NODE-WISE FILTERING IN GRAPH NEURAL NETWORKS: A MIXTURE OF EXPERTS APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have proven to be highly effective for node classification tasks across diverse graph structural patterns. Most GNNs employ a uniform global filter—typically a low-pass filter for homophilic graphs and a high-pass filter for heterophilic graphs. However, real-world graphs often exhibit a complex mix of homophilic and heterophilic patterns, rendering a single global filter approach suboptimal. While few methods have introduced multiple global filters, they often apply these filters uniformly across all nodes, which may not effectively capture the diverse structural patterns present in real-world graphs. In this work, we theoretically demonstrate that a global filter optimized for one pattern can adversely affect performance on nodes with differing patterns. To address this, we introduce a novel GNN framework NODE-MOE that utilizes a mixture of experts to adaptively select the appropriate filters for different nodes. Extensive experiments demonstrate the effectiveness of NODE-MOE on both homophilic and heterophilic graphs.

1 INTRODUCTION

Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Veličković et al., 2017) have emerged as powerful tools in representation learning for graph structure data, and have achieved remarkable success on various graph learning tasks (Wu et al., 2020; Ma & Tang, 2021), especially the node classification task. GNNs usually can be designed and viewed from two domains, i.e., spatial domain and spectral domain. In the spatial domain, GNNs (Kipf & Welling, 2016; Hamilton et al., 2017; Gasteiger et al., 2018) typically follow the message passing mechanism (Gilmer et al., 2017), which propagate messages between neighboring nodes. In the spectral domain, GNNs (Defferrard et al., 2016; Chien et al., 2020) apply different filters on the graph signals in the spectral domain of the graph Laplacian matrix.

Most GNNs have shown great effectiveness in the node classification task of homophilic graphs (Veličković et al., 2017; Wu et al., 2019; Gasteiger et al., 2018; Baranwal et al., 2021), where connected nodes tend to share the same labels. These GNNs usually leverage the low-pass filters, where the smoothed signals are preserved. However, the heterophilic graphs exhibit the heterophilic patterns, where the connected nodes tend to have different labels. As a result, several GNNs (Sun et al., 2022; Li et al., 2024; Bo et al., 2021) designed for heterophilic graphs introduce the high-pass filter to better handle such diversity. To adapt to both homophilic and heterophilic graphs, GNNs with learnable graph convolution (Chien et al., 2020; Bianchi et al., 2021; He et al., 2021; 2022) can automatically learn different types of filters for different types of graphs. Despite the great success, these GNNs usually apply a uniform global filter across all nodes.

However, real-world graphs often display a complex interplay of homophilic and heterophilic patterns (Li et al., 2022; Luan et al., 2022; Mao et al., 2024), challenging this one-size-fits-all filtering approach. Specifically, while some nodes tend to connect with others that share similar labels, reflecting homophilic patterns, others are more inclined to form connections with nodes that have differing labels, indicative of heterophilic patterns. There are few methods, such as ACM-GNN (Luan et al., 2022), AutoGCN (Wu et al., 2022), PC-Conv (Li et al., 2024) and ASGAT (Li et al., 2021) leverage different filters to alleviate this issue. These methods, referred to as post-fusion methods, apply multiple filters to all nodes and subsequently combine the predictions of different filters. However, applying the same filters to all nodes can lead to potential issues. For example, applying a uniform type of filter, tailored for just one of these patterns, across all nodes may hurt the performance of

