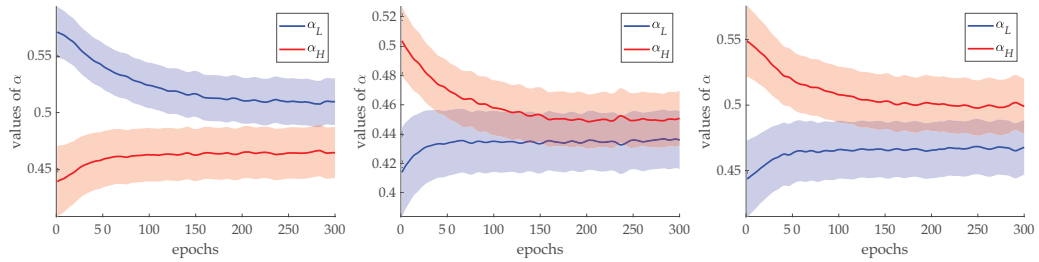
B Graph Classification

For the graph classification tasks, we compare the patched methods FB-GIN-0 and FB-GIN-ϵ against the baselines GIN-0 and GIN-ϵ, with the same experiment setting as [56]. The results (accuracy and standard deviation) are provided in Table 6.

C More Ablation Tests

C.1 Ablation Coefficients

C.2 Real Time Change of Coefficients



(a) Cora (diff = -0.429) (b) Cornell (diff = -0.033) (c) Texas (diff = 0.096)

Figure 3: α_L and α_H in the output layer of FB-GraphSAINT trained on Cora, Cornell and Texas. The mean curves and the std bands are obtained over 20 independent runs. See diff values in table .

C.3 Ablation Tests on PPI

D t-SNE Visualization

Table 6: Results and Hyperparameters of Graph Classification

Task	Method	lr	weight decay	gamma	width	batch size	dropout	concat	Acc
MUTAG	GIN-0								89.4
	FB-GIN-0	0.036104	0.0001034	-0.48239	128	32	0.75127	0	**91.4035**
	GIN-eps								89
	FB-GIN-eps	0.003584	0.011275		64	32	0.75859	0	**94.74**
PROTEINS	GIN-0								76.2
	FB-GIN-0	0.047597	0.00042991	1.269	8	32	0.064654	1	**80.784**
	GIN-eps								75.9
	FB-GIN-eps	0.006173	0.02751		8	128	0.26964	0	**79.4375**
PTC	GIN-0								64.6
	FB-GIN-0	0.0083924	0.0059482	0.72171	16	32	0.082378	0	**68.578**
	GIN-eps								63.7
	FB-GIN-eps	0.049951	0.00029225		16	128	0.30306	0	**71.429**
NCI1	GIN-0								82.7
	FB-GIN-0	0.00039327	0.01014	0.95777	128	128	0.01526	1	**84.428**
	GIN-eps								82.7
	FB-GIN-eps	9.94E-05	0.0083156		128	128	0.70224	1	**84.123**
IMDB-B	GIN-0								75.1
	FB-GIN-0	0.010815	0.00024241	0.83553	128	128	0.97456	1	**83**
	GIN-eps								74.3
	FB-GIN-eps	0.015596	0.0047105		32	32	0.80636	1	**78.111**
IMDB-M	GIN-0								52.3
	FB-GIN-0	0.00067325	0.0042346	1.4691	64	128	0.80828	1	**53.467**
	GIN-eps								52.1
	FB-GIN-eps	0.00061908	0.037266		64	128	0.92727	1	**53.259**
RDT-B	GIN-0								92.4
	FB-GIN-0	0.01262	0.047278	-0.41963	8	128	0.48795	0	**94**
	GIN-eps								92.2
	FB-GIN-eps	0.0068918	0.016003		128	128	0.4131	0	**93**
RDT-M5K	GIN-0								57.5
	FB-GIN-0	0.0011204	0.017434	-0.02748	8	128	0.35465	1	**65.6**
	GIN-eps								57
	FB-GIN-eps	0.0026491	0.0492		8	128	0.55127	1	**68.4**
COLLAB	GIN-0								80.2
	FB-GIN-0	2.88E-04	0.047982	-0.3438	128	128	0.66614	1	**86.3**
	GIN-eps								80.1
	FB-GIN-eps	0.00019472	0.00031991		128	128	0.1088	1	**85**

