LDF-GNN: Alleviating Structural Misguidance in Graph Neural Networks via Local Dynamic Fusion Filter

Zhongming Mei*, Rui Duan†, Wenzhuo Fan*, Mingjian Guang*
*College of Information Science and Technology, Donghua University, Shanghai, China
†School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China
Email:13918342732@163.com, duan@gzhu.edu.cn, yuxinzfw@gmail.com, guangmingjian@dhu.edu.cn

***Abstract*—Graph neural networks (GNNs) have emerged as a fundamental framework for graph representation learning. A major research direction in this field is spectral graph filters, which process different frequency components of graph signals through polynomial approximations. However, most existing studies primarily focus on optimizing the design of the graph filters while ignoring the interaction between input signals and the filtering process and applying the same filtering mechanism for all regions of the graph. This uniform approach may lead to the structural misguidance problem, making the filter susceptible to regional variations or heterophilic connections. To address this issue, we propose LDF-GNN, a spectral GNN based on the local dynamic fusion (LDF) filter. Specifically, we first generate localized graph signals using a random walk-based method to ensure adaptive filtering across different regions. Then we employ the dynamic fusion (DF) filter which dynamically adjusts its filtering mechanism in response to the frequency characteristics of each localized signal. By dynamically integrating input signals with the filtering process, LDF-GNN can capture both local discrepancies and structural heterophily to alleviate the structural misguidance problem. Experimental results demonstrate that LDF-GNN achieves state-of-the-art performance, and ablation studies further validate the importance of its two dynamic properties in mitigating structural misguidance.**

***Index Terms*—Graph Neural Networks, Graph Representation Learning, Spectral Graph Filtering, Structural Misguidance**

I. Introduction

Graph neural networks (GNNs) have become a cornerstone in graph representation learning, demonstrating strong performance across diverse applications such as social networks [1], recommender systems [2], and biological networks [3]. By aggregating information from neighboring nodes through graph convolution operations, GNNs enhance node representations for graph related tasks, including node classification [4], [5] and link prediction [6], [7]. Among various GNN approaches, spectral methods have emerged as a significant direction [8].

Spectral graph filters transform graph signals into the spectral domain where models are mostly represented by polynomial approximations [9]–[20]. For example, ChebNet [9] and BernNet [14] use Chebyshev and Bernstein polynomials to approximate the spectral filter respectively. These spectral GNNs handle the frequency components differently to learn graph signals. Despite their improved performance, existing spectral GNNs still suffer from structural misguidance.

Structural misguidance behaves differently on homophilic and heterophilic graphs. In highly heterophilic graphs, adjacent nodes often have distinct features and labels [21], so excessive cross-type connections can distort the information distribution, causing models to aggregate mismatched features and learn inaccurately. In contrast, homophilic graphs typically connect nodes that share similar characteristics [21]. However, some regions may be significantly different from other regions, so filters learned on the global graph may fail to capture the characteristics of these local structures. For instance, in a social network graph [1], the overall graph can be homophilic while communities differ significantly. In this case, a global filter may not adapt to the feature variations in different regions, leading to the loss of local information.

The main reason why most existing spectral filters cannot solve the structural misguidance problem is that they only emphasize filter design while ignoring the input signals of the filter. The design of filters, whether fixed or learnable, tends to rely on fixed or globally uniform models. These filters assume that the same filtering mechanism applies to all regions in the global graph. Therefore, models may be misled by global noise or local discrepancies in the graph. Methods based only on low-pass filters [9]–[12] suffer from structural misguidance on both homophilic and heterophilic graphs. Some other methods [13], [18]–[20] have achieved good results in solving the problem of heterophilic graphs, but their filter inputs are of a fixed form. So, they fail to address local discrepancies in homophilic graphs, and still suffer from structural misguidance in such graphs.

To solve the above problem, we propose the **l**ocal **d**ynamic **f**usion (LDF) filter to mitigate structural misguidance. The model consists of two components: localized random walk graph reduction and frequency-aware feature fusion. During training, we first apply a random walk–based graph reduction to sample multiple local subgraphs and generate the corresponding local graph signals. Each signal captures the local relationships independently, enabling more accurate learning of local structural information. We then apply the **d**ynamic

fusion (DF) filter on each local graph signal. Inspired by Gaussian filters, we design the frequency-aware feature fusion approach. In this process, the model dynamically adjusts the ratio of low- and high-pass filters based on the frequency characteristics of the current local graph signal, thereby adaptively focusing on the homophilic and heterophilic relationships in different regions of the graph. The two dynamic properties of the model are the dynamic change of the local graph signals and the dynamic adjustment of the filters.

Unlike existing spectral GNNs that focus only on filter design, the LDF method emphasizes the dynamic interaction between the input signals and the filtering mechanism. The LDF-GNN model can adjust its parameters according to the structural characteristics of different local graph signals, and thus adapt to the varying homophilic levels in the global graph to avoid the influence of structural misguidance. In summary, our contributions are as follows:

- We analyze structural misguidance in spectral GNNs, explain why most spectral filters fail to address it, and introduce the local dynamic fusion filter to alleviate it.
- We propose a random walk–based graph reduction method to generate diverse local graph signals from the global graph. Furthermore, we design the dynamic fusion filter that adapts its filtering mechanism based on the frequency characteristics of these local signals. By integrating these two components, the model can capture both local discrepancies and structural heterophily.
- Extensive experiments on seven benchmark datasets demonstrate that LDF-GNN outperforms state-of-the-art spectral GNNs, especially on graphs with high heterophilic levels or significant local discrepancies.

II. Related Work

Spectral graph neural networks (GNNs) define graph convolution in the spectral domain of the graph Laplace matrix. The eigenvalues and eigenvectors of the graph Laplacian matrix convert the node features into frequency domain representations. Based on spectral graph theory, these methods can filter graph signals to extract local and global structural information.