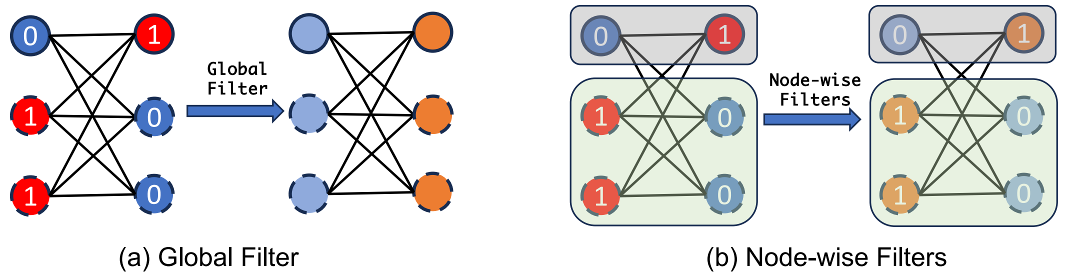


Figure 1: A toy example to illustrate the effect of global and node-wise filters. The node color represents features, and the number indicates the labels. Solid nodes represent nodes that follow homophilic patterns, whereas dotted circle nodes represent those with heterophilic patterns. For the solid-edge nodes, 2 out of their 3 neighbors have the same label, indicating homophilic patterns. Conversely, for the dashed-edge nodes, 2 out of their 3 neighbors have different labels, indicating heterophilic patterns.

other patterns. To illustrate this, we provide an example as shown in Figure 1(a), where different colors represent distinct node features, and numbers indicate node labels. The nodes are marked as either solid or dotted circles to denote homophilic and heterophilic patterns, respectively. Applying a global low-pass filter $1 - \lambda$, where λ is the eigenvalue of graph Laplacian matrix, uniformly across all nodes results in a scenario where nodes on the left possess the same feature, while those on the right possess another. Therefore, all the left nodes or the right nodes will have the same prediction. However, nodes on the left or right don't share the same label. Consequently, this global filtering approach leads to misclassification. Moreover, the indistinguishability of the filtered features can adversely impact post-fusion methods, such as those using attention mechanisms for combination.

This toy example clearly illustrates the limitations of a one-size-fits-all filtering strategy and motivates the need for a more tailored approach. To address this, Instead of applying one or multiple filters to all nodes, we propose a node-wise filtering method that apply different filters to different nodes based on their specific structural patterns. Figure 1(b) provides an example that we apply a low-pass filter, such as $1 - \lambda$, to homophilic nodes, and a high-pass filter, such as $\lambda - 1$, to heterophilic nodes. From the results, nodes in the same class would have the same features. Therefore, this node-wise filtering approach allows for the perfect classification of all nodes in this example.

Present work. In this work, we observe that nodes in many real-world graphs not only exhibit diverse structural patterns, but these patterns also vary significantly among different communities within the same graph. Utilizing the CSBM model to generate graphs with mixed structural patterns, we theoretically demonstrate that a global filter optimized for one pattern may incur significant losses for nodes with other patterns, while node-wise filtering can achieve linear separability for all nodes under mild conditions. Building on these insights, we propose a node-wise filtering method - NODE-MOE, which leverages a Mixture of Experts framework to adaptively select appropriate filters for different nodes. Extensive experiments validate the effectiveness of the proposed NODE-MOE on both homophilic and heterophilic graphs, as well as the explainability of the method.

2 PRELIMINARY

In this section, we explore the structural patterns present in various graph datasets, which usually exhibit mixed homophilic and heterophilic patterns. Then, we theoretically demonstrate that a global filter often fails in graphs characterized by such mixed structural patterns. In contrast, node-wise filtering can achieve linear separability under mild conditions. Before we start, we first define the notations used in this paper and background knowledge.

Notations. We use bold upper-case letters such as $\mathbf{X}$ to denote matrices. $\mathbf{X}_i$ denotes its i-th row and $\mathbf{X}_{ij}$ indicates the i-th row and j-th column element. We use bold lower-case letters such as $\mathbf{x}$ to denote vectors. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the node set, $\mathcal{E}$ is the edge set, and $|\mathcal{V}| = n$. $\mathcal{N}_i$ denotes the neighborhood node set for node v_i. The graph can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{ij} > 0$ indices that there exists an edge between nodes v_i and v_j in $\mathcal{G}$, or otherwise $\mathbf{A}_{ij} = 0$. For a node v_i, we use $\mathcal{N}(v_i) = \{v_j : \mathbf{A}_{ij} > 0\}$ to denote its neighbors. Let $\mathbf{D} = diag(d_1, d_2, \ldots, d_n)$ be the degree matrix, where $d_i = \sum_j \mathbf{A}_{ij}$ is the degree of node v_i. Furthermore, suppose that each node is associated with a d-dimensional feature $\mathbf{x}$ and we use $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ to denote the feature matrix. Besides, the label matrix is $\mathbf{Y} \in \mathbb{R}^{n \times c}$, where c is the number of classes. We use y_v to denote the label of node v.