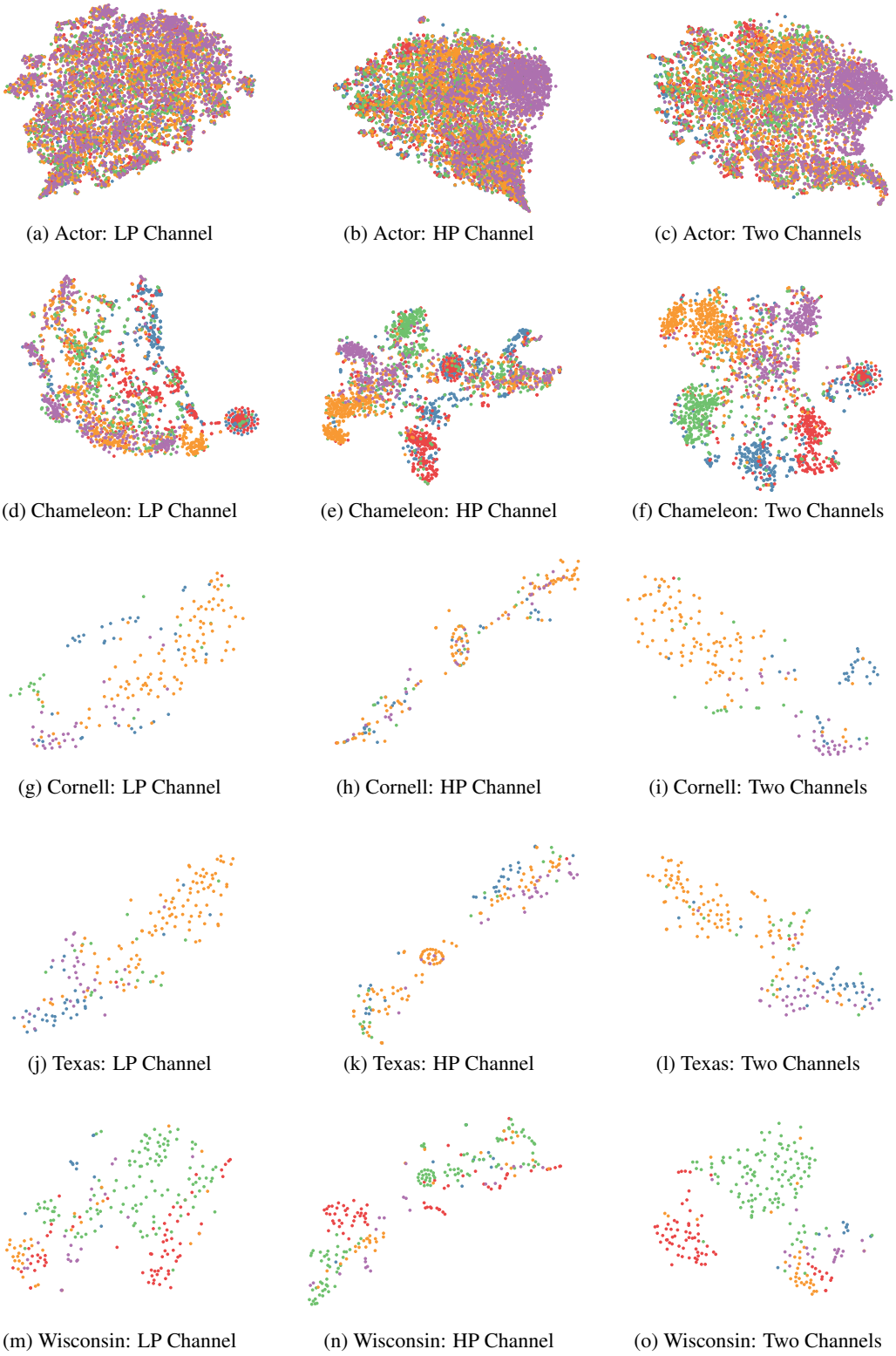
Table 7: α_L and α_H in the Output Layer of FB-GraphSAINT

	Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	Citeseer	Pubmed
α_L	0.436	0.441	0.57	0.54	0.701	0.675	0.509	0.514	0.473
α_H	**0.45**	**0.499**	**0.6**	**0.557**	**0.713**	0.65	0.464	0.503	**0.478**
α_H/α_L	**1.032**	**1.132**	**1.053**	**1.031**	**1.017**	0.963	0.912	0.979	**1.011**

The results are averaged from 10 independent runs. If the ratio is higher than 1.0, then the high frequency signals are more important. The higher the ratio, the more important HP filter is.