Spectral CNN [22] first introduces spectral graph convolution and generalizes CNN to the graph Laplacian spectrum. ChebNet [9] uses Chebyshev polynomial approximation to design localized convolutional filters on graphs, which reduces the computational complexity. GCN [10] simplifies spectral filters by a localized first-order approximation for scalable semi-supervised graph learning. CayleyNets [12] employ Cayley polynomials for efficient filtering computations in specific frequency bands. ARMA [11] adopts an auto-regressive moving average mechanism to provide a more flexible frequency response and better captures the global graph structure. GPR-GNN [13] uses a learnable PageRank architecture that processes both high- and low-frequency graph signals. BernNet [14] approximates spectral filters with K-order Bernstein polynomials, allowing it to learn arbitrary spectral filters. EvenNet [15] uses an even-polynomial graph filter that ignores odd-hop neighbors to improve robustness. ChebNetII [16] enhances ChebNet with a Chebyshev interpolation spectral filter. JacobiConv [17] overcomes the nonlinearity problem using the Jacobi basis, enabling adaption to a wide range of weighting functions. PCNet [18] and TFE-GNN [19] use two-fold and triple filtering mechanisms to unify homophily and heterophily in graphs. AFGNN [20] combines low-pass, medium-pass, and high-pass filters with an attentional mechanism to capture all frequency information from a large-scale graph.

Recently, more and more studies [18]–[20] focus on the fusion of spectral filters, and some other works [23] explore numerical inputs to the filters. However, most works concentrate on the design of the spectral filter itself, assuming a uniform filtering mechanism across the graph. This assumption leads to structural misguidance problems, as they ignore the dynamic interaction between the input signals of the filter and its model design. Therefore, we propose LDF-GNN to dynamically adjust the filters according to frequency differences of local graph signals. By dynamically fusing high- and low-pass filters, LDF-GNN mitigates structural misguidance and improves performance.

III. Preliminaries

Notations. We consider an undirected graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ denotes the node set, $\mathcal{E}$ denotes the edge set, and $\mathbf{X}$ is the node feature matrix, where $n = |\mathcal{V}|$ is the number of nodes. Let $\mathbf{A}$ be the adjacency matrix and $\mathbf{D}$ the degree matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$. With self-loops, the symmetrically normalized adjacency matrix is $\tilde{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-1/2}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-1/2}$, and the normalized Laplacian is $\mathbf{L} = \mathbf{I} - \tilde{\mathbf{A}}$. We use the eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$, where $\mathbf{U}$ contains the graph Fourier basis and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ collects the frequency component of the graph signal.

Spectral graph filtering A spectral graph filter $g(\cdot)$ operates in the Fourier domain as a diagonal frequency response:

$$y = g(\mathbf{L}) \cdot s = \mathbf{U} \cdot g(\Lambda) \cdot \mathbf{U}^{T} \cdot s, \tag{1}$$

where s denotes the time-domain signal of the graph [17]. Existing spectral graph filters use polynomials to approximate the filtering operator $g(\Lambda)$:

$$g(\Lambda) \approx \sum_{i=0}^{K-1} \alpha_i \Lambda^i, \tag{2}$$

where $\alpha_i \in \mathbb{R}^K$ denotes a vector of polynomial coefficients and K represents the order of the approximate polynomial.

Homophily of graph The graph homophily describes the similarity between adjacent nodes and reflects the relationship between node attributes and graph structure. Its metrics can be quantified as node homophily [24], edge homophily [25], class homophily [26], and adjusted homophily [27]. We use edge homophily to measure the level of homophily in the graph:

$$\mathcal{H} = \frac{\left|\{(v_i, v_j)\} \in E : y_{v_i} = y_{v_j}\right|}{|E|} \tag{3}$$

where y_{v_i} denotes the class label of node v_i. A higher value of $\mathcal{H}$ generally indicates stronger homophily of the graph.

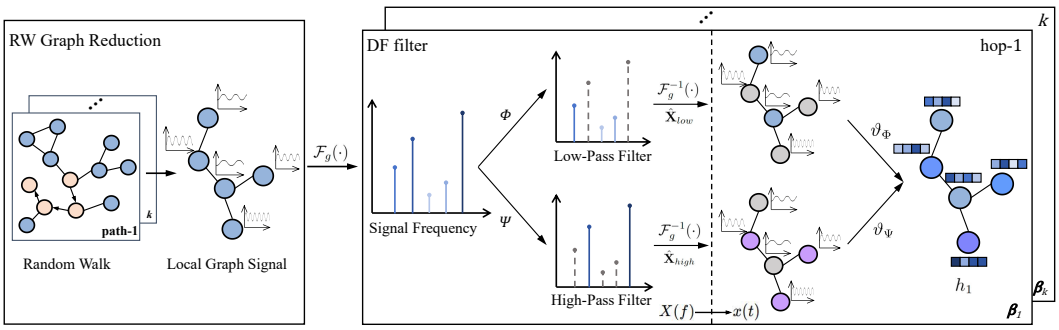

Fig. 1. Overall architecture of LDF-GNN. Left: random walk (RW) graph reduction samples local subgraphs to form local graph signals. Right: the dynamic fusion (DF) filter applies low-pass (Φ) and high-pass (Ψ) filtering and fuses them with $\vartheta_{\Phi k}$ and $\vartheta_{\Psi k}$ at hop k to obtain h_k. The hop-wise outputs are aggregated with β_k to produce the final embedding $\mathbf{Z}$.

IV. Methodology

In this section, we propose LDF-GNN to alleviate the structural misguidance problem by coupling graph reduction and dynamic fusion filter. Specifically, we first sample a local graph signal from the input graph using random-walk graph reduction. We then fuse high-pass filters and low-pass filters to dynamically aggregate low- and high-frequency information in the feature aggregation process of each hop. Finally, we explain the dynamic interaction of two components and their dynamic properties, describe parameter propagation during training, and present a theoretical analysis of the filter.

A. Localized random walk graph reduction

To alleviate structural noise while enabling subsequent filters to dynamically learn graph information, we use a graph reduction method to remove certain nodes and their connections to generate local graphs.