Graph Laplacian. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. We define the normalized adjacency matrix as $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and the normalized Laplacian matrix as $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$. Its eigendecomposition can be represented by $\tilde{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where the $\mathbf{U} \in \mathbb{R}^{n \times n}$ is the eigenvector matrix and $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \dots, \lambda_n)$ is the eigenvalue matrix. Specifically, $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n < 2$. The filtered signals can be represented by $\hat{\mathbf{X}} = \mathbf{U}f(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{X}$, where f is the filter function. As a result, the graph convolution $\tilde{\mathbf{A}}\mathbf{X}$ can be viewed as a low-pass filter, with the filter $f(\lambda_i) = 1 - \lambda_i$. Similarly, the graph convolution $-\tilde{\mathbf{A}}\mathbf{X}$ is a high-pass filter with filter $f(\lambda_i) = \lambda_i - 1$.

Homophily metrics. Homophily metrics measure the tendency of edges to connect nodes with similar labels (Platonov et al., 2024). There are several commonly used homophily metrics, such as edge homophily (Zhu et al., 2020), node homophily (Pei et al., 2020), and class homophily (Lim et al., 2021b). In this paper, we adopt the node homophily $H(\mathcal{G}) = \frac{1}{|\mathcal{V}|}\sum_{v_i \in \mathcal{V}} h(v_i)$, where $h(v_i) = \frac{|\{u \in \mathcal{N}(v_i) : y_u = y_v\}|}{d_i}$ measures the label similarity between node v_i with its neighbors. A node with higher $h(v)$ exhibits a homophilic pattern while a low $h(v)$ indicates a heterophilic pattern.

2.1 Structural Patterns in Existing Graphs

In this subsection, we examine the structural patterns present in existing graph datasets. Specifically, we select two widely used homophilic datasets, i.e., Cora and CiteSeer (Sen et al., 2008), and two heterophilic datasets, i.e., chameleon and squirrel (Rozemberczki et al., 2021). We first calculate the homophily distribution for all nodes in the graph. As shown in probability density function (PDF) Figure 2, while the majority of nodes in homophilic graphs predominantly exhibit homophilic patterns, and those in heterophilic graphs display heterophilic patterns, exceptions are evident. Notably, some nodes in homophilic graphs show heterophilic tendencies, and conversely, some nodes in heterophilic graphs demonstrate homophilic patterns. Consequently, **all these graphs exhibit a mixture of homophilic and heterophilic patterns**, which aligns with the findings in the previous works (Luan et al., 2022; Mao et al., 2024).

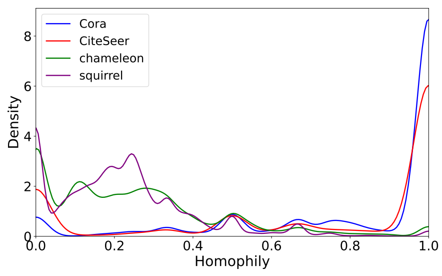


Figure 2: Node homophily ($h(v)$) density.

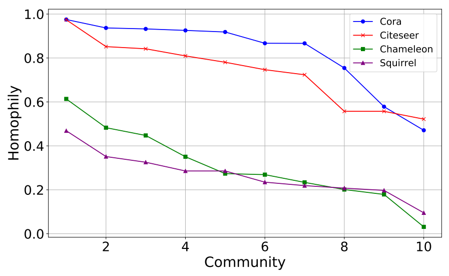


Figure 3: Homophily in different communities.