Table 8: Ablation Tests on PPI

#Channels	Transformation	F1-score	Std
1	linear	59.4	0.8
1	nonlinear	69.5	0.3
2	linear	71.8	0.6
2	nonlinear	73.9	0.4



(a) Actor: LP Channel (b) Actor: HP Channel (c) Actor: Two Channels

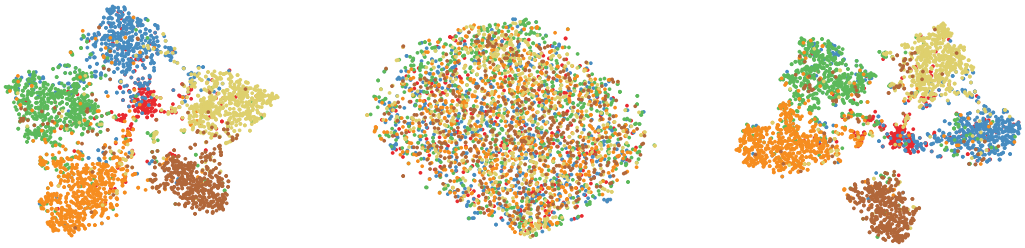
(d) Chameleon: LP Channel (e) Chameleon: HP Channel (f) Chameleon: Two Channels

(g) Cornell: LP Channel (h) Cornell: HP Channel (i) Cornell: Two Channels

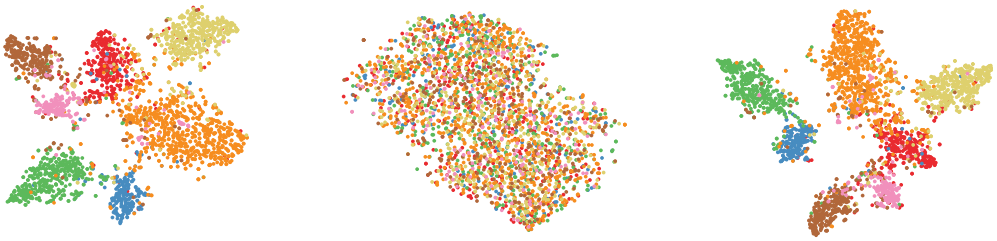
(j) Texas: LP Channel (k) Texas: HP Channel (l) Texas: Two Channels

(m) Wisconsin: LP Channel (n) Wisconsin: HP Channel (o) Wisconsin: Two Channels

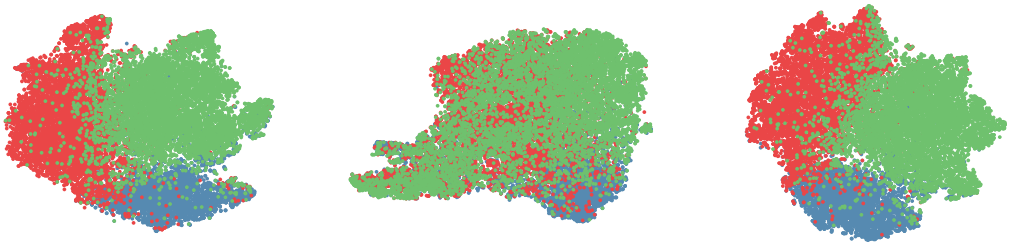
Figure 4: t-SNE Visualization of the Learned Node Embeddings for heterophilic datasets (other than Squirrel, which is presented in the main manuscript) under 3 configurations.



(a) Citeseer: LP Channel (b) Citeseer: HP Channel (c) Citeseer: Two Channels



(d) Cora: LP Channel (e) Cora: HP Channel (f) Cora: Two Channels



(g) Pubmed: LP Channel (h) Pubmed: HP Channel (i) Pubmed: Two Channels

Figure 5: t-SNE Visualization of the Learned Node Embeddings for homophilic datasets under 3 configurations.

E Discussion of Filters

E.1 Lazy Random Walk Matrix

In graph signal processing, lazy random walk defined in the following equation is often used as a LP filter [14],

$$A_{\text{lrw}} = \frac{1}{2}(I + A_{\text{rw}}). \tag{9}$$

It can be seen as adding D_{ii} self-loops to the i-th node of A and normalize it to be a random walk matrix and $0 \leq \lambda_i(A_{\text{lrw}}) \leq 1$. Such spectral property makes it a standard band-pass filter, which can avoid some theoretical confusion due to negative eigenvalues. Unlike $\hat{A}_{\text{rw}}$, A_{lrw} maintains certain topology properties of A_{rw}, *e.g.,* stationary distribution and eigenvectors. In practice, these properties are supposed to be changed, unless one has strong prior knowledge that GNNs will benefit from the renormalized one.

