Graph Neural Networks with Diverse Spectral Filtering

Jingwei Guo*
University of Liverpool
Liverpool, UK
Jingwei.Guo@Liverpool.ac.uk

Kaizhu Huang†
Duke Kunshan University
Suzhou, China
Kaizhu.Huang@dukekunshan.edu.cn

Xinping Yi
University of Liverpool
Liverpool, UK
Xinping.Yi@Liverpool.ac.uk

Rui Zhang
Xi'an Jiaotong-Liverpool University
Suzhou, China
Rui.Zhang02@xjtlu.edu.cn

ABSTRACT

Spectral Graph Neural Networks (GNNs) have achieved tremendous success in graph machine learning, with polynomial filters applied for graph convolutions, where all nodes share the *identical* filter weights to mine their local contexts. Despite the success, existing spectral GNNs usually fail to deal with complex networks (e.g., WWW) due to such *homogeneous* spectral filtering setting that ignores the regional *heterogeneity* as typically seen in real-world networks. To tackle this issue, we propose a novel *diverse* spectral filtering (DSF) framework, which automatically learns node-specific filter weights to exploit the varying local structure properly. Particularly, the diverse filter weights consist of two components — A global one shared among all nodes, and a local one that varies along network edges to reflect node difference arising from distinct graph parts — to balance between local and global information. As such, not only can the global graph characteristics be captured, but also the diverse local patterns can be mined with awareness of different node positions. Interestingly, we formulate a novel optimization problem to assist in learning diverse filters, which also enables us to enhance any spectral GNNs with our DSF framework. We showcase the proposed framework on three state-of-the-arts including GPR-GNN, BernNet, and JacobiConv. Extensive experiments over 10 benchmark datasets demonstrate that our framework can consistently boost model performance by up to 4.92% in node classification tasks, producing diverse filters with enhanced interpretability. Code is available at https://github.com/jingweio/DSF.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Graph Neural Networks, Spectral Filtering, Diverse Mixing Patterns

*Also with Xi'an Jiaotong-Liverpool University.
†Corresponding Author

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583324

ACM Reference Format:
Jingwei Guo, Kaizhu Huang, Xinping Yi, and Rui Zhang. 2023. Graph Neural Networks with Diverse Spectral Filtering. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583324

1 INTRODUCTION

Recent years have witnessed the explosive growth of learning from graph-structured data. As an emerging technique, Graph Neural Networks (GNNs) have recently attracted significant attention in handling data with complex relationships between entities. Capable of exploiting node features and graph topology simultaneously and adaptively, GNNs achieve state-of-the-art performance in a wide variety of graph analytical tasks, such as social analysis and recommendation system [8, 45]. Spectral GNNs are a class of GNN models that implement convolution operations in the spectral domain. Recent studies show that modern variants mostly function as polynomial spectral filters [5, 9, 11, 20, 24, 42]. Specifically, these filters transform the input source (node features) into a new desired space by selectively attenuating or amplifying its Fourier coefficients induced by the graph Laplacian. Existing efforts either design or learn the polynomial coefficients to simulate different types of filters including low, band, and/or high-pass. Despite their success, high degree polynomials are necessary as typically required by their expressive power [9, 20, 28] so as to reach high-order neighborhoods. Nevertheless, most spectral GNNs would fail practically due to the overfitting and/or over-squashing problem [2, 9, 42]. They usually end up enforcing *identical* filter weights among nodes, albeit lying in different network areas, to mine their distinct local contexts (see details in Section 4.1). Namely, the existing spectral GNN models (including GPR-GNN [9], BernNet [20], and JacobiConv [42]) are mostly restricted in the *homogeneous* spectral filtering framework, and focus on the uniform filter weights learning. Such limitation is induced by simplifying diverse regional graph patterns as homogeneous ones at different localities.

However, real-world networks typically exhibit *heterogeneous* mixing pattern [38], i.e., different graph parts may possess diverse characteristics (e.g. local assortative level could vary across the graph as illustrated in Figure 2). Apparently, GNNs with classic *homogeneous* spectral filtering are inadequate to model the varying regional pattern; this could result in poor interpretability on the micro graph mining which is important in accomplishing node-level tasks. A single shared weight set tends to pull the model in many opposite directions, which may lead to a biased model that

merely captures the most common graph patterns while leaving others not well covered. Ideally, in order to properly mine different local contexts, distinct filter weights might be needed for different nodes. To do so, one may parameterize each node a separate set of trainable filter weights. Unfortunately, this would substantially increase model complexity and cause severe overfitting to local noises, especially in case of large-scale graphs with complex linking patterns (see Section 5.3).

This work focuses on adapting spectral GNN models to graphs with diverse mixing patterns. We argue that, instead of parameterizing each node arbitrarily different filter weights or a uniform weight, a reasonable design should be *built upon a shared global model whilst locally adapted to each node with awareness of its location in the graph*. Such a proposition is well evidenced by the key observation that nearby nodes tend to display similar local contexts because of their overlapped neighborhoods. For nodes far apart, they are likely to have more possibilities, e.g., even if residing at disjoint graph regions, these nodes may also possess akin local structures due to their similar positions such as graph borders (see Figure 1d and Figure 5). As such, we aim to take advantage of such rationale as a guide to make node-wise adjustments. To this end, we formulate a novel optimization problem to first encode the positional information of nodes. It embeds graph vertices into a low-dimensional coordinate space, based on which we learn node-specific coefficients properly to alter the original filter weights. Accordingly, the global graph filter can be localized on the micro level, allowing individual nodes to adaptively exploit their diverse local contexts. Meanwhile, some beneficial invariant graph properties can still be preserved by virtue of the mutual filter weights. The proposed framework, named *diverse* spectral filtering (DSF), flexibly handles the complex graph and makes a proper balance between its conformal and disparate regional patterns with enhanced interpretability, as shown in Figure 1. Besides, our DSF is easy to implement and can be readily plug-and-play in any spectral GNNs with considerable performance gains.

To summarize, the main contributions of this work are three-fold:

- We show that many existing spectral GNNs are restricted in the form of *homogeneous* spectral filtering, and identify the need to break this ceiling to deal with complex graphs with regional *heterogeneity*.
- We propose a novel *diverse* spectral filtering (DSF) framework to learn diverse and interpretable spectral filters on the micro level, which consistently leads to performance gains for many spectral GNNs.
- We showcase our DSF framework on three state-of-the-arts including GPR-GNN [9], BernNet [20], and JacobiConv [42]. Extensive evaluations on 10 real-world datasets demonstrate the superiority of DSF framework in node classification tasks.

2 RELATED WORK

2.1 Spectral Graph Neural Networks

Recent studies have shown that most spectral GNNs operate as polynomial spectral filters [9, 20, 42] with either fixed designs such as GCN [24], APPNP [15], and GNN-LF/HF [51] or learnable forms, e.g., ChebNet [11], AdaGNN [12], GPR-GNN [9], ARMA [5], BernNet [20], and JacobiConv [42]. Further remarks can be found in Appendix A. For both types, it is identified that most spectral GNNs apply a *homogeneous* setting for spectral filtering. These present methods tend to focus on the most frequent graph layouts while under-exploring the rich and diverse local patterns. To alleviate this issue, we introduce a *diverse* spectral filtering (DSF) framework to enhance spectral GNNs with trainable diverse filters. It is worth noting that though both JacobiConv [42] and AdaGNN [12] learn multiple filters in a seemingly similar way, they are essentially different from our method in nature. Concretely, their adaptive filters are mainly for studying each feature channel independently, whilist our diverse filters aim at individual context modeling for each node.

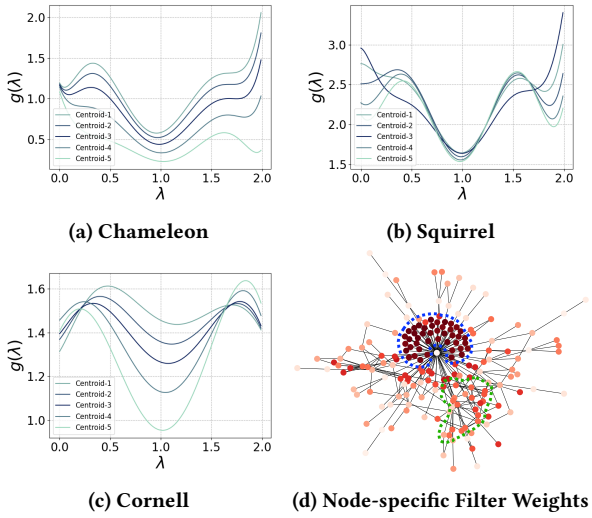


Figure 1: (a)-(c) Diverse filters learned from real-world networks, where five representative curves are plotted for illustration. On each graph, these filters display similar overall shapes but different local details in function curves, showing the capability of our DSF in capturing both the global graph structure and locally varied linking patterns. (d) Visualization of node-specific filter weights on Cornell dataset, where alike color indicates similar filter weights between nodes. Overall, nodes can be differentiated based on their disjoint underlying regions as circled by the blue and green dashed lines, and far-reaching nodes can still learn similar filter weights due to their akin local structures. E.g., vertices on the graph border are mostly ingrained in a line subgraph such as • – • – •, and some unusual cases can be handled (see details in Section 5.4). These results justify the enhanced model interpretability by learning diverse spectral filters on the micro level.