We further analyze the position of nodes with different structural patterns within the graphs. To do this, we divide each graph into several subgraphs using community detection algorithms (Fortunato, 2010). We focus on the largest 10 communities and calculate the homophily level for each subgraph. The results, as shown in Figure 3, reveal **significant variations in homophily across different communities**. For instance, in the Cora dataset, homophily levels in some communities approach 1, indicating strong homophily, while in some communities it drops below 0.5. Similarly, in the chameleon dataset, the lowest homophily levels are near 0, with the highest reaching above 0.6. These findings highlight the considerable diversity in node interaction patterns, even within the same graph, underscoring the complexity of graph structures in real-world datasets. The variability in homophily levels clearly illustrates that nodes in various parts of the graph may require distinct processing approaches. Therefore, applying the same global filter to all nodes may lead to suboptimal performance.

2.2 Analysis based on CSBM model

To further illustrate why applying a global filter may result in suboptimal performance, we utilize the Contextual Stochastic Block Model (CSBM) (Deshpande et al., 2018), which has been widely applied to graph analysis (Fortunato & Hric, 2016; Jiang et al., 2023), such as analyzing the behavior of GNNs (Palowitch et al., 2022; Baranwal et al., 2021; Ma et al., 2021). The CSBM is a generative model, which is often used to generate graph structures and node features. Typically, CSBMs are based on the assumption that graphs are generated following a uniform pattern, such as nodes with the same label are connected with probability p while nodes with different labels are connected with probability q (Ma et al., 2021). However, the real-world complexity of graphs features a mixture of homophilic and heterophilic patterns, as illustrated in section 2.1. We adapt the CSBM by mixing two CSBMs to generate one graph, following Mao et al. (2024).

Definition 1. *$CSBM(n, \boldsymbol{\mu}, \boldsymbol{\nu}, (p_0, q_0), (p_1, q_1), P)$. The generated nodes consist of two classes, $C_0 = \{i \in [n] : y_i = 0\}$ and $C_1 = \{j \in [n] : y_j = 1\}$. For each node, consider $\mathbf{X} \in \mathbb{R}^{n\times d}$ to be the feature matrix such that each row $\mathbf{X}_i$ is an independent d-dimensional Gaussian random vectors with $\mathbf{X}_i \sim N\left(\boldsymbol{\mu}, \frac{1}{d}\mathbf{I}\right)$ if $i \in C_0$ and $\mathbf{X}_j \sim N\left(\boldsymbol{\nu}, \frac{1}{d}\mathbf{I}\right)$ if $j \in C_1$. Here $\boldsymbol{\mu}, \boldsymbol{\nu}$ are the fixed class mean vectors with $\|\boldsymbol{\mu}\|_2, \|\boldsymbol{\nu}\|_2 \leq 1$ and $\mathbf{I}$ is the identity matrix. Suppose there are two patterns of nodes in the adjacency matrix $\mathbf{A} = (a_{ij})$, i.e., the homophilic pattern: $H_0 = \{i \in [n] : a_{ij} = \mathrm{Ber}(p_0)$ if $y_i = y_j$ and $a_{ij} = \mathrm{Ber}(q_0)$ if $y_i \neq y_j, p_0 > q_0\}$ and the heterophilic pattern: $H_1 = \{i \in [n] : a_{ij} = \mathrm{Ber}(p_1)$ if $y_i = y_j$ and $a_{ij} = \mathrm{Ber}(q_1)$ if $y_i \neq y_j, p_1 < q_1\}$. P denotes the probability that a node is in the homophilic pattern. We also assume the nodes follow the same degree distribution $p_0 + q_0 = p_1 + q_1$.*

For simplification, we consider a linear model with parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, following the approach (Baranwal et al., 2021). The predicted label for nodes is given by $\hat{\mathbf{y}} = \sigma(\tilde{\mathbf{X}}\mathbf{w} + b\mathbf{1})$, where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, and $\tilde{\mathbf{X}}$ represents the features after filtering. The binary cross-entropy loss over nodes $\mathcal{V}$ is formulated as $L(\mathcal{V}, \mathbf{w}, b) = -\frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$.