We adopt random walk–based sampling rather than uniform sampling, as it better preserves local structural information. Starting from randomly selected seed nodes, the graph is explored according to the transition probability matrix [28]. The probability of moving from node v_i to v_j is as follows:

$$\mathbf{P}(v_{t+1} = v_j \mid v_t = v_i) = \mathbf{P}_{ij} \tag{4}$$

where $\mathbf{P} \in \mathbb{R}^{n\times n}$ is computed by dividing the adjacency matrix $\mathbf{A}$ by the degree matrix $\mathbf{D}$:

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A} \tag{5}$$

where $\sum_{j=1}^{N} \mathbf{P}_{ij} = 1$. To obtain a random walk sequence $\{v_0, v_1, \ldots, v_T\}$ of T steps, we represent the process using the transition probability matrix as follows:

$$\pi^{(t)} = \pi^{(0)}\mathbf{P}^t \tag{6}$$

where $\pi^{(0)}$ is the initial node distribution and $\pi^{(t)}$ is the node distribution after t steps of random walk. We define the visit frequency vector $f = \sum_{t=0}^{T} \pi^{(t)}$ and set the threshold τ based on f as follows:

$$\tau = Q_p(f) \tag{7}$$

where $Q_p(f)$ represents the value of f at the $p-th$ quantile and p is a hyperparameter manually adjusted for different graphs. Nodes with higher visit frequencies are retained via

$$s_i = \begin{cases} 1, & \text{if } f_i \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where f_i is the visit frequency of node v_i and s_i indicates whether node v_i is retained. We then generate the mask r from the matrix s, and sample the feature matrix $\mathbf{X}$ and the adjacency matrix $\tilde{\mathbf{A}}$ to obtain $\mathbf{X}'$ and $\tilde{\mathbf{A}}'$ of the local graph:

$$\{\mathbf{X}', \tilde{\mathbf{A}}'\} = \mathcal{S}_r(\mathbf{X}, \tilde{\mathbf{A}}) \tag{9}$$

where $\mathcal{S}_r$ denotes the sampling operation based on mask r.

Although random walk explores local areas of the graph, multiple transitions can still capture widely global structural information, helping balance global information and local details. By learning on these local graph signals, the model can reduce the impact of noise in the global structure. In addition, this method reduces the aggregation frequency of adjacent nodes, which maintains the distinction between nodes. It prevents node features from becoming consistent due to multi-layer aggregation, thereby mitigating over-smoothing.

B. Frequency-aware feature fusion

Many real-world graphs exhibit strong heterophily, where misleading structural distributions can result in the propagation of incorrect information and misguide GNNs. To solve this, we explore the frequency domain of information and propose a frequency-aware feature fusion method. Figure 1 shows the filtering mechanism in our proposed model. By integrating high- and low-frequency information, it can adapt to both homophilic and heterophilic structures in the graph data.

In frequency-aware feature fusion, the low-pass filter smooths node features and emphasizes the similarity among homophilic nodes. By suppressing high-frequency noise in the spectrum, it exhibits strong capabilities in local information extraction and noise resistance. Its definition is as follows:

$$s_{\text{low}}^{(k)} = \mathbf{U}\Phi_\gamma(\omega)\mathbf{U}^T\mathbf{X} \tag{10}$$

where $s_{\text{low}}^{(k)}$ is the output signal of the k-hop low-pass filter, $\mathbf{U}$ is the eigenvector matrix of the Laplace matrix, ω is the eigenvalue matrix and $\Phi_\gamma(\omega) = \exp(-\gamma\omega^k)$ is the frequency

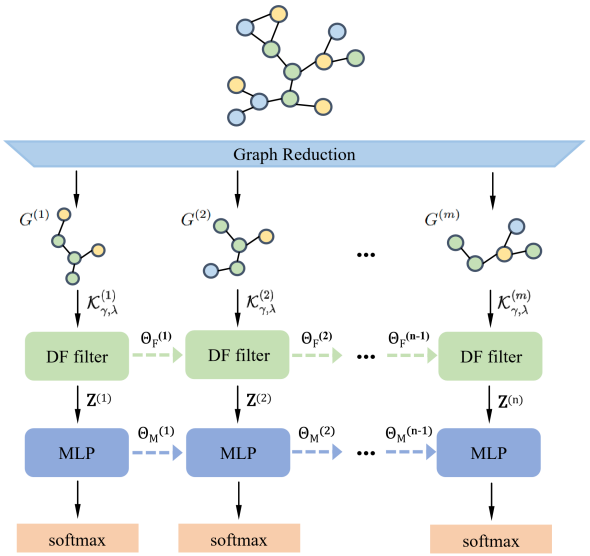

Fig. 2. Iteration process. In each iteration, the DF filter learns different local graph features and passes the parameters to the next iteration.

response function of the low-pass filter. The parameter γ controls the attenuation rate of high-frequency signals, and $\mathbf{X} \in \mathbb{R}^{n \times n}$ denotes the graph feature matrix.

The high-pass filter emphasizes high-frequency information to capture node differences and alleviate misleading distributions in the graph structure. Its definition is as follows:

$$\mathbf{s}_{\text{high}}^{(k)} = \mathbf{U}\Psi_{\lambda}(\omega)\mathbf{U}^{T}\mathbf{X} \tag{11}$$

where $\mathbf{s}_{\text{high}}^{(k)}$ is the output signal of the k-hop high-pass filter, $\Psi_{\lambda}(\omega) = \mathbf{I} - \exp(-\lambda\omega^{k})$ is the frequency response function of the high-pass filter, and λ controls the preservation strength of high-frequency signals. The theory of low- and high-pass filter design is analyzed in detail in Section IV-D.

For $k = \{k_1, k_2, \ldots, k_M\}$, the low- and high-pass filters perform k-hop propagation respectively. We perform a weighted summation at each propagation step. This allows for adaptive fusion of the filters during each hop. The fusion process is defined as follows:

$$h_k = \mathcal{C}\left\{s_{\text{low}}^{(k)}, s_{\text{high}}^{(k)}\right\} = \vartheta_{\Phi k} s_{\text{low}}^{(k)} + \vartheta_{\Psi k} s_{\text{high}}^{(k)} \tag{12}$$

where h_k denotes the fused feature embedding at the k-th hop, $\mathcal{C}$ denotes the fusion operation of the filtered signal, and the learnable parameters $\vartheta_{\Phi k}$ and $\vartheta_{\Psi k}$ denote the weight of low- and high-frequency signals of the k-th hop. We then aggregate the outputs across hops as follows:

$$\mathbf{Z} = \sum_{k=1}^{K} \beta_k h_k \tag{13}$$

where $\mathbf{Z}$ denotes the filtered output embedding, and the learnable parameter β_k denotes the aggregation weight of the k-th hop. The above two-step fusion enables the model to dynamically adjust the proportion of high- and low-frequency signals in each order of propagation to achieve accurate learning of both local and global graph signals while enhancing the robustness to structural noise and abnormal connections.