2.2 Graphs with Complex Linking Patterns

Early wisdom in the community was mainly dedicated to learning from graphs with strong homophiliy (assortativity) where most connected nodes share similar attributes and same label [15, 24, 40]. Until 2020, Pei et al. [33] and Zhu et al. [50] first emphasized the importance of studying GNNs in the heterophily (disassortativity) setting, thus categorizing real-world networks into homophilic and heterophilic graphs. Recently massive works [6, 9, 18, 47, 48] have been done which focus more on complex graph scenarios. For example, FAGCN [6] captured both similarity and dissimilarity between pairwise nodes. As much attention from the community

was paid in analyzing graph patterns on the macro/global-level, some researchers [29, 38] start looking into the micro/local graph structures surrounding nodes. In particular, Suresh et al. [38] introduced a node-level assortativity to show heterogeneous mixing patterns inherent in real-world graphs. However, these existing works mostly tackle this regional heterogeneity phenomenon under the intuitive message passing framework [16]. Surprisingly, none of them attempts to solve it from the spectral perspective, a theoretically more elegant framework. To this end, we explore in this paper how to learn spectral GNNs on graphs with diverse mixing patterns and propose a novel framework called *diverse* spectral filtering.

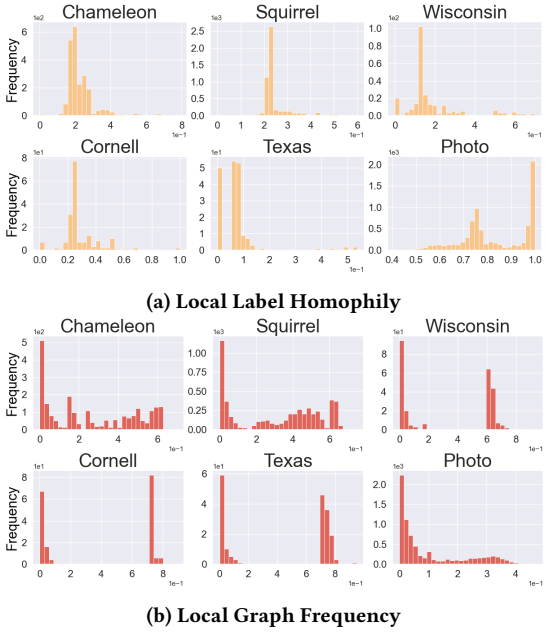
It is noted that a recent proposal [49] shares some similarity with our work. This method takes the idea of dynamic neural networks [19], and introduce PA-GNN based on GPR-GNN [9] by learning node-specific weight offsets. Though PA-GNN is also originated in a spectral-based framework, it leverages a simple but less-justified node-specific aggregation scheme and encodes different (but probably inconsistent) sources of information for prediction. On one hand, this would fail to learn interpretable filters; on the other hand, such drawback limits the accuracy gain and may even hurt the performance, which can be later seen in the experiment part.

2.3 Positional Encoding

Positional encoding (PE) aims to quantify the global position of, e.g., pixels in images, words in texts, and nodes in graphs. It plays a crucial role in facilitating various neural networks such as CNNs [21], RNNs [17], and Transformer [13]. For GNNs, PE has been widely used to increase their model expression [7, 14, 41] as bounded by Weisfeiler-Leman (WL) graph isomorphism test [31, 43, 46]. However, existing focus is mainly put on message-passing GNNs [16], while few research has been conducted on the models defining graph filter in the spectral domain. Recently, Wang and Zhang [42] has established a connection between WL test and the expressive power of spectral GNNs, which motivates us to investigate further how to leverage PE for spectral GNN models. We note that PA-GNN [49] also extract latent positional embeddings from the graph structure, which however cannot be better changed/adjusted to tasks and is mixed with other (even incompatible) attributes. Compared to this, our DSF framework model the positional information of nodes in an independent channel via an iterative updating, thereby producing more expressive and task-beneficial representations. The advantages of this decoupled learning paradigm on node positional and structural features are also verified by other recent works [14, 41].

3 NOTATIONS AND PRELIMINARIES

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an undirected graph with node set $\mathcal{V} = \{v_n\}_{n=1}^N$ with $N = |\mathcal{V}|$ and edge set $\mathcal{E}$ where $(v_i, v_j) \in \mathcal{E}$ if two distinct nodes v_i and v_j are linked by an edge. We use $\mathcal{N}_{i,k} = \{v_j | dis(v_i, v_j) \leq k, \forall v_j \in \mathcal{V}\}$ to represent the k-hop neighborhood of node v_i. The graph topology is symbolized with an $N \times N$ adjacent matrix $\mathbf{A}$ such that $\mathbf{A}_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise. The degree matrix $\mathbf{D}$ is a diagonal matrix with node degrees, e.g., $\deg_i$ w.r.t. node v_i, in its diagonal elements. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ with normalized version $\hat{\mathbf{L}} = \mathbf{I} - \hat{\mathbf{A}}$ where



(a) Local Label Homophily

(b) Local Graph Frequency

Figure 2: Distributions of two graph properties on various real graph data (see details in Section 4.1).

$\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and $\mathbf{I}$ is an $N \times N$ identity matrix. Node features are denoted by $\mathbf{X} \in \mathbb{R}^{N \times f}$ with f being the raw feature dimension. We use $\mathbf{x}_i$ to represent the i^{th} row of $\mathbf{X}$ with respect to node v_i. In node classification tasks, each node v_i is assigned with a class c_i. Thus it has a ground truth one-hot vector $\mathbf{y}_i \in \mathbb{R}^C$, where $C \leq N$ denotes class number. Real-world networks are then divided into homophilic and heterophilic graphs with edge homophily ratio [50] on node labels, $\mathcal{H} = \frac{|\{(v_i,v_j)|\mathbf{y}_i=\mathbf{y}_j \wedge (v_i,v_j)\in\mathcal{E}\}|}{|\mathcal{E}|}$ ranging from 0 to 1 with higher values suggesting higher homophily (lower heterophily).

3.1 Laplacian Decomposition

Let $\hat{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ denote the eigen decomposion of $\hat{\mathbf{L}}$, where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N] \in \mathbb{R}^{N \times N}$ is a matrix of eigenvectors, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_N)$ is the diagonal matrix of eigenvalues. With $\hat{\mathbf{L}}$ being positive semidefinite, we have $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$, of which $\{\mathbf{u}_n\}_{n=1}^N$ is also called the Laplacian eigenbases. We then have

$$\lambda_n = \mathbf{u}_n^T \hat{\mathbf{L}} \mathbf{u}_n = \sum_{(v_p, v_q) \in \mathcal{E}} \left(\frac{1}{\sqrt{\deg_p}} \mathbf{u}_{n,p} - \frac{1}{\sqrt{\deg_q}} \mathbf{u}_{n,q} \right)^2 \quad (1)$$

which measures the frequency or smoothness level of each eigenbasis $\mathbf{u}_n$ on the graph. In this paper, we refer to λ_n as the global graph frequency. Based on all above, the graph Fourier transform and inverse Fourier transform can be respectively formulated as $\mathbf{S} = \mathcal{F}(\mathbf{X}) = \mathbf{U}^T\mathbf{X}$ and $\mathbf{X} = \mathcal{F}^{-1}(\mathbf{S}) = \mathbf{U}\mathbf{S}$, where $\mathbf{S}$ is often called as the Fourier transformed features or Fourier coefficients of $\mathbf{X}$.

3.2 Graph Spectral Filtering

The central idea of spectral GNNs is to transform the graph signal (an instance of node features) in the Fourier space by applying graph spectral filters. They usually take the form as $\mathbf{Z} = g(\hat{\mathbf{L}})\mathbf{X} =$

$\mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X}$ where $g : [0, 2] \to \mathbb{R}$ is a filter function defined on the spectrum of graph Laplacian. This function creates frequency response to filter different components of $\mathbf{X}$ on Laplacian eigenbases. Take one-channeled $\mathbf{X}$ as an example. As $\mathbf{S} = \mathbf{U}^T\mathbf{X} = [s_1, s_2, ..., s_N]^T$, we have $\mathbf{X} = \sum_{n=1}^{N} s_n \cdot \mathbf{u}_n$ and $\mathbf{Z} = \sum_{n=1}^{N} \Big(g(\lambda_n)s_n\Big)\cdot\mathbf{u}_n$ with scalar s_n. It can be seen that node feature matrix $\mathbf{X}$ is mapped into a new $\mathbf{Z}$, by either decreasing or increasing Fourier coefficients $\mathbf{S}$ via $s_n \mapsto g(\lambda_n)s_n$ selectively. Recent studies have shown that most spectral GNN models implement the filter function $g(\cdot)$ with polynomials [5, 20, 42] following $\mathbf{Z} = \sum_{k=0}^{K} \omega_k \hat{\mathbf{L}}^k\mathbf{X} = \sum_{k=0}^{K} \alpha_k P_k(\hat{\mathbf{L}})\mathbf{X}$, where both ω_k and α_k denote the polynomial coefficient (also called filter weight), and $P_k : [0, 2] \to \mathbb{R}$ is a polynomial basis in the k^{th} order. Taking one-channel $\mathbf{X}$ as an example, the existing spectral GNNs can then be unified as

$$\mathbf{Z} = \sum_{k=0}^{K} \alpha_k P_k(\hat{\mathbf{L}})\mathbf{X} = \sum_{n=1}^{N} \hat{s}_n \cdot \mathbf{u}_n \tag{2}$$

where $\hat{s}_n = \sum_{k=0}^{K} \alpha_k P_k(\lambda_n)s_n$ for brief symbolization. Present efforts either fix or learn filter weights with different classes of polynomial basis. For instance, APPNP [15] leverages personalized PageRank [32] to make polynomial basis $P_k(\lambda) = (1 - \lambda)^k$ and set $\alpha_k = \frac{\alpha^k}{1-\alpha}$ with constant hyper-parameter α. As an extension of APPNP, GPR-GNN [9] directly trains α_k with gradient descent. A comprehensive summarization of various spectral GNNs as polynomial spectral filters can be found in [42].