Furthermore, it is found that adding self-loops can shrink the magnitude of the dominant eigenvalue so that the influence of long-distance nodes will be reduced, which makes the filtered signal more dependent on local information [21]. There is growing empirical evidence showing that adding self-loops will lead to effective graph convolutions on some applications [29, 53]. Compared to $\hat{A}_{\text{rw}}$, A_{lrw} works better at reducing the magnitude of dominant eigenvalue under certain conditions. We show it in the following theorem.

Theorem 1. We denote the generalized lazy random walk matrix and the generalized renormalized adjacency matrix respectively by

$$A^{\gamma}_{\text{lrw}} = \frac{1}{1+\gamma}(\gamma I + A_{\text{rw}}), \;\; \hat{A}^{\gamma}_{\text{rw}} = \tilde{D}^{-1}_{\gamma}\tilde{A}_{\gamma},$$

where $\tilde{A}_{\gamma} = \gamma I + A, \tilde{D}_{\gamma} = \gamma I + D$. Suppose $\mathcal{G}$ has no isolated node, *i.e.,* $D_{ii} > 0$ for all i, and for positive γ, we have

$$\frac{\lambda_2(A^{\gamma}_{\text{lrw}})}{\lambda_1(A^{\gamma}_{\text{lrw}})} \geq \frac{\lambda_2(\hat{A}^{\gamma}_{\text{rw}})}{\lambda_1(\hat{A}^{\gamma}_{\text{rw}})}, \tag{10}$$

where $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$ are the largest and second largest eigenvalues a matrix.

Proof. Detailed proof can be found in Appendix E.2. □

The HP filer derived from A_{lrw} is $(I - A_{\text{rw}})/2$ and they can be used as a set of filterbank in FB-GNN framework. From table 9 we can see that, FB-GNNs with lazy random walk matrix can boost the performance of baseline GNNs more significantly than symmetric renormalized affinity matrix on heterophilic datasets *Cornell, Wisconsin, Texas* and *Film*, where baseline GNNs underperform MLP. And on homophilic datasets, where baseline GNNs outperform MLP, FB-GNNs with symmetric renormalized affinity matrix perform better.

Table 9: Comparison of FB-GNNs with lazy random walk and symmetric renormalized affinity matrix

Models\Datasets	Cornell	Wisconsin	Texas	Film	Chameleon	Squirrel	Cora	Citeseer	Pubmed
MLP	85.14	87.25	84.59	36.08	46.21	29.39	74.81	73.45	87.86
GCN	60.81	63.73	61.62	30.98	61.34	41.86	87.32	76.70	88.24
FB-GCM-sym	83.78	87.45	84.59	35.47	**65.66**	**50.56**	87.34	76.42	**89.74**
FB-GCN-lazy	85.14	87.25	86.49	36.11	61.07	46.28	87.46	76.24	89.57
GAT	59.19	60.78	59.73	29.71	61.95	43.88	**88.07**	76.42	87.81
FB-GAT-sym	87.30	85.88	82.70	36.00	63.75	44.07	87.34	76.45	89.03
FB-GAT-lazy	**88.92**	**89.22**	**88.65**	**36.83**	62.80	43.56	87.28	76.92	89.43
GraphSage	82.97	87.84	82.43	35.28	47.32	30.16	85.98	77.07	88.59
FB-GraphSage-sym	86.44	88.43	86.22	35.88	48.11	33.24	86.62	**78.01**	89.05
FB-GraphSage-lazy	86.49	87.45	87.30	35.00	48.57	30.44	86.35	76.48	88.58
diff(FB-sym, baseline)	18.18	16.47	16.58	3.79	2.30	3.99	-0.02	0.23	1.06
diff(FB-lazy, baseline)	19.19	17.19	19.55	3.99	0.61	1.46	-0.09	-0.18	0.98

Table 10: More comparisons of FB-GNN with lazy random walk and symmetric renormalized matrix