C. LDF-GNN

This section provides an overview of the LDF-GNN model.

Applying filters to local graphs. We name the filter model proposed in Section IV-B as the DF filter. As shown in Figure 2 , in each iteration, the model first performs graph reduction on the global graph G to generate a new local graph signal $G^{(m)}$, with m denoting the number of iterations. Then we apply the DF filter to these local graph signals. The kernel $\mathcal{K}_{\gamma,\lambda}^{(m)}$ of the filtering operation can be obtained from the local graphs:

$$\mathcal{K}_{\gamma,\lambda}^{(m)} = \left\{\mathbf{U}\Phi_{\gamma}^{(m)}(\omega)\mathbf{U}^{T}, \mathbf{U}\Psi_{\lambda}^{(m)}(\omega)\mathbf{U}^{T}\right\} \tag{14}$$

The low- and high-frequency signals in the local graph are fused in the filter. The output embedding Z is fed into an MLP for classification, and the final prediction is obtained through a softmax function, as follows:

$$\tilde{\mathbf{Z}} = \text{MLP}(\mathbf{Z}) = \text{MLP}\big(\mathcal{C}(\mathcal{K}_{\gamma,\lambda} \cdot \mathbf{X})\big) \tag{15}$$

$$\tilde{\mathbf{Y}}_i = \text{softmax}(\tilde{\mathbf{Z}}) \tag{16}$$

We use cross-entropy loss function in backpropagation.

Parameter passing and sharing. Parameter passing is indicated by dashed arrows in Figure 2. The weight parameters $\vartheta_{\Phi k}$, $\vartheta_{\Psi k}$, and β_k of the DF filter are dynamically adjusted according to the frequency characteristics of the current local graph signal. The local graph learning parameter sets Θ_{F} and Θ_{M} from the previous iteration are passed to the next iteration of local graph learning, where Θ_{F} denotes the parameter set of the DF filter and Θ_{M} denotes the MLP parameter set. This enables parameter sharing during training and ensures effective learning of the filter across different local graph signals.

Dynamic property. The dynamic property of LDF-GNN is reflected in two aspects. One is the dynamic change of the local graph signal. Since the local graph in each iteration is random, the filter learns different structural features in each update. This enhances the model's adaptability to different local structures in a graph, enabling the model to establish a balance between global and local information, thus reducing the impact of structural noise on model performance. The second is the dynamic adjustment of the filter. The model introduces learnable parameters in the fusion filter so that it can be dynamically adjusted according to the homophilic level in different regions. For regions with strong homophily, the model attenuates the high-frequency noise to highlight the low-frequency signal. And for regions with strong heterophily, the model enhances the propagation of high-frequency signals. This property not only mitigates over-smoothing in graph learning, but also avoids structural distribution misguidance.

D. Theoretical analysis

For an undirected graph $G = (V, E)$, let its adjacency matrix be $\mathbf{A}$ and its degree matrix be $\mathbf{D}$. The normalized graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, which is a symmetric semi-positive definite matrix with a set of orthogonal eigenvectors and non-negative eigenvalues. Its spectral decomposition is $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{T}$ where $\mathbf{U} \in \mathbb{R}^{N \times N}$ is the eigenvector matrix, and the column vectors are the

eigenvectors of $\mathbf{L}$, which form the frequency domain basis. $\Lambda \in \mathbb{R}^{N\times N}$ is the diagonal matrix whose elements are the eigenvalues, which denote the frequencies of the graph signal. A graph Fourier transform of the eigenmatrix $\mathbf{X}$ maps the graph signal from the time domain to the frequency domain:

$$\hat{\mathbf{X}} = \mathcal{F}_g(\mathbf{X}) = \mathbf{U}^T\mathbf{X} \tag{17}$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{n\times F}$ denotes the feature matrix in the frequency domain and $\mathcal{F}_g(\cdot)$ denotes graph Fourier transform. The Fourier inverse transform restores a frequency domain signal to a time domain signal:

$$\mathbf{X} = \mathcal{F}_g^{-1}(\hat{\mathbf{X}}) = \mathbf{U}\hat{\mathbf{X}} \tag{18}$$

where $\mathcal{F}_g^{-1}(\cdot)$ denotes graph Fourier inverse transform.

Based on the above conversion of the graph signal between the time and frequency domains, we design a filter model for graphs with reference to the Gaussian filter in conventional signal processing. To be consistent with the notation of the conventional filtering formula, the frequency of the graph signal Λ is denoted by ω in the following. In the frequency domain, the filter can be realized by element-wise weighting of the frequency components [15]. Since the Fourier transform decomposes the graph signal into different frequency components, element-wise multiplication is equivalent to weighting each frequency component individually. Then the low-pass filter in the frequency domain can be expressed as:

$$\hat{\mathbf{X}}_{\text{low}} = \Phi(\omega)\hat{\mathbf{X}} \tag{19}$$

where $\Phi(\omega)$ is the frequency response function. From Equation 17 and 18, the graph signal is transformed to the frequency domain, and the weighted signal is restored to the time domain. Then Equation 19 is transformed to the time domain space:

$$\mathbf{X}_{\text{low}} = \mathbf{U}\Phi(\omega)\mathbf{U}^T\mathbf{X} \tag{20}$$

Similarly, the high-pass filter in the time domain is:

$$\mathbf{X}_{\text{high}} = \mathbf{U}\Psi(\omega)\mathbf{U}^T\mathbf{X} \tag{21}$$

where $\Psi(\omega)$ is the frequency response function. The next question is how to represent the two frequency response functions. We refer to the Gaussian filter $G(\omega) = \exp\left(-\frac{\omega^2}{2\sigma^2}\right)$. The goal of the low-pass filter is to suppress high-frequency signals and retain low-frequency components to enhance the propagation of information between homophilic nodes. We set the low-pass frequency response function as:

$$\Phi_\gamma(\omega) = \exp(-\gamma\omega^k) \tag{22}$$

where k denotes the hop count of the filter and γ denotes the smoothing coefficient, which controls the attenuation rate of high-frequency signals. When the signal frequency is low and ω is small, $\Phi_\gamma(\omega) \to 1$, and the low-frequency signal has almost no attenuation. When the signal frequency is high and ω is large, $\Phi_\gamma(\omega)$ remains relatively low and the high-frequency signal is attenuated.