4 DIVERSE SPECTRAL FILTERING

In this section, the motivation of *diverse* spectral filtering is first provided with both theoretical and empirical analysis. We then present our novel diverse filtering framework.

4.1 Motivations

The unified formula in Eq. 2 can be considered as the *homogeneous* spectral filtering, where all nodes share the identical coefficient $\hat{s}_n$ equally operated on their basis signals, i.e., all elements in $\mathbf{u}_n$, for feature transformation. It seems reasonable as one can learn arbitrary $\hat{s}_n$ with a polynomial graph filter [37], which formally requires high-degree polynomials and reaching high-order node neighborhood [9, 20, 28]. However, aggregating/passing information across a long path via $\hat{\mathbf{L}}^k\mathbf{X}$ with $k \to \infty$ is prone to cause overfitting to noises and/or over-squashing problem [2]. Chien et al. [9] practically show that the polynomial coefficients in Eq. 2 converges to zero as k gets larger. With this empirical finding, Wang and Zhang [42] even propose strategies to optimize $\{\alpha_k\}_{k=0}^{K}$ with decreasing scale. All the above reveal the local modeling nature of the existing spectral GNNs. In other words, nodes, albeit lying in different graph parts, are enforced to mine their distinct local contexts with the identical filter weights $\{\alpha_k\}_{k=0}^{K}$. Such filtering scheme implicitly assumes the similar distributions between different graph regions.

This hypothesis however may not be accurate due to the intrinsic complexity in forming real-world networks. To make further investigation, we define two essential graph properties on the local graph level to empirically observe their changing behaviors across the graph. The definitions are given below.

Definition 1 (Local Label Homophily)**.** We define the Local Label Homophily as a measure of the local homophily level surrounding each node v_i:

$$h_i = \frac{|\{(v_p, v_q)|\mathbf{y}_p = \mathbf{y}_q \wedge (v_p, v_q) \in \mathcal{E}_{i,k}\}|}{|\mathcal{E}_{i,k}|}$$

Here, h_i directly computes the edge homophily ratio [50] on the subgraph made up of the k-hop neighbors, and $\mathcal{E}_{i,k} = \{(v_p, v_q)|v_p, v_q \in \mathcal{N}_{i,k} \wedge (v_p, v_q) \in \mathcal{E}\}$ denotes its edge set.

Definition 2 (Local Graph Frequency)**.** The Local Graph Frequency is defined by measuring the local smoothness level of the decomposed Laplacian eigenbases, and for each node v_i we have:

$$\lambda_{n,i} = \sum_{(v_p,v_q)\in\mathcal{E}_{i,k}} \Big(\frac{1}{\sqrt{\deg_p}}\mathbf{u}_{n,p} - \frac{1}{\sqrt{\deg_q}}\mathbf{u}_{n,q}\Big)^2$$

where $\lambda_{n,i}$ denotes the frequency or smoothness level of each Laplacian eigenbasis $\mathbf{u}_n$ upon the subgraph induced by the k-hop neighbors. Since all summed elements in Eq. 1 are positive and $\mathcal{E}_{i,k} \subseteq \mathcal{E}$, we can always have a $\xi_i \in (0, 1)$ such that $\lambda_{n,i} = \xi_i\lambda_n$.

Figure 2 shows the distributions of these two graph properties on various real-world data from different domains (see details in Section 5.1.1). We take two-hop neighborhood to illustrate the local graph pattern and remove a few extreme statistical values for better visualization. Moreover, as we usually have a large number of the decomposed eigenbases $\{\mathbf{u}_n\}_{n=1}^{N}$, it is not quite feasible to visualize them all. Thus, we rank them based on their global graph frequency λ_n, and calculate the local graph frequencies with the middle one as the most representative demonstration. Similar statistics with low- and high-frequency can be found in the Appendix (see Figure 6).

Overall, we observe skewed and even multi-modal distributions. These phenomena imply that the local structural patterns are not uniformly distributed between different graph regions, but exhibiting evident heterogeneity. More importantly, in the spectral domain, the global graph frequency λ_n usually fails to capture the diverse local characteristics of $\mathbf{u}_n$ as shown in Figure 2b. Thus, weighing $\mathbf{u}_n$ with the only one scalar coefficient, $\hat{s}_n$ computed as a function of λ_n, may not be appropriate and tends to cause ineffective modeling. In other words, a diverse filtering framework appears necessary so that one could fully exploit the heterogeneous mixing patterns for adaptive micro graph learning.

4.2 Diverse Filtering Framework

To implement diverse filtering, we aim to improve the classic *homogeneous* spectral filtering by endowing each node a different set of transforming coefficients on the basis signals. Particularly, the scalar $\hat{s}_n$ is expanded as a vector $\hat{\mathbf{s}}_n = [\hat{\mathbf{s}}_{n,1}, \hat{\mathbf{s}}_{n,2}, ..., \hat{\mathbf{s}}_{n,N}]^T$ with the same dimensions as the eigenbasis $\mathbf{u}_n$. Eq. 2 can be thereby enhanced into $\mathbf{Z} = \sum_{n=1}^{N} \hat{\mathbf{s}}_n \odot \mathbf{u}_n$ where $\odot$ denotes element-wise multiplication, and each element in $\hat{\mathbf{s}}_n$ independently operates on the corresponding signal in $\mathbf{u}_n$. Since $\hat{s}_n$ is originally expressed as a polynomial function of λ_n, it is reasonable to make

$$\hat{\mathbf{s}}_{n,i} = f(\lambda_{n,i}) = \sum_{k=0}^{K} \alpha_k P_k(\lambda_{n,i})s_n \tag{3}$$

based on our analysis in the previous section, where $\lambda_{n,i}$ denotes the local graph frequency specified in Definition 2. However, it would

be computationally expensive to calculate $\lambda_{n,i}$, which requires not only Laplacian decomposition but also subgraph extraction. To mitigate this issue, we turn to exploiting the substitution using $\lambda_{n,i} = \xi_i \lambda_n$ s.t. $0 < \xi_i < 1$ with the following proposition.

Proposition 1. Suppose a K-order polynomial function $f : [0, 2] \rightarrow \mathbb{R}$ with polynomial basis $P_k(\cdot)$ and coefficients $\{\alpha_k\}_{k=0}^K$ in real number. For any pair of variables $x, \hat{x} \in [0, 2]$ satisfying $x = \xi\hat{x}$ where ξ is a constant real number, we always have a function $g : [0, 2] \rightarrow \mathbb{R}$ with the same polynomial basis but a different set of coefficients $\{\beta_k\}_{k=0}^K$ such that $f(x) = g(\hat{x})$.

Proposition 1 suggests that the polynomial $f(\lambda_{n,i})$ computing $\hat{\mathbf{s}}_{n,i}$ in Eq. 3 can be reformulated into another function of variable λ_n, using the same basis $P_k(\cdot)$ but a different coefficient set $\{\beta_k\}_{k=0}^K$, i.e., $\hat{\mathbf{s}}_{n,i} = f(\lambda_{n,i}) = \sum_{k=0}^K \alpha_k P_k(\xi_i\lambda_n)s_n = \sum_{k=0}^K \beta_{k,i} P_k(\lambda_n)s_n$. Therefore, our *diverse* spectral filtering can be formulated as:

$$\mathbf{Z} = \sum_{n=1}^{N} \hat{\mathbf{s}}_n \odot \mathbf{u}_n = \sum_{k=0}^{K} \mathrm{diag}(\beta_{k,1}, \beta_{k,2}, ..., \beta_{k,N}) P_k(\hat{\mathbf{L}})\mathbf{X}$$

where each node v_i is parameterized with a different set of filter weights $\{\beta_{k,i}\}_{k=0}^K$. The remaining issue is then how to learn these weights. Existing state-of-the-art spectral GNNs [9, 20, 42] usually set filter weight as free parameters to be directly trained. However, in our framework, doing this would not only lead to a high computational complexity, but also could cause severe overfitting to local noises. In the following, we introduce two strategies so as to deal with the issue.