Models\Datasets	CitationFull_dblp	Coauthor_CS	Coauthor_Physics	Amazon_Computers	Amazon_Photo
MLP	77.39	93.72	95.77	83.89	90.87
GCN	85.87	93.91	96.84	87.03	93.61
FB-GCM-sym	**85.90**	**95.33**	97.03	**91.54**	**95.57**
FB-GCN-lazy	85.51	95.31	**97.07**	91.32	95.53
GAT	85.89	93.41	96.32	89.74	94.12
FB-GAT-sym	84.94	94.13	OOM	89.57	94.58
FB-GAT-lazy	85.35	95.30	OOM	91.10	94.88
GraphSage	81.19	94.38	OOM	83.70	NA
FB-GraphSage-sym	85.66	94.97	OOM	88.01	91.40
FB-GraphSage-lazy	85.02	95.00	OOM	87.12	91.02
diff(FB-sym, baseline)	1.18	0.91	0.45	2.88	-0.02
diff(FB-lazy, baseline)	0.98	1.30	0.49	3.02	-0.06

E.2 Proof of Eigengap

Proof. Denote the symmetric normalized lazy random walk matrix and the generalized symmetric renormalized adjacency matrix respectively by

$$A^{\gamma}_{\text{slrw}} = \frac{1}{1+\gamma}(\gamma I + A_{\text{sym}}), \;\; \hat{A}^{\gamma}_{\text{sym}} = \tilde{D}_{\gamma}^{-1/2}\tilde{A}_{\gamma}\tilde{D}_{\gamma}^{-1/2}$$

It is easy to verify that

$$\lambda(A^{\gamma}_{\text{slrw}}) = \lambda(A^{\gamma}_{\text{lrw}}), \;\; \lambda(\hat{A}_{\text{sym}}) = \lambda(\hat{A}^{\gamma}_{\text{rw}})$$

where $\lambda(\cdot)$ denotes the spectrum of a matrix. Since $\lambda_1(A^{\gamma}_{\text{lrw}}) = \lambda_1(\hat{A}^{\gamma}_{\text{rw}}) = 1$, to prove the theorem, it is necessary and sufficient to prove

$$\lambda_2(A^{\gamma}_{\text{slrw}}) \geq \lambda_2(\hat{A}^{\gamma}_{\text{sym}}) \tag{11}$$

It is easy to show that $D^{1/2}\mathbf{1}$ is an eigenvector of A^{γ}_{slrw} corresponding to $\lambda_1(A^{\gamma}_{\text{slrw}})$ and $\tilde{D}_{\gamma}^{1/2}\mathbf{1}$ is an eigenvector $\hat{A}^{\gamma}_{\text{sym}}$ corresponding to $\lambda_1(\hat{A}^{\gamma}_{\text{sym}})$.

By the Rayleigh quotient theorem ([25]),

$$\lambda_2(A^{\gamma}_{\text{slrw}}) = \max_{\boldsymbol{x}\perp D^{1/2}\mathbf{1}} \frac{\boldsymbol{x}^T D^{-1/2}(\gamma D + A)D^{-1/2}\boldsymbol{x}}{(1+\gamma)\boldsymbol{x}^T\boldsymbol{x}} \tag{12}$$

By the Rayleigh quotient theorem and the Courant-Fischer min-max theorem ([25]),

$$\begin{aligned}\lambda_2(\hat{A}^{\gamma}_{\text{sym}}) &= \max_{\boldsymbol{x}\perp\tilde{D}_{\gamma}^{1/2}\mathbf{1}} \frac{\boldsymbol{x}^T\tilde{D}_{\gamma}^{-1/2}(\gamma I+A)\tilde{D}_{\gamma}^{-1/2}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \\ &= \min_{\{S:\dim S=n-1\}}\ \max_{\{\boldsymbol{x}:0\neq\boldsymbol{x}\in S\}} \frac{\boldsymbol{x}^T\tilde{D}_{\gamma}^{-1/2}(\gamma I+A)\tilde{D}_{\gamma}^{-1/2}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \\ &\leq \max_{\boldsymbol{x}\perp D^{1/2}\mathbf{1}} \frac{\boldsymbol{x}^T\tilde{D}_{\gamma}^{-1/2}(\gamma I+A)\tilde{D}_{\gamma}^{-1/2}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \end{aligned} \tag{13}$$