TABLE I
DATA STATISTICS.

Statistic	Nodes	Edges	Features	Classes	$\mathcal{H}$
Cora	2708	5278	1433	7	0.83
Citeseer	3327	4552	3703	6	0.71
Pubmed	19717	44324	500	3	0.80
Cornell	183	277	1703	5	0.30
Squirrel	5201	198353	2089	5	0.22
Texas	183	279	1703	5	0.07
Chameleon	2277	31371	2325	5	0.25

Since the high-pass filter highlights the heterophilic level of node features, we set the high-pass frequency response function as:

$$\Psi_\lambda(\omega) = \mathbf{I} - \exp(-\lambda\omega^k) \tag{23}$$

where λ denotes the high-frequency enhancement coefficient, which is used to adjust the degree of retention of high-frequency signals. In contrast to the low-pass frequency response function, for low-frequency signals, $\Psi_\lambda(\omega)$ tends to be small, which suppresses low-frequency signals. And high-frequency signals are amplified as $\Psi_\lambda(\omega)$ approaches 1.

The graph in the time domain represents the k-hop neighborhood information of a node in terms of the k-th power $\mathbf{A}^k$ of the adjacency matrix [29]. In the frequency domain, the k-th power of the Laplace matrix L is used to represent the k-hop neighborhood information of a node. The power operation of the Laplace matrix proceeds as follows:

$$\mathbf{L}^k = (\mathbf{U}\Lambda\mathbf{U}^T)^k = \prod_{i=1}^{k}(\mathbf{U}\Lambda\mathbf{U}^T) \tag{24}$$

Since $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, Equation 24 can be simplified as $\mathbf{L}^k = \mathbf{U}\Lambda^k\mathbf{U}^T$. So k-hop neighborhood information aggregation can be expressed as the k-th power of the graph signal frequency component ω in the frequency domain. So Equation 22 and Equation 23 express the layer-by-layer propagation of different frequencies by ω^k.

V. EXPERIMENTS

In this section, we conduct experiments on real-world datasets to evaluate the performance of the proposed LDF-GNN model in addressing the structural misguidance problem. We compare our model with other GNN models in both fully-supervised and semi-supervised learning tasks and perform hyperparameter analysis and ablation studies.

A. Datasets

We evaluate the performance of LDF-GNN on seven widely used datasets, including three citation graphs (Cora, Citeseer, and Pubmed [30]), two WebKB graphs (Texas and Cornell [19]), and two Wikipedia graphs (Squirrel and Chameleon [24]). The citation graphs consist of sparse bag-of-words feature vectors for each document and a list of citation links between them, where documents are treated as nodes and citation links as edges. The WebKB graphs are webpage databases from Carnegie Mellon University. In Wikipedia graphs, nodes represent webpages, node features correspond to several informative nouns extracted from Wikipedia pages

TABLE II
MEAN ACCURACY RESULTS FOR FULL-SUPERVISED NODE CLASSIFICATION

Datasets	Cora	Citeseer	Pubmed	Cornell	Squirrel	Texas	Chameleon
MLP	$76.91_{\pm 0.78}$	$76.45_{\pm 0.75}$	$86.21_{\pm 0.24}$	$84.18_{\pm 2.52}$	$30.94_{\pm 1.09}$	$86.72_{\pm 2.75}$	$46.59_{\pm 1.38}$
ChebNet	$87.28_{\pm 0.65}$	$79.39_{\pm 0.49}$	$87.79_{\pm 0.20}$	$83.97_{\pm 1.98}$	$40.76_{\pm 0.60}$	$86.24_{\pm 2.53}$	$59.53_{\pm 1.17}$
GCN	$87.14_{\pm 0.90}$	$79.87_{\pm 0.61}$	$86.75_{\pm 0.27}$	$65.75_{\pm 4.25}$	$45.86_{\pm 0.69}$	$77.04_{\pm 3.79}$	$59.92_{\pm 1.85}$
GCNII	$88.39_{\pm 0.61}$	$79.80_{\pm 0.50}$	$88.92_{\pm 0.32}$	$84.27_{\pm 2.48}$	$41.95_{\pm 1.36}$	$80.52_{\pm 2.86}$	$63.45_{\pm 1.02}$
ARMA	$87.16_{\pm 0.76}$	$80.08_{\pm 0.57}$	$86.97_{\pm 0.19}$	$85.56_{\pm 2.31}$	$37.23_{\pm 0.42}$	$83.96_{\pm 3.35}$	$60.21_{\pm 1.09}$
APPNP	$88.10_{\pm 0.53}$	$80.54_{\pm 0.63}$	$88.14_{\pm 0.28}$	$91.43_{\pm 1.76}$	$35.67_{\pm 0.53}$	$90.63_{\pm 1.67}$	$52.15_{\pm 1.24}$
GPR-GNN	$88.51_{\pm 0.58}$	$80.10_{\pm 0.74}$	$88.38_{\pm 0.25}$	$91.56_{\pm 1.37}$	$50.39_{\pm 0.71}$	$92.89_{\pm 1.14}$	$67.48_{\pm 1.41}$
BernNet	$88.53_{\pm 0.82}$	$80.05_{\pm 0.79}$	$88.49_{\pm 0.34}$	$92.11_{\pm 1.63}$	$51.32_{\pm 0.75}$	$92.67_{\pm 1.30}$	$68.53_{\pm 1.89}$
ChebNetII	$88.76_{\pm 0.68}$	$80.58_{\pm 0.78}$	$88.91_{\pm 0.26}$	$92.36_{\pm 1.27}$	$56.83_{\pm 0.34}$	$93.27_{\pm 1.60}$	$71.39_{\pm 1.02}$
EvenNet	$87.24_{\pm 1.29}$	$78.69_{\pm 0.23}$	$89.58_{\pm 0.25}$	$92.15_{\pm 2.11}$	$50.36_{\pm 0.59}$	$93.75_{\pm 1.59}$	$67.41_{\pm 1.71}$
PCNet	$\underline{90.06}_{\pm 0.49}$	$81.73_{\pm 0.62}$	$\underline{91.32}_{\pm 0.31}$	$\underline{93.79}_{\pm 1.70}$	$\underline{64.41}_{\pm 0.40}$	$93.69_{\pm 1.47}$	$\underline{72.18}_{\pm 1.56}$
JacobiConv	$88.95_{\pm 0.53}$	$80.74_{\pm 0.71}$	$89.65_{\pm 0.39}$	$92.65_{\pm 2.28}$	$61.74_{\pm 0.63}$	$93.47_{\pm 1.95}$	$71.84_{\pm 1.79}$
VR-GNN	$88.23_{\pm 0.61}$	$\underline{81.93}_{\pm 0.46}$	$89.67_{\pm 0.36}$	$92.67_{\pm 3.52}$	$57.51_{\pm 1.07}$	$\underline{94.72}_{\pm 1.21}$	$72.05_{\pm 1.80}$
LDF-GNN	$\mathbf{90.15}_{\pm \mathbf{0.92}}$	$\mathbf{82.75}_{\pm \mathbf{1.04}}$	$\mathbf{91.62}_{\pm \mathbf{0.27}}$	$\mathbf{94.11}_{\pm \mathbf{1.29}}$	$\mathbf{68.26}_{\pm \mathbf{1.44}}$	$\mathbf{94.77}_{\pm \mathbf{2.00}}$	$\mathbf{76.28}_{\pm \mathbf{1.52}}$