4.2.1 ***Position-aware Filter Weights.*** It has been shown that $\{\beta_{k,i}\}_{k=0}^K$ is utilized to mine the local context of each node v_i. The differences among these filter weight sets are meant to capture regional heterogeneity on the graph. From another angle, if the filter weights are learned to be similar between nodes, they are more likely to lie in the same region sharing almost identical local structural patterns. While, distant node pairs may have more possibilities, e.g., even if residing at disjoint graph regions, these vertices could still possess alike local subgraphs due to their similar positions in the network such as graph borders. This motivates us to make use of node positional information as a guide to learn diverse filter weights.

To do so, the first step is to encode the node positions into a latent space while preserving their graph-based distance. To attain this, inspired by graph signal denoising [51] and Laplacian loss [4, 25], we formulate an novel optimization problem with the objective $\mathcal{L}_p$:

$$\arg\min_{\mathbf{P}} \ \mathcal{L}_p = \|\mathbf{X}_p - \mathbf{P}\|_F^2 + \kappa_1 tr(\mathbf{P}^T\hat{\mathbf{L}}\mathbf{P}) + \kappa_2\|\mathbf{P}^T\mathbf{P} - \mathbf{I}_d\|_F^2 \quad (4)$$

where $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N]^T \in \mathbb{R}^{N\times d}$ is a matrix of node positional embeddings, $\mathbf{X}_p$ initializes $\mathbf{P}$ (more information can be seen in Appendix C.2), $\mathbf{I}_d$ is an $d \times d$ identity matrix, and both κ_1 and κ_2 are non-negative trade-off coefficients. The first term guides $\mathbf{P}$ to be close to $\mathbf{X}_p$, while the middle term enforces adjacent nodes to stay closer in the positional latent space. A penalty term is lastly appended to ensure orthogonal feature channels for attaining a valid coordinate system. Minimizing $\mathcal{L}_p$ therefore enables a canonical positioning of nodes in the graph. We take an iterative gradient method to solve Eq. 4, and derive the iterative updating rule:

$$\mathbf{P}^{(k+1)} = \eta_1\mathbf{X}_p + (1-\eta_1)\Big((1+\eta_2)\hat{\mathbf{A}} - \eta_2(\mathbf{P}^{(k)}{\mathbf{P}^{(k)}}^T)\Big)\mathbf{P}^{(k)} \quad (5)$$

where $\mathbf{P}^{(0)} = \mathbf{X}_p$, $\eta_1 = \frac{1}{1+\kappa_1-2\kappa_2}$, $\eta_2 = \frac{2\kappa_2}{\kappa_1-2\kappa_2}$, and the stepsize is set as $\frac{\eta_1}{2}$. Both η_1 and η_2 are constant hyper-parameters searched from $\{0, 0.1, ..., 1.0\}$ by 0.1, and the case $\eta_1 = 1.0$ examines the effectiveness of the initial $\mathbf{X}_p$. By iteratively updating $\mathbf{P}^{(k)}$, the objective $\mathcal{L}_P$ can be progressively minimized to solve the optimization problem. In practical training, we need to normalize $\mathbf{P}^{(k)}{\mathbf{P}^{(k)}}^T$ to ensure numerical stability and benefit computational efficiency. Thus, Eq. 5 is enhanced into

$$\mathbf{P}^{(k+1)} = \eta_1\mathbf{X}_p + (1-\eta_1)\Big((1+\eta_2)\hat{\mathbf{A}} - \eta_2\sigma(\mathbf{P}^{(k)}\mathbf{W}{\mathbf{P}^{(k)}}^T)\Big)\mathbf{P}^{(k)} \quad (6)$$

where σ is a sigmoid function to produce values between 0 to 1, and $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a learnable mapping matrix to improve model capacity. Besides, we also add a tanh activation function between updating steps to make both positive and negative values in the derived coordinate system, i.e., $\mathbf{P}^{(k+1)} \leftarrow \mathrm{Tanh}(\mathbf{P}^{(k+1)})$. We further refer to this process as iterative positional encoding (IPE).

So far, the positional information of nodes can be encoded into a low-dimensional metric space by applying Eq. 6 recursively. To learn polynomial filter weights with awareness of node positions, it is empirically found that a simple yet effective non-linear mapping works well:

$$\beta_{k,i} = \sigma_p({\mathbf{W}^{(k)}}^T\mathbf{P}_i^{(k)} + \mathbf{b}^{(k)}) \quad (7)$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^d$ and $\mathbf{b}^{(k)} \in \mathbb{R}$ are learnbale parameters for polynomial order k, and σ_p is an activation function. As such, we manage to train models appropriately with guidance from node positions, while avoiding the possible overfitting induced by parameterizing each node arbitrarily with different filter weights. Moreover, model complexity can also be greatly reduced with the lowered number of trainable parameters from $N \times (K+1)$ to $(d+1) \times (K+1)$ where $d \ll N$ is feature dimension. Besides, we show our DSF framework, albeit with the simple mapping formula in Eq. 7, is able to deal with complex or even unusual graph cases in Section 5.4.

4.2.2 ***Local and Global Weight Decomposition.*** Though real-world networks exhibit rich and diverse local patterns, the global graph structure still matters, as it encodes some invariant graph properties while simultaneously pruning local noises. Accordingly, we decompose our node-specific filter weights $\beta_{k,i}$ into two independent coefficients γ_i and $\theta_{k,i}$ with multiplication, i.e., $\beta_{k,i} = \gamma_i\theta_{k,i}$. We call $\gamma_i \in \mathbb{R}$ the global filter weight responsible for capturing the global graph structure, and name $\theta_{k,i} \in (-1, 1)$ as local filter weight which is learned by the non-linear mapping in Eq. 7. As a benefit, the local coefficients can flexibly rescale and/or flip the sign of the global ones to capture node differences, while global connecting patterns can also be mined with diminished noisy information. In this paper, we call this technique Local and Global Weight Decomposition (LGWD). Additionally, we find that JacobiConv [42] also leverages a similar technique called PCD that decomposes the filter weight as $\alpha_k = \pi_k \prod_{s=1}^k \rho_s$ (we replace their symbols to avoid confusions with ours). This design aims to facilitate model training with a decreasing scale on $\{\alpha_k\}_{k=0}^K$ as k grows, and all the nodes still share the same parameter set. In contrast, our method works

on individual vertices through disentangling the globally shared and locally varied node coefficients.

4.3 Overall Algorithm

As our DSF framework is independent of any underlying model, it can flexibly improve any spectral GNNs. The overall pipeline of our DSF framework is presented in Appendix C. In practice, we find that the term $\mathbf{P}^{(k)}\mathbf{P}^{(k)^T}$ in Eq. 5 involves a high computational complexity in $O(N^2)$, and possibly causes a memory leak while running models on large-scale graphs. To alleviate this, we remove the corresponding term in the objective function, i.e., $\left\|\mathbf{P}^T\mathbf{P}-\mathbf{I}_d\right\|_F^2$ in Eq. 4, and reformulate it into a regularizer:

$$\mathcal{L}_{\text{Orth}} = \left\|\hat{\mathbf{P}}^{(K)}\hat{\mathbf{P}}^{(K)} - \mathbf{I}_d\right\|_F^2 \tag{8}$$

where $\hat{\mathbf{P}}^{(K)}$ is normalized from $\mathbf{P}^{(K)}$ such that each column of $\hat{\mathbf{P}}^{(K)}$ has zero mean and one l_2 norm. Accordingly, we set $\eta_2 = 0$ in Eq. 6 and have another hyper-parameter λ_{Orth} called orthogonal regularization parameter. In training, $\mathcal{L}_{\text{Orth}}$ is penalized with the task loss as $\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{Orth}}\mathcal{L}_{\text{Orth}}$. We further name this variant as DSF-x-R where x denotes the backbone GNN, while referring to the original one as DSF-x-I.

4.3.1 ***Model Analysis.*** The proposed DSF framework extends the existing spectral GNNs as

$$\sum_{k=0}^{K}\alpha_k P_k(\hat{\mathbf{L}})\mathbf{X} \rightarrow \sum_{k=0}^{K}\gamma_k \text{diag}(\theta_{k,1},\theta_{k,2},...,\theta_{k,N})P_k(\hat{\mathbf{L}})\mathbf{X}$$

where a functional space $\{g_i(\cdot)=\sum_{k=0}^{K}\gamma_k\theta_{k,i}P_k(\cdot)|\forall v_i\in\mathcal{V}\}$ made up of diverse filters is derived to enable node-wise learning. Specifically, the underlying graph region of individual node is mined with a different filter function. Existing spectral GNN models mostly learn with one filter function, and thereby can be strengthened with our DSF framework. Besides, in comparison with the advocated interpretability in BernNet [20], our DSF framework is able to offer better interpretability by further differentiating micro graph structures with learned diverse filters (see Section 5.4).

4.3.2 ***Time Complexity.*** Since our framework requires node-wise computations with positional features, compared to its underlying GNNs, the model complexity is increased by $O(N(f_p d+2Kd+2d+K+1)+2|\mathcal{E}|Kd+2N^2Kd)$ in our DSF-x-I. By the regularization term $\mathcal{L}_{\text{Orth}}$, we further mange to reduce it by $O(N^2Kd)$, and introduce our major model named DSF-x-R. The average running time is reported on Table 1. Despite the slightly higher computational overhead, we argue that DSF framework works still reasonably efficient in practice, especially considering the remarkable performance gains and the enhanced model interpretability (see Section 5.2 and Section 5.4).