Theorem 1. *Suppose n is relatively large, the graph is not too sparse with $p_i, q_i = \omega(\log^2(n)/n)$ and the feature center distance is not too small with $\|\boldsymbol{\mu} - \boldsymbol{\nu}\| = \omega(\frac{\log n}{\sqrt{\mathrm{dn}(p_0+q_0)}})$ and $\|\mathbf{w}\| \leq R$. For the graph $G(\mathcal{V}, \mathcal{E}, \mathbf{X}) \sim CSBM(n, \boldsymbol{\mu}, \boldsymbol{\nu}, (p_0, q_0), (p_1, q_1), P)$, we have the following:*

1. If the low-pass global filter, i.e., $1 - \lambda$, is applied to the whole graph G, we can find a optimal $\mathbf{w}^*, b^*$ that achieve near linear separability for the homophilic node set H_0. However, the loss for the heterophilic node set H_1 can be relatively large with:

$$L(H_1, \mathbf{w}^*, b^*) \geq \frac{R(q_1 - p_1)}{2(q_1 + p_1)}\|\boldsymbol{\mu} - \boldsymbol{\nu}\|\left(1 + o_d(1)\right).$$

2. If different filters are applied to homophilic and heterophilic sets separately, we can find an optimal $\mathbf{w}^*, b^*$ that all the nodes are linear separable with the probability:

$$\mathbb{P}\left(\left(\tilde{\mathbf{X}}_i\right)_{i\in\mathcal{V}} \text{ is linearly separable}\right) = 1 - o_d(1).$$

3. Homophilic and heterophilic nodes can be separated based on the feature distance between a node and the average feature vector of its neighbors, given by $\|\mathbf{X}_i - \sum_{j\in\mathcal{N}(i)}\frac{\mathbf{X}_j}{D_{ii}}\|$ with probability $P = 1 - o_d(1)$.

The proof of these results is detailed in Appendix A. Theorem 1 reveals critical insights into the filtering strategies for graphs with mixed homophilic and heterophilic patterns, as generated by the CSBM model. The first part of the theorem illustrates that applying a global low-pass filter can create an optimal classifier for homophilic nodes, achieving near-linear separability. However, this classifier may result in a large loss value for heterophilic nodes, highlighting the limitations of a uniform filtering strategy. Conversely, the second part of the theorem demonstrates that by applying different filters to different patterns of nodes separately, it is possible to achieve linear separability across all nodes. These findings strongly motivate the exploration of a node-wise filtering method, which can automatically apply different filters to distinct nodes based on their specific patterns, to improve the overall performance.

3 The Proposed Method

The investigations presented in Section 2 underscore the complex nature of real-world datasets, revealing a mixture of homophilic and heterophilic patterns within them. Additionally, these patterns are not uniformly distributed throughout the graph; rather, the level of homophily varies significantly across different communities. Our theoretical analysis further demonstrates that global filtering, as commonly employed in numerous GNNs, may not effectively capture such complex patterns, often leading to suboptimal performance. In contrast, node-wise filtering, which applies distinct filters to individual nodes based on their specific patterns, shows great promise in handling the intricacies of such complex graphs.

However, implementing the node-wise filtering approach presents two significant challenges. First, how can we incorporate various filters into a single unified framework? It requires a flexible architecture that can seamlessly accommodate multiple filtering mechanisms without compromising the efficiency and scalability of the model. Second, without ground truth on node patterns, how can we select the appropriate filters for different nodes? In the following subsections, we aim to address these challenges.

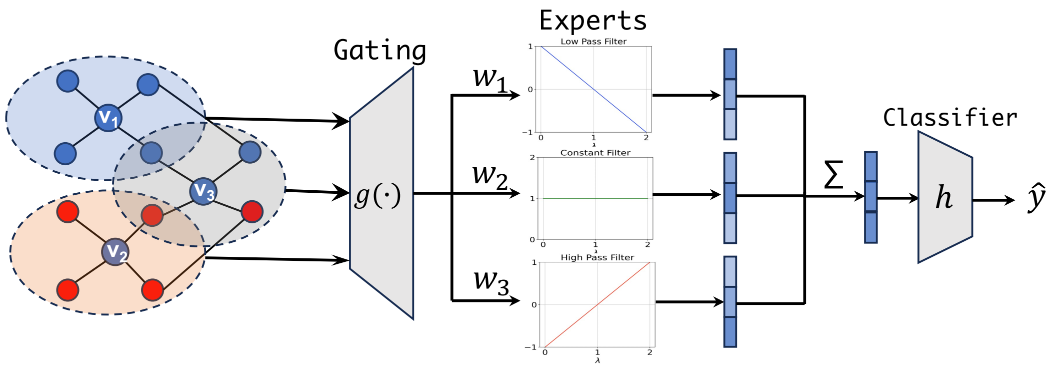


Figure 4: The overall framework of the proposed NODE-MOE. For each node, the gating model will assign different weights for each expert based on the node's feature and context. The experts can be any GNNs with different filters. The number of experts is also flexible.