and edges denote mutual links between pages. These datasets are chosen to provide a diverse evaluation of LDF-GNN's performance on both homophilic and heterophilic graphs. Dataset statistics are summarized in Table I.

B. Experimental setup

Baselines. To evaluate the performance of LDF-GNN, we compare it with a series of state-of-the-art spectral GNNs on both full-supervised and semi-supervised node classification tasks. In full-supervised learning, we use ChebNet [9], GCN [10], GCNII [31], ARMA [11], APPNP [32], GPR-GNN [13], BernNet [14], ChebNetII [16], EvenNet [15], PCNet [18], JacobiConv [17] and VR-GNN [33] as baseline models. And in semi-supervised learning, we use ChebNet [9], GCN [10], ARMA [11], APPNP [32], GPR-GNN [13], BernNet [14], ChebNetII [16], EvenNet [15] and PCNet [18] as baselines. All baseline models are implemented with their suggested settings.

Setting. In full-supervised learning, we use all datasets mentioned in Section V-A and randomly divide the nodes of each class into 60%, 20%, and 20% for training, validation, and testing sets. For semi-supervised learning, we use Cora, Citeseer, Pubmed, Cornell, and Texas datasets. We split the three citation graphs (Cora, Citeseer, and Pubmed), into 20 nodes per category for training, 500 for validation, and 1000 for testing. For Cornell and Texas, we randomly divided each class of nodes into 2.5% for training, 2.5% for validation, and 95% for testing. In the training process, we apply an MLP after the DF filter and set the number of layers to 2 with 512 hidden units per layer. We select nodes using the quantile $p \in [0.25, 0.7]$, and run the DF filter for $K \in [3, 7]$ hops.

C. Full-supervised node classification

We conduct full-supervised node classification experiments on seven real-world datasets and compare our model with several state-of-the-art methods. Table II reports the mean classification accuracy and standard deviation of these models over ten run times. We use **bold** for the best and underlining for the second best on each dataset. We observe that LDF-GNN outperforms the baseline models on all datasets.

For the homophilic datasets (Cora, Citeseer, and Pubmed), LDF-GNN achieves improvements over the baseline by 0.09%, 0.82%, and 0.30%, respectively. This indicates that the model effectively extracts and learns the local information from graph signals, adapts to regional differences, and thus enhances its performance on homophilic graphs.

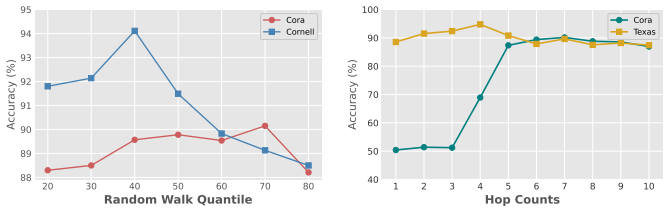

Fig. 3. Sensitivity analysis of RW quantile and hop counts

Our model also performs competitively on heterophilic graphs. For example, it outperforms BernNet by 2.00% on Cornell, EvenNet by 1.02% on Texas, and 4.89% by ChebNetII on Chameleon. Notably, the average accuracy on Squirrel is 68.26%, outperforming PCNet by 3.85%, which corresponds to a performance improvement of 5.9%. These results show that LDF-GNN not only mitigates the local misguidance problem in homophilic graphs but also adaptively preserves discrepancies among nodes due to its fusion method across different frequency bands.

D. Semi-supervised node classification

To further evaluate the performance of LDF-GNN in semi-supervised tasks, we conducted experiments on five datasets. Table III reports the mean classification accuracy and standard deviation of semi-supervised node classification for each model on these datasets. On the three homophilic graphs, our model achieves state-of-the-art performance with accuracies of 83.42%, 73.57%, and 80.87%, respectively. Additionally, on Cornell and Texas, LDF-GNN outperforms all baseline models too, surpassing the best-performing PCNet by 3.92% and 1.97%, respectively. The experimental results demonstrate that LDF-GNN maintains competitive performance even in semi-supervised learning by its strong dynamic adaptability.