5 EXPERIMENTS

In this section, we design experiments to answer the following research questions: (**RQ1**) How effective is our DSF framework in improving state-of-the-art spectral GNNs for node classification? (**RQ2**) Is there a negative impact on accuracy when each node is parameterized by a separate set of trainable weights? If so, would the proposed strategies, i.e., Position-aware Filter Weights and Local and Global Weight Decomposition (LGWD) take effect in alleviating this? and (**RQ3**) Could the proposed DSF framework indeed learn diverse and interpretable filters capturing both the common graph structure and regional heterogeneity?

Table 1: Average running time per epoch (ms)/average total running time (s). Although DSF-GPR-I is less efficient on large networks, DSF-GPR-R, (our major model) can reduce it by more than 75% on average (though reasonably slower than GPR-GNN).

Datasets	Small-scale	Large-scale	Average
GPR-GNN	1.10/2.24	0.98/5.01	1.08/2.74
DSF-GPR-I	5.96/12.19	40.34/131.77	12.21/33.93
DSF-GPR-R	2.49/6.29	3.02/14.48	2.59/7.78

5.1 Datasets and Experimental Setup

5.1.1 ***Datasets.*** We examine models over 11 real-world datasets from various domains including 6 heterophilic graphs as Chameleon, Squirrel [34], Wisconsin, Cornell, Texas [33] (webpage networks), and Twitch-DE [27, 34] (social network), as well as 5 homophilic graphs, i.e., Cora, Citeseer, Pubmed [35] (citation networks), Computers, and Photo [30, 36] (the Amazon co-purchase graphs). Detailed statistics are provided in the Appendix (see Table 4). We divide each dataset into 60%/20%/20% for training/validation/testing by following [9, 20, 42], and create 10 random splits for evaluation.

5.1.2 ***Baselines.*** To verify the effectiveness of the proposed DSF framework, we implement it upon three state-of-the-art spectral GNNs with trainable polynomial filters, i.e., GPR-GNN [9], BernNet [20], and JacobiConv [42]. Implementation details are provided in Appendix C. Therefore, we have six variants with names formatted as DSF-x-ϕ for $x\in\{\text{GPR},\text{Bern},\text{Jacobi}\}$ and $\phi\in\{\text{I},\text{R}\}$. For a more comprehensive comparison, we also consider another 8 baseline GNNs including GCN [24], GAT [40], ChebNet [11], APPNP [15], GNN-LF/HF [51], FAGCN [6], and PA-GNN [49].

5.1.3 ***Setup.*** We fix the number of hidden features $d = 64$ for all models, and set the polynomial order $K = 10$ to follow [9, 20, 42]. For each dataset, we tune the hyper-parameters of all models, including baselines with their specified parameter ranges, on the validation split using Optuna [1] for 200 trails. With the best hyper-parameters, we train models in 1,000 epochs using early-stopping strategy and a patience of 100 epochs. The average performance over 100 runs (10 runs $\times$ 10 splits) are reported. For reproducibity, our implementation and the searching space of hyper-parameters are available at https://github.com/jingweio/DSF.

5.2 RQ1. Overall Evaluation

To answer RQ1, we report the average node classification accuracies with a 95% confidence interval. From Table 2, we have the following observations: **1)** Spectral GNNs with trainable filters generally yield better classification results than other baseline models. This is because conventional GNNs typically fail to deal with complex linking patterns, e.g., in heterophilic graphs, using their fixed frequency response filters. Contrastively, GPR-GNN, BernNet, and JacobiConv can simulate different types of filters to learn from both assortative and disassortative label patterns. FAGCN is able to

Table 2: Node classification accuracies (%) ± 95% confidence interval over 100 runs. The row of PA-GNN [49]* lists the relative improvements of PA-GNN upon GPR-GNN based on the results obtained from its paper, where – denotes values not provided. Our Improv. gives the best relative improvements between our DSF variants over their common underlying model.

Datasets	Heterophilic Graphs						Homophilic Graphs				
	Chameleon	Squirrel	Wisconsin	Cornell	Texas	Twitch-DE	Cora	Citeseer	Pubmed	Computers	Photo
GCN [24]	67.22±0.43	54.21±0.41	59.45±0.72	52.76±1.17	61.66±0.71	73.94±0.15	88.13±0.25	77.00±0.27	89.07±0.11	91.06±0.12	93.99±0.12
GAT [40]	67.72±0.41	52.26±0.58	57.94±0.89	50.20±0.93	55.37±1.10	73.00±0.15	88.47±0.22	77.23±0.27	88.30±0.11	91.69±0.11	94.55±0.11
ChebNet [11]	64.85±0.44	48.14±0.33	80.93±0.72	77.98±1.00	75.83±1.20	73.73±0.14	87.64±0.21	76.93±0.24	89.91±0.11	91.65±0.12	95.27±0.07
APPNP [15]	53.66±0.33	36.08±0.36	81.23±0.64	81.29±0.78	79.42±1.05	72.65±0.11	88.70±0.21	77.77±0.24	89.93±0.09	91.62±0.10	94.92±0.09
GNN-LF [51]	54.29±0.36	36.87±0.33	59.85±0.60	62.90±0.98	61.88±0.95	73.03±0.13	88.90±0.25	77.35±0.29	88.89±0.10	91.12±0.11	95.13±0.08
GNN-HF [51]	55.22±0.42	35.45±0.30	68.17±0.72	72.98±1.02	66.66±1.34	71.92±0.13	89.01±0.19	77.74±0.23	89.53±0.10	90.73±0.10	95.26±0.09
FAGCN [6]	68.38±0.51	50.08±0.60	82.11±0.85	79.00±0.93	81.00±0.95	74.15±0.13	88.82±0.20	77.65±0.29	90.13±0.11	91.90±0.11	95.25±0.10
GPR-GNN [9]	69.01±0.50	55.39±0.33	82.72±0.85	80.81±0.78	81.66±1.02	74.07±0.18	89.03±0.20	77.63±0.28	90.10±0.44	92.34±0.13	95.34±0.09
DSF-GPR-I	71.18±0.52	57.08±0.29	**87.64**±0.79	84.76±0.90	85.44±1.05	74.58±0.16	**89.64**±0.20	78.03±0.26	90.26±0.08	92.49±0.12	95.64±0.07
DSF-GPR-R	**71.64**±0.55	**58.44**±0.30	87.43±0.74	**84.93**±0.90	**85.56**±0.93	**74.81**±0.14	89.63±0.17	**78.22**±0.29	**90.51**±0.07	**92.80**±0.12	**95.73**±0.08
Our Improv.	2.63%	3.05%	4.92%	4.12%	3.9%	0.74%	0.61%	0.59%	0.41%	0.46%	0.39%
PA-GNN [49]*	0.66%	1.28%	–	–	–	–	-0.09%	-0.74%	-0.03%	1.03%	0.02%
BernNet [20]	70.59±0.42	56.63±0.32	85.00±0.94	82.10±0.95	82.20±0.98	74.45±0.15	88.72±0.23	77.52±0.29	90.21±0.46	92.57±0.10	95.42±0.08
DSF-Bern-I	72.95±0.53	59.45±0.32	**88.23**±0.81	**85.07**±0.93	**84.59**±1.07	74.96±0.15	89.05±0.22	**78.32**±0.27	90.40±0.10	92.76±0.10	95.73±0.07
DSF-Bern-R	**73.60**±0.53	**59.99**±0.30	88.02±0.91	84.29±0.93	84.42±1.00	**75.00**±0.15	**89.10**±0.22	78.27±0.26	**90.52**±0.10	**92.84**±0.10	**95.79**±0.06
Our Improv.	3.01%	3.36%	3.23%	2.97%	2.39%	0.55%	0.38%	0.80%	0.31%	0.27%	0.37%
JacobiConv [42]	73.71±0.42	57.22±0.24	83.21±0.68	82.34±0.88	82.42±0.90	74.34±0.12	89.24±0.19	77.81±0.29	89.50±0.47	92.26±0.10	95.62±0.06
DSF-Jacobi-I	74.88±0.39	58.26±0.26	85.34±0.74	**84.54**±0.81	83.68±1.12	74.65±0.13	89.54±0.19	78.18±0.26	89.78±0.09	92.38±0.11	**95.76**±0.07
DSF-Jacobi-R	**75.00**±0.38	**59.23**±0.27	**86.13**±0.70	84.39±0.88	**84.46**±0.81	**74.75**±0.15	**89.66**±0.19	**78.23**±0.25	**90.07**±0.10	**92.44**±0.11	95.75±0.08
Our Improv.	1.29%	2.01%	2.92%	2.20%	2.04%	0.41%	0.42%	0.42%	0.41%	0.18%	0.14%