Then from (12) and (13) we obtain

$$\begin{aligned}&\lambda_2(A^{\gamma}_{\text{slrw}}) - \lambda_2(\hat{A}^{\gamma}_{\text{sym}}) \\ &\geq \max_{\boldsymbol{x}\perp D^{1/2}\mathbf{1}} \frac{\boldsymbol{x}^T D^{-1/2}(\gamma D+A)D^{-1/2}\boldsymbol{x}}{(1+\gamma)\boldsymbol{x}^T\boldsymbol{x}} \\ &\quad - \max_{\boldsymbol{x}\perp D_{\gamma}^{1/2}\mathbf{1}} \frac{\boldsymbol{x}^T\tilde{D}_{\gamma}^{-1/2}(\gamma I+A)\tilde{D}_{\gamma}^{-1/2}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \\ &= \max_{\boldsymbol{y}\perp D\mathbf{1}} \frac{\boldsymbol{y}^T(\gamma D+A)\boldsymbol{y}}{(1+\gamma)\boldsymbol{y}^T D\boldsymbol{y}} + \min_{\boldsymbol{y}\perp D\mathbf{1}} \left(-\frac{\boldsymbol{y}^T(\gamma I+A)\boldsymbol{y}}{\boldsymbol{y}^T\tilde{D}_{\gamma}\boldsymbol{y}}\right) \\ &\geq \min_{\boldsymbol{y}\perp D\mathbf{1}} \left(\frac{\boldsymbol{y}^T(\gamma D+A)\boldsymbol{y}}{(1+\gamma)\boldsymbol{y}^T D\boldsymbol{y}} - \frac{\boldsymbol{y}^T(\gamma I+A)\boldsymbol{y}}{\boldsymbol{y}^T\tilde{D}_{\gamma}\boldsymbol{y}}\right) \\ &= \min_{\boldsymbol{y}\perp D\mathbf{1}} \left(\frac{(\boldsymbol{y}^T(\gamma D+A)\boldsymbol{y})(\boldsymbol{y}^T(\gamma I+D)\boldsymbol{y})}{((1+\gamma)\boldsymbol{y}^T D\boldsymbol{y})(\boldsymbol{y}^T(\gamma I+D)\boldsymbol{y})}\right. \\ &\qquad \left. - \frac{(\boldsymbol{y}^T(\gamma I+A)\boldsymbol{y})(((1+\gamma)\boldsymbol{y}^T D\boldsymbol{y}))}{((1+\gamma)\boldsymbol{y}^T D\boldsymbol{y})(\boldsymbol{y}^T(\gamma I+D)\boldsymbol{y})}\right) \\ &= \min_{\boldsymbol{y}\perp D\mathbf{1}} \frac{\gamma\left(1+\frac{\boldsymbol{y}^T A\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}\frac{\boldsymbol{y}^T\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}} - \frac{\boldsymbol{y}^T\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}} - \frac{\boldsymbol{y}^T A\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}\right)}{((1+\gamma)\boldsymbol{y}^T D\boldsymbol{y})(\boldsymbol{y}^T(\gamma I+D)\boldsymbol{y})/(\boldsymbol{y}^T D\boldsymbol{y})^2} \\ &= \min_{\boldsymbol{y}\perp D\mathbf{1}} \frac{\gamma\left(1-\frac{\boldsymbol{y}^T A\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}\right)\left(1-\frac{\boldsymbol{y}^T\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}\right)}{((1+\gamma)\boldsymbol{y}^T D\boldsymbol{y})(\boldsymbol{y}^T(\gamma I+D)\boldsymbol{y})/(\boldsymbol{y}^T D\boldsymbol{y})^2} \geq 0\end{aligned}$$

□

F Hyperparameters

In this subsection, we report the optimal hyperparameters that are searched for FB-GNNs.