E. Parameter sensitivity analysis

We analyze the random walk quantile p and the hop count K in Figure 3, where RW denotes random walk. The precision varies with p and peaks in the middle around 0.7 on Cora and 0.4 on Cornell. And extremes degrade performance by making

TABLE III
MEAN ACCURACY RESULTS FOR SEMI-SUPERVISED NODE CLASSIFICATION

Datasets	Cora	Citeseer	Pubmed	Cornell	Texas
MLP	$58.03_{\pm 0.31}$	$56.87_{\pm 0.53}$	$71.74_{\pm 0.26}$	$40.36_{\pm 4.61}$	$39.72_{\pm 2.68}$
ChebNet	$79.41_{\pm 0.33}$	$69.11_{\pm 0.71}$	$75.73_{\pm 0.47}$	$24.78_{\pm 3.54}$	$32.45_{\pm 3.13}$
GCN	$81.25_{\pm 0.46}$	$71.84_{\pm 0.52}$	$79.18_{\pm 0.13}$	$34.96_{\pm 4.31}$	$34.42_{\pm 2.83}$
ARMA	$82.18_{\pm 0.37}$	$71.25_{\pm 0.19}$	$76.42_{\pm 0.24}$	$30.17_{\pm 5.72}$	$49.76_{\pm 3.72}$
APPNP	$83.19_{\pm 0.42}$	$72.84_{\pm 0.13}$	$79.46_{\pm 0.37}$	$42.73_{\pm 5.39}$	$48.25_{\pm 3.59}$
GPR-GNN	$83.74_{\pm 0.35}$	$69.58_{\pm 0.43}$	$76.21_{\pm 0.83}$	$40.74_{\pm 4.22}$	$45.93_{\pm 3.26}$
BernNet	$81.53_{\pm 0.37}$	$72.68_{\pm 0.27}$	$79.42_{\pm 0.35}$	$43.64_{\pm 5.39}$	$46.20_{\pm 2.74}$
ChebNetII	$83.04_{\pm 0.32}$	$71.32_{\pm 0.21}$	$79.98_{\pm 0.34}$	$46.53_{\pm 5.35}$	$52.63_{\pm 4.85}$
EvenNet	$81.61_{\pm 0.30}$	$72.09_{\pm 0.64}$	$79.35_{\pm 0.27}$	$51.62_{\pm 4.12}$	$53.18_{\pm 4.25}$
PCNet	$82.81_{\pm 0.33}$	$70.04_{\pm 0.49}$	$80.53_{\pm 0.17}$	$51.89_{\pm 2.28}$	$63.12_{\pm 3.74}$
LDF-GNN	$\mathbf{83.42_{\pm 0.22}}$	$\mathbf{73.57_{\pm 0.45}}$	$\mathbf{80.87_{\pm 0.21}}$	$\mathbf{55.81_{\pm 3.98}}$	$\mathbf{65.09_{\pm 3.16}}$

TABLE IV
ABLATION STUDY RESULTS

Datasets	Cora	Citeseer	Pubmed	Cornell	Squirrel	Texas	Chameleon
DF filter	$88.87_{\pm 0.73}$	$80.96_{\pm 0.61}$	$90.74_{\pm 0.28}$	$91.85_{\pm 1.45}$	$68.21_{\pm 1.26}$	$93.11_{\pm 3.93}$	$74.91_{\pm 1.30}$
LDF-GNN$_{\text{low}}$	$87.48_{\pm 0.69}$	$80.10_{\pm 0.89}$	$86.41_{\pm 0.15}$	$74.59_{\pm 8.80}$	$42.82_{\pm 1.88}$	$64.54_{\pm 7.85}$	$64.38_{\pm 2.27}$
LDF-GNN$_{\text{high}}$	$74.12_{\pm 1.34}$	$60.93_{\pm 1.73}$	$82.56_{\pm 0.42}$	$89.34_{\pm 3.37}$	$64.07_{\pm 1.12}$	$89.50_{\pm 2.95}$	$75.09_{\pm 1.38}$
LDF-GNN	$\mathbf{90.15_{\pm 0.92}}$	$\mathbf{82.75_{\pm 1.04}}$	$\mathbf{91.62_{\pm 0.27}}$	$\mathbf{94.11_{\pm 1.29}}$	$\mathbf{68.26_{\pm 1.44}}$	$\mathbf{94.77_{\pm 2.00}}$	$\mathbf{76.28_{\pm 1.52}}$

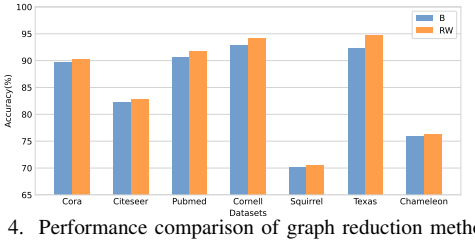

Fig. 4. Performance comparison of graph reduction methods.

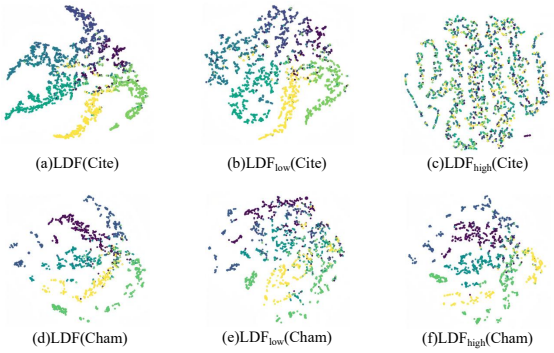

Fig. 5. Visualization of ablation study results.

local views too sparse or too dense. This result indicates that the DF filter learns more effectively on specific local signals than on global signals. Increasing K brings early gains and then little decline. Cora plateaus near 6–7 hops with a slight drop afterward, whereas Texas prefers shallower propagation of about 3–5 hops. The trend indicates that the model can capture the discrepancies of features among nodes in a smaller propagation range.

F. Comparison of graph reduction methods

To evaluate the advantage of random walk in the graph reduction process, we compare it with an alternative method that relies on a Bernoulli distribution, where each node in the graph is randomly dropped with a completely random probability. Figure 4 shows the classification results of using the two graph reduction methods separately on each dataset, where RW denotes random walk-based reduction and B denotes random reduction with the Bernoulli distribution. All random walk-based methods outperform the unstructured graph reduction methods. The experimental results demonstrate that using structure-aware approaches for localization during graph reduction can preserve structural information, allowing the filter to better adapt to both global and local graph structures. Therefore, random walk is more advantageous than the unstructured downsampling method in mitigating the problem of structure misguidance.