capture both low- and high-frequency information but is limited as they can only model pairwise node relationship. **2)** The proposed DSF framework consistently produces performance boost over its underlying models, especially on heterophilic graphs with the maximal improvement up to 4.92%. This can be mainly explained by the diverse characteristics inherent in their local graph patterns, as shown in Figure 2. By comparison, the existing spectral GNN models assume the *homogeneous* spectral filtering, and neglect regional *heterogeneity* at different graph localities. **3)** For Twitch-DE dataset, albeit also being heterophilic graphs, we only observe an marginal improvement made by our framework. This is due to the fact that its local structural patterns are naturally similar and distributed uniformly across the graph, as indicated by the concentrated histogram in the Appendix (see Figure 6). Similar results can be spotted on homophilic graphs with assortative linking patterns, which can be easily modeled with classic *homogeneous* spectral filtering. The refined classification accuracies made by DSF is mainly contributed by dealing with some sudden changes on the graph boundaries, e.g., between different community regions or social circles. **4)** We conduct comparisons with PA-GNN [49] which also tries to learn node-specific parameter offsets. As no codes are publicly available for PA-GNN, we simply compute the relative improvements upon its base model GPR-GNN by copying the values from their paper (listed in the row of PA-GNN [49]*). In general, PA-GNN shows marginal or even negative performance gains. We conjecture that PA-GNN might encode different sources of information for predicting the offsets with small value constraint, thus limiting their performance. **5)** Interestingly, it is noted our variant DSF-x-R not only decreases the model complexity of DSF-x-I but also achieves higher performance gains on average. This is partially because DSF-x-I minimizes the orthogonal penalty, i.e., the last term in Eq. 4, mainly by means of the iterative aggregation on a non-sparse graph computed by $\mathbf{P}^{(k)}\mathbf{W}\mathbf{P}^{(k)^T}$. Despite the theoretical convergence, aggregating features on dense graph is prone to mistakenly preserve noises and cause model overfitting. On the other hand, DSF-x-R with the regularization loss $\mathcal{L}_{\text{Orth}}$ offers a more flexible and accurate control, provides extra supervisory signals directly operated on model parameters, and could also benefit from the advanced optimization technique such as Adam [23] algorithm.

Table 3: Reduced classification accuracies (%) of our DSF framework compared to base models while learning without IPE.

Datasets	Chameleon	Squirrel	Wisconsin	Cornell	Texas	Photo
DSF-GPR w/o IPE	22.62	22.55	5.81	11.20	11.10	1.51
DSF-Bern w/o IPE	17.47	18.65	4.72	6.25	6.54	3.59
DSF-Jacobi w/o IPE	24.64	26.93	1.10	3.10	5.88	1.53
Average Reduction	**21.58**	**22.71**	**3.88**	**6.85**	**7.84**	**2.21**

5.3 RQ2. Ablation Study

This subsection aims to validate our designs through ablation study. We earlier argue that it is inappropriate to directly parameterize each node a separate set of trainable filter weights. To provide empirical evidences, we first experiment with our DSF framework in node classification tasks while ablating the module of iterative positional encoding (IPE). That is to directly make $N \times (K+1)$ filter weights w.r.t. nodes to be trained as model parameters. We then report the downgraded model performance compared to the underlying models in Table 3, where six datasets are experimented for illustration. As observed, learning without IPE leads to a clear accuracy drop, notably on networks Chameleon and Squirrel with complex connecting patterns and relative a large number of nodes. This confirms our early conjecture as well as the importance of the proposed IPE strategy. In this work, we also constrain the channel

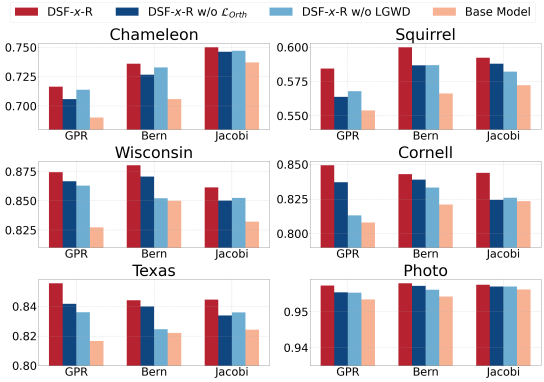
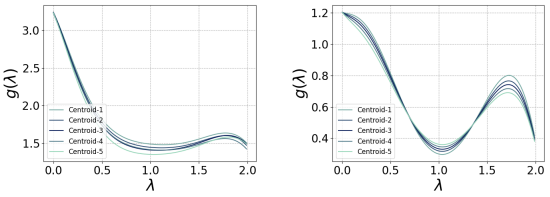


Figure 3: Ablation study of DSF framework on six datasets with our variants DSF-x-R for all $x \in \{\text{GPR}, \text{Bern}, \text{Jacobi}\}$ as an example.



(a) Citeseer **(b) Photo**

Figure 4: Diverse filters on homophilic graphs, which are learned to be similar due to the intrinsic assortative linking patterns distributed uniformly on these networks. Our DSF presents one general framework which can be adaptive to different types of networks.

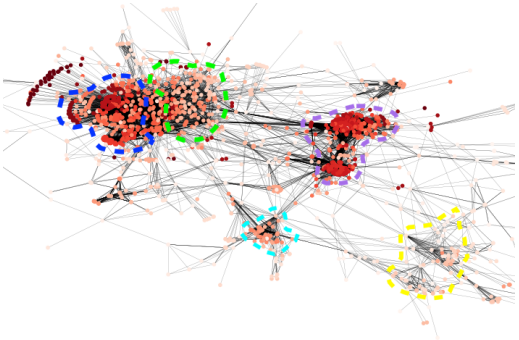


Figure 5: Visualization of node-specific filter weights on Chameleon dataset, where a few border nodes are cropped away for better picturing. We annotate the captured regional distinctions using irregular circles with different colors.

orthogonality while encoding positional features, and introduce a technique called Local and Global Weight Decomposition (LGWD). To examine their effectiveness, we conduct comprehensive ablation study over six datasets in node classification. For simplicity, we take DSF-x-R, one variant of our framework, as an example. Similar results can be obtained on the other variants. From Figure 3, two conclusions can be drawn. First, removing either $\mathcal{L}_{\text{Orth}}$ or LGWD from our framework causes an evident performance downgrade, validating the usefulness of these two developed techniques. Second, the ablated variants still outperform their underlying models. This further underpins the advantages offered by learning diverse filters with awareness of positional information.

5.4 RQ3. Analysis on Diverse Filters

We now answer RQ3 by first plotting the diverse filter functions learned by our DSF with BernNet as the illustrative base model. Without loss of generality, we cluster the node-specific filter weights, i.e., $\{[\beta_{0,i}, \beta_{1,i}, ..., \beta_{K,i}]^T | \forall v_i \in \mathcal{V}\}$, into five groups with k-means algorithm [22], and only plot the filters w.r.t. the representative centroids for better visualization. From Figure 1 on heterophilc graphs, we observe a group of function curves showing similar overall shapes but different local aspects. This implies that the proposed DSF framework is able to grasp both conformal and disparate regional information on the graph. In addition, we also draw the diverse filters learned from homophilic graphs including Citeseer and Photo. These graph networks have assortative mixing patterns with homogeneous local structures. The learned filter functions produce almost identical curves fluctuating within a reasonable interval in Figure 4. It further shows our DSF framework could work on different types of graphs. On the other hand, we present t-SNE [39] visualization of the node-specific filter weights. The color likeness reflects the corresponding similarity. From Figure 1d, disparate regional patterns can be distinguished, and far-reaching nodes with conformal local subgraphs still learn similar filter weights. Besides, we notice the node in graph center displays a salient white color, obviously divergent from its neighborhood. This is because such vertex possesses the unique local context characterized by the densest graph neighborhood, and thereby deserves a special treatment. This phenomenon shows the flexibility of our DSF framework in dealing with complex or even unusual cases, instead of learning some strict relationships, e.g., nearby nodes ought to possess similar local structures (similar filter weights) and otherwise. We also present the visualization on graphs in relative larger scale, such as Chameleon network in Figure 5. More results can be found in Appendix D. These analytical results demonstrate the strong interpretability of our DSF framework.

6 CONCLUSION

This paper focuses on learning GNNs on complex graphs with regional *heterogeneity* from the spectral perspective. We show that most existing spectral GNNs implicitly assume invariance between local networking patterns, and are restricted in the *homogeneous* spectral filtering, thus limiting their performance. To this end, we propose a novel *diverse* spectral filtering (DSF) framework generalizing spectral GNNs to better exploit rich and diverse local graph information. Both theoretical and empirical investigations validate the effectiveness of our DSF framework and its enhanced interpretability by learning diverse filters.

ACKNOWLEDGMENTS

The work was partially supported by the following: Jiangsu Science and Technology Programme (Natural Science Foundation of Jiangsu Province) under No. BE2020006-4; Key Program Special Fund in XJTLU under No. KSF-T-06.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[2] Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*. https://openreview.net/forum?id=i80OPhOCVH2

[3] Muhammet Balcilar, Pierre Héroux, Benoit Gauzere, Pascal Vasseur, Sébastien Adam, and Paul Honeine. 2021. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*. PMLR, 599–608.

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

[5] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[6] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3950–3957.

[7] Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. 2022. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[8] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1172–1180.

[9] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive universal generalized PageRank graph neural network. In *International Conference on Learning Representations*. https://openreview.net/forum?id=n6jl7fLxrP

[10] Fan R. K. Chung. 1996. Spectral graph theory.

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[12] Yushun Dong, Kaize Ding, Brian Jalaian, Shuiwang Ji, and Jundong Li. 2021. AdaGNN: Graph neural networks with adaptive frequency response filter. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 392–401.