3.1 NODE-MOE: NODE-WISE FILTERING VIA MIXTURE OF EXPERTS

Mixture of Experts (MoE) (Jacobs et al., 1991; Jordan & Jacobs, 1994), which follows the divide-and-conquer principle to divide the complex problem space into several subspaces so that each one can be easily addressed by specialized experts, have been successfully adopted across various domains (Masoudnia & Ebrahimpour, 2014; Shazeer et al., 2017; Riquelme et al., 2021). For node classification tasks in graphs exhibiting a mixture of structural patterns, the diversity of node interactions necessitates applying distinct filters to different nodes as we discussed in Sections 2. This necessity aligns well with the MoE methodology, which processes different samples with specific experts. Building on this principle, we introduce a flexible and efficient Node-wise Filtering via Mixture of Experts (NODE-MOE) framework, designed to dynamically apply appropriate filters to nodes based on their structural characteristics.

The overall NODE-MOE framework is illustrated in Figure 4, which consists of two primary components: the gating model and the multiple expert models. With the graph data as input, the gating model $g(\cdot)$ computes the weight assigned to each expert for every node, reflecting the relevance of each expert's contribution to that specific node. Each expert model, implemented as any GNN with different filters, generates node representations independently. The final node classification is determined by a weighted sum of these representations, where the weights are those assigned by the gating model. The prediction for node i can be represented by:

$$\hat{y}_i = \text{Classifier}\left(\sum_{o=1}^{m} g(\mathbf{A},\mathbf{X})_{i,o} E_o(\mathbf{A},\mathbf{X})_i\right), \tag{1}$$

where m is the number of experts, E_o denotes the o-th expert, $g(\mathbf{A},\mathbf{X})_{i,o}$ represents the weight assigned to the o-th expert for node i by the gating model, and Classifier is a classifier, which could be a model like a neural network or a simple activation function like Softmax. In the following, we will delve into the specific designs of the gating model and the expert models.

3.2 Gating Model

The gating model is a pivotal component of the Node-MoE framework, aimed at selecting the most appropriate experts for each node. Its primary function is to dynamically assign higher weights to experts whose filtering characteristics best match the node's patterns. For instance, an expert utilizing a high-pass filter may receive a higher weight for a node that exhibits heterophilic patterns. However, a significant challenge arises as there is no explicit ground truth indicating which pattern each node belongs to. In traditional MoE models, the gating model often utilizes a straightforward feed-forward network that processes the features of the sample as its input (Shazeer et al., 2017; Riquelme et al., 2021; Du et al., 2022; Wang et al., 2024). Nevertheless, the nodes with different patterns may share similar node features, making this method ineffective.

To address this challenge, we estimate node patterns by incorporating the contextual features surrounding each node. If a node's features significantly differ from those of its neighboring nodes, it is likely that this node exhibits a heterophilic pattern. Specifically, the input to our gating model includes a composite vector $[\mathbf{X}, |\mathbf{A}\mathbf{X} - \mathbf{X}|, |\mathbf{A}^2\mathbf{X} - \mathbf{X}|]$. This vector combines the node's original features with the absolute differences between its features and those of its neighbors over one and two hops, respectively, to indicate the node's structural patterns. Moreover, as discussed in Section 2.1, different structural patterns are not uniformly distributed across the graph, and distinct communities may exhibit varying structural characteristics. To capitalize on this phenomenon, we employ GNNs with low-pass filters, such as GIN (Xu et al., 2018), for the gating model. These networks are chosen due to their strong community detection capabilities (Shchur & Günnemann, 2019; Bruna & Li, 2017), ensuring that neighboring nodes are likely to receive similar expert selections. Experimental results in Section 4.3 clearly demonstrate the proposed gating can efficiently assign different nodes to their suitable filters.