G. Ablation study of two dynamic properties

We evaluate LDF-GNN's two dynamic properties, with results and t-SNE views shown in Table IV and Figure 5.

Dynamic inputs. We first conduct experiments on the DF filter without the graph reduction process. On the three citation graphs, the improvements are around 1% - 2%. And we also see benefits on Chameleon and Texas. These results suggest that dynamic inputs of the graph filter helps extract reliable local information and mitigates structural noise.

Dynamic fusion. In Table IV, LDF-GNN$_{\text{low}}$ denotes the model using only low-pass filters and LDF-GNN$_{\text{high}}$ denotes the model using only high-pass filters. On homophilic citation graphs, we obtain better results with the low-pass filter than the high-pass filter, while on Chameleon and Texas the high-pass filter surpasses the low-pass filter. When we fuse the two filters, we achieve the best performance in all cases, which demonstrates that dynamically adjusting the weights of low- and high-frequency components is necessary. Figure 5 shows that with only the low-pass filter, the class boundaries on Citeseer are fuzzy, and with only the high-pass filter, the categories are almost impossible to observe. However, with the fused model, we achieve clearer and more separable clusters on both datasets. These observations align with the results in Table IV and explain the advantage of LDF-GNN.

VI. CONCLUSION AND FUTURE WORK

In this paper, we illustrate the structural misguidance problem in graph learning caused by local discrepancies and het-

erophilic connections, and propose LDF-GNN to mitigate it. By combining random walk graph reduction with frequency-aware feature fusion, our model adapts filter responses to local signal characteristics and accommodates both homophilic and heterophilic relations across regions. Experiments on multiple benchmarks show state-of-the-art performance, and ablation studies verify the importance of the two dynamic properties in mitigating the structural misguidance problem. Future work will extend LDF-GNN to broader tasks and explore more efficient or adaptive sampling strategies and filtering mechanism.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62502084; in part by Shanghai Sailing Program under Grant 24YF2701600; and in part by the Fundamental Research Funds for the Central Universities under Grant 2232025D-46.

REFERENCES

[1] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar, "A survey of graph neural networks for social recommender systems," *ACM Computing Surveys*, vol. 56, no. 10, pp. 1–34, 2024.

[2] Hao Chen, Yuanchen Bei, Qijie Shen, Yue Xu, Sheng Zhou, Wenbing Huang, Feiran Huang, Senzhang Wang, and Xiao Huang, "Macro graph neural networks for online billion-scale recommender systems," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3598–3608.

[3] Hakan T Otal, Abdulhamit Subasi, Furkan Kurt, M Abdullah Canbaz, and Yasin Uzun, "Analysis of gene regulatory networks from gene expression using graph neural networks," in *Digital Healthcare, Digital Transformation and Citizen Empowerment in Asia-Pacific and Europe for a Healthier Society*, pp. 249–270. Elsevier, 2025.

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

[5] Yonghua Zhu, Lei Feng, Zhenyun Deng, Yang Chen, Robert Amor, and Michael Witbrock, "Robust node classification on graph data with graph and label noise," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 17220–17227.

[6] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin, "Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[7] Peng Mei and Yu Hong Zhao, "Dynamic network link prediction with node representation learning from graph convolutional networks," *Scientific Reports*, vol. 14, no. 1, pp. 538, 2024.

[8] Xinyi Gao, Junliang Yu, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin, "Graph condensation: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

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

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

[11] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi, "Graph neural networks with convolutional arma filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3496–3507, 2021.

[12] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

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

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

[15] Runlin Lei, Zhen Wang, Yaliang Li, Bolin Ding, and Zhewei Wei, "Evennet: Ignoring odd-hop neighbors improves robustness of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4694–4706, 2022.

[16] Mingguo He, Zhewei Wei, and Ji-Rong Wen, "Convolutional neural networks on graphs with chebyshev approximation, revisited," *Advances in neural information processing systems*, vol. 35, pp. 7264–7276, 2022.

[17] Xiyuan Wang and Muhan Zhang, "How powerful are spectral graph neural networks," in *International conference on machine learning*. PMLR, 2022, pp. 23341–23362.

[18] Bingheng Li, Erlin Pan, and Zhao Kang, "Pc-conv: Unifying homophily and heterophily with two-fold filtering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 13437–13445.

[19] Rui Duan, Mingjian Guang, Junli Wang, Chungang Yan, Hongda Qi, Wenkang Su, Can Tian, and Haoran Yang, "Unifying homophily and heterophily for spectral graph neural networks via triple filter ensembles," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[20] Qi Zhang, Jinghua Li, Yanfeng Sun, Shaofan Wang, Junbin Gao, and Baocai Yin, "Beyond low-pass filtering on large-scale graphs via adaptive filtering graph neural networks," *Neural Networks*, vol. 169, pp. 1–10, 2024.

[21] Han Yang, Junyuan Fang, Jiajing Wu, and Zibin Zheng, "Homophily-heterophily graph neural," *Blockchain Transaction Data Analytics: Complex Network Approaches*, p. 101, 2025.

[22] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[23] Kangkang Lu, Yanhua Yu, Hao Fei, Xuan Li, Zixuan Yang, Zirui Guo, Meiyu Liang, Mengran Yin, and Tat-Seng Chua, "Improving expressive power of spectral graph neural networks with eigenvalue correction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 14158–14166.

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

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

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

[27] Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova, "Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy," *arXiv preprint arXiv:2209.06177*, vol. 21, 2022.

[28] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, and Xiangjie Kong, "Random walks: A review of algorithms and applications," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 95–107, 2019.

[29] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang, "How powerful are k-hop message passing graph neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4776–4790, 2022.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[31] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li, "Simple and deep graph convolutional networks," in *International conference on machine learning*. PMLR, 2020, pp. 1725–1735.

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

[33] Fengzhao Shi, Yanan Cao, Ren Li, Xixun Lin, Yanmin Shang, Chuan Zhou, Jia Wu, and Shirui Pan, "Vr-gnn: variational relation vector graph neural network for modeling homophily and heterophily," *World Wide Web*, vol. 27, no. 3, pp. 32, 2024.