Table 11: Hyperparameters for baseline models

Datasets	Models\Hyperparameters	lr	weight_decay	dropout	hidden	results
Cornell	GCN	0.05	5.00E-04	0.4	32	60.81
	GAT	0.005	5.00E-04	0.6	8	59.19
	GraphSAGE	0.1	1.00E-04	0.1	32	82.97
	MLP	0.05	1.00E-04	0.5	32	85.14
Wisconsin	GCN	0.05	5.00E-04	0.3	32	63.73
	GAT	0.005	5.00E-04	0.2	8	60.78
	GraphSAGE	0.1	1.00E-04	0.1	32	87.84
	MLP	0.05	1.00E-04	0.4	32	87.25
Texas	GCN	0.05	5.00E-05	0.4	32	61.62
	GAT	0.005	5.00E-04	0.1	8	59.73
	GraphSAGE	0.1	5.00E-04	0.2	32	82.43
	MLP	0.05	5.00E-04	0.3	32	84.59
Film	GCN	0.05	5.00E-04	0.3	32	30.98
	GAT	0.005	1.00E-04	0.2	8	29.71
	GraphSAGE	0.1	5.00E-04	0.3	32	35.28
	MLP	0.05	5.00E-05	0.9	32	36.08
Chameleon	GCN	0.05	5.00E-05	0.3	32	61.34
	GAT	0.005	1.00E-04	0.3	8	61.95
	GraphSAGE	0.1	5.00E-04	0.5	32	47.32
	MLP	0.05	5.00E-05	0.3	32	46.21
Squirrel	GCN	0.05	5.00E-05	0.6	32	41.86
	GAT	0.005	1.00E-04	0.2	8	43.88
	GraphSAGE	0.1	5.00E-05	0.6	32	30.16
	MLP	0.05	5.00E-05	0.4	32	29.39
Cora	GCN	0.05	5.00E-05	0.9	32	87.32
	GAT	0.005	1.00E-04	0.7	8	88.07
	GraphSAGE	0.1	5.00E-05	0.6	32	85.98
	MLP	0.05	5.00E-04	0.4	32	74.81
Citeseer	GCN	0.05	5.00E-04	0.5	32	76.7
	GAT	0.005	5.00E-04	0.6	8	76.42
	GraphSAGE	0.1	1.00E-04	0.6	32	77.07
	MLP	0.05	5.00E-05	0.6	32	73.45
Pubmed	GCN	0.05	5.00E-05	0.2	32	88.24
	GAT	0.005	5.00E-05	0.1	8	87.81
	GraphSAGE	0.1	5.00E-05	0.2	32	88.59
	MLP	0.05	1.00E-04	0.1	32	87.86
CitationFull_dblp	GCN	0.05	5.00E-05	0.8	32	85.87
	GAT	0.005	1.00E-04	0.3	8	85.89
	GraphSAGE	0.05	5.00E-05	0.2	32	81.19
	MLP	0.05	5.00E-04	0.3	32	77.39
Coauthor_CS	GCN	0.05	5.00E-05	0.2	32	93.91
	GAT	0.005	5.00E-05	0.2	8	93.41
	GraphSAGE	0.1	5.00E-05	0.2	32	94.38
	MLP	0.05	5.00E-05	0.2	32	93.72
Coauthor_Physics	GCN	0.05	5.00E-05	0.1(0.3)	32	96.84
	GAT	0.005	5.00E-04	0.5	8	96.32
	GraphSAGE	-	-	-	-	OOM
	MLP	0.05	5.00E-05	0.5(0.6)	32	95.77
Amazon_Computers	GCN	0.05	5.00E-05	0.1	32	87.03
	GAT	0.005	5.00E-05	0.2	8	89.74
	GraphSAGE	0.1	5.00E-05	0.1	32	83.7
	MLP	0.05	5.00E-05	0.2	32	83.89
Amazon_Photo	GCN	0.05	5.00E-05	0.2	32	93.61
	GAT	0.005	5.00E-05	0.2	8	94.12
	GraphSAGE					
	MLP	0.05	5.00E-05	0.4	32	90.87