[13] Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2022. Position information in transformers: An overview. *Computational Linguistics* 48, 3 (2022), 733–763.

[14] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2022. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=wTTjnvGphYj

[15] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations (ICLR)*.

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[17] Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45.

[18] Jingwei Guo, Kaizhu Huang, Xinping Yi, and Rui Zhang. 2022. ES-GNN: Generalizing graph neural networks beyond homophily with edge splitting. *arXiv preprint arXiv:2205.13700* (2022).

[19] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2021. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[20] Mingguo He, Zhewei Wei, Hongteng Xu, et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems* 34 (2021), 14239–14251.

[21] Md Amirul Islam, Sen Jia, and Neil DB Bruce. 2020. How much position information do convolutional neural networks encode? *arXiv preprint arXiv:2001.08248* (2020).

[22] Anil K Jain and Richard C Dubes. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[25] Rongjie Lai and Stanley Osher. 2014. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing* 58, 2 (2014), 431–449.

[26] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

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

[28] Vijay Lingam, Chanakya Ekbote, Manan Sharma, Rahul Ragesh, Arun Iyer, and Sundararajan Sellamanickam. 2021. A piece-wise polynomial filtering approach for graph neural networks. *arXiv preprint arXiv:2112.03499* (2021).

[29] Xiaojun Ma, Qin Chen, Yuanyi Ren, Guojie Song, and Liang Wang. 2022. Meta-weight graph neural network: Push the limits beyond global homophily. In *Proceedings of the ACM Web Conference 2022*. 1270–1280.

[30] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[31] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4602–4609.

[32] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

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

[34] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[35] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[36] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018* (2018).

[37] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. 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 (2013), 83–98.

[38] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. 2021. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021).

[39] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9, 11 (2008).

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ accepted as poster.

[41] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. 2022. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*.

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

[43] Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series* 2, 9 (1968), 12–16.

[44] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 6861–6871.

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

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[47] Liang Yang, Mengzhe Li, Liyang Liu, Chuan Wang, Xiaochun Cao, Yuanfang Guo, et al. 2021. Diverse message passing for attribute with heterophily. *Advances in Neural Information Processing Systems* 34 (2021), 4751–4763.

[48] Liang Yang, Wenmiao Zhou, Weihang Peng, Bingxin Niu, Junhua Gu, Chuan Wang, Xiaochun Cao, and Dongxiao He. 2022. Graph neural networks beyond compromise between attribute and topology. In *Proceedings of the ACM Web Conference 2022*. 1127–1135.

[49] Yuxin Yang, Yitao Liang, and Muhan Zhang. [n. d.]. PA-GNN: Parameter-adaptive graph neural networks. ([n. d.]).

[50] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.

[51] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*. 1215–1226.

A FURTHER REMARKS ABOUT RELATED WORK

Existing GNNs are often divided into spatial-based and spectral-based methods. The former is mainly built upon a message-passing framework [3, 16] where nodes exchange information with their spatial neighbors. The latter is rooted in graph signal processing [37] and spectral graph theory [10], and are mostly divided into two categories. One class is spectral GNNs with fixed filters: GCN [24] truncates Chebyshev polynomials to a simple first-order, and works as a low-pass filter [44]. APPNP [15] constructs filter functions with personalized PageRank [32] to overcome the over-smoothing problem [26]. GNN-LF/HF [51] is derived from the perspective of graph optimization to simulate low/high-pass filters. In the second class, spectral GNNs are mostly designed with trainable filters: ChebNet [11] approximates the filtering operation with Chebyshev polynomials whose coefficients are learnable. AdaGNN [12] learns adaptive filters to model each feature channel independently. GPR-GNN [9] extends APPNP [15] by directly parameterizing its filter weights and training them with gradient descent. ARMA [5] takes rational filter functions while approximating them still with polynomials. BernNet [20] employs positive weight constraints in learning spectral filters with the Bernstein polynomial approximation. Wang and Zhang [42] further analyze the expressive power of spectral GNNs in a general form, and propose JacobiConv with an orthogonal polynomial basis and feature-wise filter learning.

B PROOF OF PROPOSITION 1

Proof. We denote $f(x) = \sum_{k=0}^{K} \alpha_k P_k(x)$, and substituting $x = \xi\hat{x}$ gives us $f(\xi\hat{x}) = \sum_{k=0}^{K} \alpha_k P_k(\xi\hat{x})$. As the maximum order on variable x is K, we can always express $f(\xi\hat{x})$ in a power series with new coefficient set $\{\omega_k\}_{k=0}^{K}$ where $\omega_k \in \mathbb{R}$, i.e., $f(\xi\hat{x}) = \sum_{k=0}^{K} \omega_k (\xi\hat{x})^k$. Moreover, since ξ is a constant, we can view $f(\xi\hat{x})$ as a function of variable $\hat{x}$, i.e., $g(\hat{x}) = \sum_{k=0}^{K} (\omega_k \xi^k)\hat{x}^k$, Similarly, with the expressive power of polynomial basis $P_k(\cdot)$, we can always find a new coefficient set $\{\beta_k\}_{k=0}^{K}$ where $\beta_k \in \mathbb{R}$ making $g(\hat{x}) = \sum_{k=0}^{K} \beta_k P_k(\hat{x})$. Therefore, we have $f(x) = g(\hat{x})$ where these two functions are made up of the same polynomial basis $P_k; [0, 2] \to \mathbb{R}$ and two different coefficient sets $\{\alpha_k\}_{k=0}^{K}$ and $\{\beta_k\}_{k=0}^{K}$. □

C IMPLEMENTATION DETAILS

The overall pipeline of the proposed DSF framework is detailed in Algorithm 1. We update node positional features and hidden states in an iterative and parallel scheme (see the lines 11-17). Noting in [14], similar approaches can be found. In our experiments, we showcase it over three SOTA baselines including GPR-GNN, BernNet, and JacobiConv. A comprehensive summary of their designed trainable spectral filters can be found in [42]. We also provide other important implementation details in the following.

C.1 Base Model Information

GPR-GNN [9] as backbone: The authors experiment GPR-GNN with several initializing strategies on $\{\alpha_k\}_{k=0}^{K}$ and take the optimal one for the final evaluation. To take advantage of this, we adopt the same strategy to initialize our $\{\gamma_k\}_{k=0}^{K}$ before training. We call the resulted variants DSF-GPR-I and DSF-GPR-R.

Table 4: Statistics of real-world datasets, where ⋆ denotes large-scale graphs. Both $\mathcal{H}$ [50] and $\mathcal{H}_{\text{class}}$ [27] (considering class-imbalance problem) measure graph homophily ratio from 0 to 1. Albeit the relative high value given by $\mathcal{H} = 0.63$, Twitch-DE is essentially a heterophlic graph with class-imbalanced issue, as suggested by $\mathcal{H}_{\text{class}} = 0.14$.

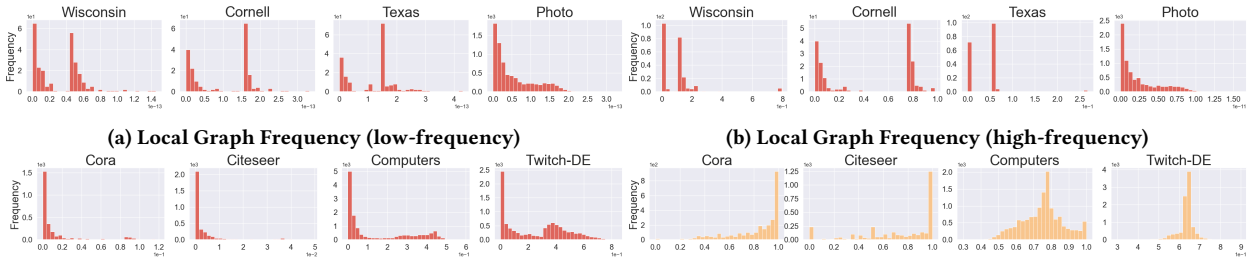
Dataset	# Nodes	# Edges	# Features	# Classes	$\mathcal{H}$	$\mathcal{H}_{\text{class}}$
Chameleon	2,227	36,101	2,325	5	0.23	0.06
Squirrel	5,201	217,073	2,089	5	0.22	0.03
Wisconsin	251	499	1,703	5	0.21	0.09
Cornell	183	295	1,703	5	0.30	0.05
Texas	183	309	1,703	5	0.11	0.00
Twitch-DE	9,498	153,138	2,545	2	0.63	0.14
Cora	2,708	5,429	1,433	7	0.81	0.77
Citeseer	3,327	4,732	3,703	6	0.74	0.63
Pubmed⋆	19,717	44,338	500	3	0.80	0.66
Computers⋆	13,752	245,861	767	10	0.78	0.70
Photo	7,650	119,081	745	8	0.83	0.77

Algorithm 1 Framework of *diverse* spectral filtering

Input: Node set: $\mathcal{V}$, Laplacian matrix: $\hat{\mathbf{L}}$, raw node content and positional features: $\mathbf{X} \in \mathbb{R}^{N\times f}, \mathbf{X}_p \in \mathbb{R}^{N\times f_p}$, polynomial basis: $P_k(\cdot)$, hyper-parameters: $K, \eta_1, \eta_2, \lambda_{\text{Orth}}$, ground truth labels for training: $\{\mathbf{y}_i \in \mathbb{R}^C | \forall v_i \in \mathcal{V}_{\text{trn}}\}$, activation function in Eq. 7: $\sigma_p(\cdot)$, and DSF-mode: $\phi \in \{\text{I}, \text{R}\}$.