Table 12: Hyperparameters for FB-GNNs

Datasets	Models\Hyperparameters	symmetric renormalized adjacency matrix					lazy random walk matrix		
		lr	weight_decay	dropout	hidden	results	weight_decay	dropout	results
Cornell	MF-GCN	0.05	1.00E-03	0.3	32	83.78	5.00E-04	0.3	85.14
	MF-GAT	0.05	5.00E-04	0.2	8	87.3	5.00E-04	0.3	**88.92**
	MF-GraphSAGE	0.05	5.00E-04	0.1	32	86.44	5.00E-04	0.1	86.49
	MF-Geom-GCN*	0.05	5.00E-04	0.3	32	82.99	5.00E-04	0.3	83.41
Wisconsin	MF-GCN	0.05	5.00E-04	0.1	32	87.45	5.00E-04	0.4	87.25
	MF-GAT	0.05	5.00E-04	0.3	8	85.88	5.00E-04	0.2	**89.22**
	MF-GraphSAGE	0.05	5.00E-04	0.2	32	88.43	5.00E-04	0.3	87.45
	MF-Geom-GCN*	0.05	5.00E-04	0.3	32	85.66	5.00E-04	0.3	86.1
Texas	MF-GCN	0.05	5.00E-04	0.1	32	84.59	1.00E-03	0.3	86.49
	MF-GAT	0.05	1.00E-04	0.6	8	82.7	5.00E-04	0.4	**88.65**
	MF-GraphSAGE	0.1	5.00E-04	0.2	32	86.22	5.00E-04	0.1	87.3
	MF-Geom-GCN*	0.05	5.00E-04	0.3	32	83.41	5.00E-04	0.3	84.41
Film	MF-GCN	0.05	5.00E-03	0.2	32	35.47	5.00E-03	0.2	36.11
	MF-GAT	0.05	5.00E-04	0.5	8	36	5.00E-04	0.5	**36.83**
	MF-GraphSAGE	0.05	5.00E-04	0.1	32	35.88	5.00E-05	0.4	35
	MF-Geom-GCN*	0.05	5.00E-05	0.6	32	34.26	5.00E-05	0.7	34.08
Chameleon	MF-GCN	0.05	5.00E-05	0.7	32	**65.66**	5.00E-05	0.7	61.07
	MF-GAT	0.005	5.00E-04	0.5	8	63.75	5.00E-04	0.4	62.8
	MF-GraphSAGE	0.05	5.00E-04	0.6	32	48.11	5.00E-04	0.6	48.57
	MF-Geom-GCN*	0.05	5.00E-05	0.8	32	63.8	5.00E-05	0.8	62.23
Squirrel	MF-GCN	0.05	5.00E-05	0.6	32	**50.56**	5.00E-05	0.6	46.28
	MF-GAT	0.005	5.00E-05	0.5	8	44.07	5.00E-04	0.5	43.56
	MF-GraphSAGE	0.05	5.00E-04	0.5	32	33.24	5.00E-04	0.6	30.44
	MF-Geom-GCN*	0.05	5.00E-05	0.7	32	40.02	5.00E-05	0.8	39.02
Cora	MF-GCN	0.05	5.00E-04	0.8	32	87.34	5.00E-04	0.7	87.46
	MF-GAT	0.05	5.00E-05	0.6	8	87.34	1.00E-04	0.6	87.28
	MF-GraphSAGE	0.05	5.00E-05	0.7	32	86.62	1.00E-04	0.6	86.35
	MF-Geom-GCN*	0.05	1.00E-04	0.6	32	**87.81**	1.00E-04	0.7	87.3
Citeseer	MF-GCN	0.05	5.00E-03	0.3	32	76.42	5.00E-03	0.3	76.24
	MF-GAT	0.05	1.00E-04	0.6	8	76.45	5.00E-04	0.6	76.92
	MF-GraphSAGE	0.05	5.00E-05	0.7	32	78.01	5.00E-05	0.7	76.48
	MF-Geom-GCN*	0.05	5.00E-04	0.6	32	**78.02**	5.00E-04	0.7	77.02
Pubmed	MF-GCN	0.05	5.00E-04	0.3	32	89.74	5.00E-04	0.2	89.57
	MF-GAT	0.05	5.00E-05	0.3	8	89.03	5.00E-05	0.4	89.43
	MF-GraphSAGE	0.05	5.00E-05	0.3	32	89.05	5.00E-05	0.3	88.58
CitationFull_dblp	MF-GCN	0.05	5.00E-05	0.6	32	**85.9**	0.00E+00	0.6	85.51
	MF-GAT	0.05	5.00E-05	0.6	8	84.94	5.00E-05	0.5	85.35
	MF-GraphSAGE	0.05	5.00E-05	0.3	32	85.66	5.00E-05	0.6	85.02
Coauthor_CS	MF-GCN	0.05	1.00E-04	0.3	32	**95.33**	1.00E-04	0.4	95.31
	MF-GAT	0.05	5.00E-05	0.4	8	94.13	5.00E-05	0.5	95.3
	MF-GraphSAGE	0.05	5.00E-05	0.3	32	94.97	5.00E-05	0.5	95
Coauthor_Physics	MF-GCN	0.05	5.00E-05	0.4	32	97.03	5.00E-05	0.4	**97.07**
	MF-GAT	-	-	-	-	OOM	-	-	OOM
	MF-GraphSAGE	-	-	-	-	OOM	-	-	OOM
Amazon_Computers	MF-GCN	0.05	1.00E-05	0.4	32	**91.54**	5.00E-05	0.4	91.32
	MF-GAT	0.05	5.00E-05	0..2	8	89.57	5.00E-05	0.3	91.1
	MF-GraphSAGE	0.05	5.00E-05	0.6	32	88.01	5.00E-05	0.5	87.12
Amazon_Photo	MF-GCN	0.05	5.00E-05	0.4	32	**95.57**	1.00E-04	0.3	95.53
	MF-GAT	0.05	1.00E-04	0.2	8	94.58	1.00E-04	0.4	94.88
	MF-GraphSAGE	0.05	5.00E-05	0.5	32	91.4	5.00E-05	0.6	91.02