Param: $\mathbf{W}_x \in \mathbb{R}^{f\times d}, \mathbf{b}_x \in \mathbb{R}^d, \mathbf{W}_p \in \mathbb{R}^{f_p\times d}, \mathbf{b}_p \in \mathbb{R}^d, \mathbf{W} \in \mathbb{R}^{d\times d}, \mathbf{W}_F \in \mathbb{R}^{d\times C}, \mathbf{b}_F \in \mathbb{R}^C, \{\mathbf{W}^{(k)} \in \mathbb{R}^d, \mathbf{b}^{(k)} \in \mathbb{R} | k = 0, 1, ..., K\}$, and $\{\gamma_k \in \mathbb{R} | \forall k = 0, 1..., K\}$.

1: Set $\eta_2 = 0$ if ϕ is R
2: // *Projection into latent space.*
3: $\mathbf{X}_i \leftarrow \text{ReLU}(\mathbf{W}_x^T \mathbf{X}_i + \mathbf{b}_x)$ for all $v_i \in \mathcal{V}$.
4: $\mathbf{P}_i^{(0)} \leftarrow \text{Tanh}(\mathbf{W}_p^T \mathbf{X}_{p_i} + \mathbf{b}_b)$ for all $v_i \in \mathcal{V}$.
5: // *enabled only for training.*
6: $\mathbf{X} \leftarrow \text{Dropout}(\mathbf{X}), \mathbf{P}^{(0)} \leftarrow \text{Dropout}(\mathbf{P}^{(0)})$.
7: // *Initialization.*
8: $\beta_{0,i} \leftarrow \gamma_0 \sigma_p(\mathbf{W}^{(0)^T} \mathbf{P}_i^{(0)} + \mathbf{b}^{(0)})$ for all $v_i \in \mathcal{V}$.
9: $\mathbf{Z}^{(0)} \leftarrow \text{diag}(\beta_{0,1}, \beta_{0,2}, ..., \beta_{0,N}) P_0(\hat{\mathbf{L}})\mathbf{X}$.
10: // *Iterate polynomial orders while updating node positions.*
11: **for** $k = 1, 2, ..., K$ **do**
12: // *Update node positional features.*
13: $\mathbf{P}^{(k)} \leftarrow \mathbf{P}^{(k-1)}$ using Eq. 6 with $\eta_1, \eta_2, \mathbf{W}$.
14: // *Update node hidden states.*
15: $\beta_{k,i} \leftarrow \gamma_k \sigma_p(\mathbf{W}^{(k)^T} \mathbf{P}_i^{(k)} + \mathbf{b}^{(k)})$ for all $v_i \in \mathcal{V}$.
16: $\mathbf{Z}^{(k)} \leftarrow \mathbf{Z}^{(k-1)} + \text{diag}(\beta_{k,1}, \beta_{k,2}, ..., \beta_{k,N}) P_k(\hat{\mathbf{L}})\mathbf{X}$.
17: **end for**
18: $\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{W}_F^T \mathbf{Z}_{[i,:]}^{(K)} + \mathbf{b}_F), \forall v_i \in \mathcal{V}$. // *Prediction.*
19: $\mathcal{L}_{\text{task}} = -\frac{1}{|\mathcal{V}_{\text{trn}}|} \sum_{v_i \in \mathcal{V}_{\text{trn}}} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i)$. // *Training.*
20: **if** ϕ is R **then**
21: Minimize $\mathcal{L}_{\text{task}} + \lambda_{\text{Orth}} \mathcal{L}_{\text{Orth}}$ with $\mathcal{L}_{\text{Orth}}$ computed in Eq. 8.
22: **else**
23: Minimize $\mathcal{L}_{\text{task}}$.
24: **end if**



(a) Local Graph Frequency (low-frequency)

(b) Local Graph Frequency (high-frequency)

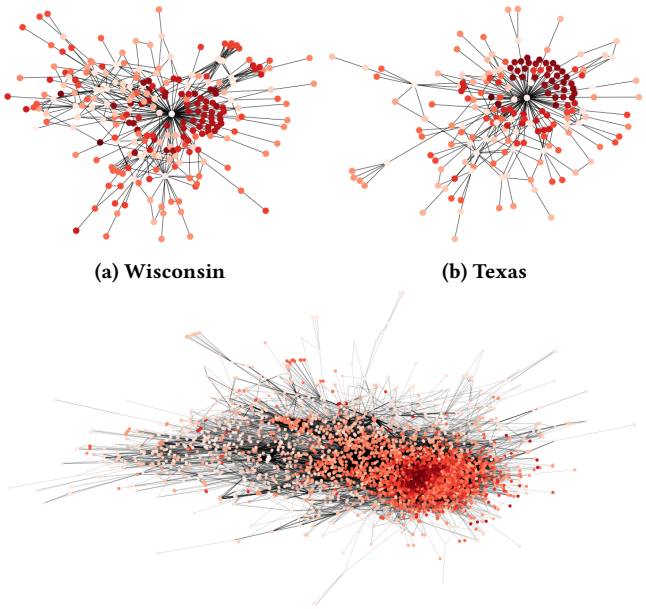
(c) Local Graph Frequency (middle-frequency)

(d) Local Label Homophily

Figure 6: Additional distributions of two essential graph properties (Better viewed in color). (a), (b), and (c) respectively show the statistics of local graph frequency using eigenvectors with low-, high-, and middle-frequency.

Table 5: Classification accuracies (%) on homophilic graphs with sparse splits.

Datasets	Cora	Citeseer	Pubmed	Computers	Photo
GPR-GNN	76.05±0.48	65.39±0.44	83.30±0.24	85.68±0.16	91.88±0.18
DSF-GPR-I	77.75±0.42	**66.83±0.37**	83.74±0.15	86.94±0.16	92.39±0.14
DSF-GPR-R	**77.94±0.48**	66.77±0.34	**84.33±0.13**	**87.65±0.15**	**92.52±0.12**
Our Improv.	1.89%	1.38%	1.03%	1.97%	0.64%



(a) Wisconsin

(b) Texas

(c) Squirrel

Figure 7: Additional visualization of node-specific filter weights.

BernNet [20] as backbone: As stated in the original paper, α_k is constrained to be non-negative. To follow up, we apply the same limit to our $\beta_{k,i} = \gamma_k \theta_{k,i}$ by making $\gamma_i \leftarrow \text{ReLU}(\gamma_i)$ and taking σ_p as a sigmoid function in Eq 7 to restrict $\theta_{k,i}$ within $(0, 1)$. The resulted models are named as DSF-Bern-I and DSF-Bern-R.

JacobiConv [42] as backbone: The authors leverage a technique called PCD, which decomposes the filter weight into multiple coefficients, such as $\alpha_k = \pi_k \prod_{s=1}^{k} \rho_s$. To deploy our DSF framework over JacobiConv, we make $\pi_k = \gamma_k$, and transform ρ_s into $\rho_{s,i}$ which is learned using Eq. 7. The produced variants are finally referred to as DSF-Jacobi-I and DSF-Jacobi-R.

C.2 Initialization on IPE

The choice of initializing node positional embeddings $\mathbf{X}_p$ is important, which usually requires to be permutation-invariant and distance-sensitive. In this work, we leverage two popular and efficient methods. The first one is widely used and called Laplacian Positional Encoding [4] (LapPE). It basically takes the decomposed eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$, and each node v_i is assigned with $\mathbf{p}_i^{\text{Lap}} = [\mathbf{u}_{1,i}, \mathbf{u}_{2,i}, ..., \mathbf{u}_{f_p,i}]^T \in \mathbb{R}^{f_p}$, where $f_p \ll N$ is the predefined feature number. The other approach has been recently proposed based on the random walk diffusion process, named RWPE [14]. It aims to moderate the sign ambiguity issue in LapPE, and is formulated as $\mathbf{p}_i^{\text{RW}} = [\mathbf{RW}_{i,i}^1, \mathbf{RW}_{i,i}^2, ..., \mathbf{RW}_{i,i}^{d_f}]^T \in \mathbb{R}^{f_p}$, where $\mathbf{RW} = \mathbf{AD}^{-1}$. To increase model capacity and efficiency, before starting iterative update, we map $\mathbf{X}_p$ into a latent space with dimension d (see line 3 in Algorithm 1).

D ADDITIONAL EXPERIMENTAL RESULTS

We also investigate the model performance in case of limited supervision. To do so, we follow the sparse splitting [9] on homophilic graphs, i.e, 2.5%/2.5%/95% for training/validation/testing. Table 5 lists the classification accuracies (%), where noticeable improvements are made by our DSF framework upon GPR-GNN albeit limited supervision. Figure 7 provides more results on the visualization of node-specific filter weights. As the same type of networks, Wisconsin and Texas yield similar pictures to Cornell in the main text. For Squirrel dataset, we can see a gradual shift on the color depth of nodes from graph center to the border, which coincides with our conclusions in the main text about our DSF framework capturing regional heterogeneity.