3.3 Expert Models

The mixed structural patterns observed in real-world graphs necessitate that the expert models in our Node-MoE framework possess diverse capabilities. To achieve this, we consider multiple existing GNNs equipped with different filters. Traditional GNNs often utilize fixed filters, which may not adequately capture the complexity of diverse structural patterns. To address this limitation, we opt for GNNs with learnable graph convolutions (Chien et al., 2020; Bianchi et al., 2021; He et al., 2021; 2022), which are capable of adapting their filters to better fit the graph structural patterns. However, the same experts would make the gating model hard to learn the right features (Chen et al., 2022) and may result in all experts' filters being optimized in the same direction. To encourage diversity and ensure that each expert is adept at handling specific structural patterns, we adopt a differentiated initialization strategy for the filters in the experts. Instead of using a fixed filter initialization, we initialize different experts with distinct types of filters, such as low-pass, constant, and high-pass filters. More details can be found in Section 4.

Filter Smoothing Loss. While integrating multiple experts with diverse filters significantly enhances the expressive capacity of our Node-MoE framework, this complexity can also make the model more challenging to fit. For example, training multiple filters simultaneously may lead to oscillations in the spectral domain for each filter as shown in Appendix B. This not only complicates fitting the model to the data but also impacts its explainability. The specific role and function of each oscillating filter become difficult to discern, making it harder to understand and interpret the model's behavior. To mitigate these issues, we introduce a filter smoothing loss designed to ensure that the learned filters exhibit smooth behavior in the spectral domain. This loss is defined as follows:

$$L_s^o = \sum_{i=1}^{K} |f_o(s_i) - f_o(s_{i-1})|^2, \tag{2}$$

where $f_o(\cdot)$ is the learnable filter function of the o-th expert, $s_0 \leq s_1 \leq \cdots \leq s_K$ are $K+1$ values spanning the spectral domain. By minimizing the activation differences between neighboring values in the spectral domain, the filter functions become smoother. The overall training loss is then given by $L = L_{task} + \gamma \sum_{o=1}^{m} L_s^o$, where the L_{task} is the node classification loss and γ is a hyperparameter that adjusts the influence of the filter smoothing loss.

3.4 Top-K gating

The soft gating that integrates all experts in the Node-MoE framework significantly enhances its modeling capabilities, but it also increases computational complexity since each expert must process all samples. To improve computational efficiency while maintaining performance, we introduce a variant of NODE-MOE by leveraging the Top-K gating mechanism (Shazeer et al., 2017). In this variant, the NODE-MOE with Top-K gating selectively activates only the top k experts with the highest relevance for each node. Specifically, the gating function for a node v_i is defined as $g(v_i) = \mathrm{Softmax}\left(\mathrm{TopK}\left(g\left(\mathbf{A}, \mathbf{X}\right)_i, k\right)\right)$. To prevent the gating model from consistently favoring a limited number of experts, we incorporate a load-balancing loss as suggested by Shazeer et al. (2017).

3.5 TIME COMPLEXITY OF NODE-MOE

The time complexity of the proposed NODE-MOE can be significantly reduced through sparse Top-K gating. For instance, when setting $K = 1$, each node only needs to be processed by a single expert. In this case, the time complexity of NODE-MOE becomes comparable to that of a single expert, with the addition of a lightweight gating model. In the section 4.4, we demonstrate the effective and efficiency of the proposed NODE-MOE.

4 EXPERIMENT

In this section, we conduct comprehensive experiments to validate the effectiveness of the proposed NODE-MOE. Specifically, we aim to address the following research questions: **RQ1:** How does NODE-MOE perform compared with the state-of-the-art baselines on both homophilic and heterophilic graphs? **RQ2:** Do the experts within NODE-MOE learn diverse structural patterns and does the gating model accurately assign each node to its most suitable experts? **RQ3:** How do different factors affect the performance of NODE-MOE?

4.1 EXPERIMENTAL SETTINGS.

Datasets. To evaluate the efficacy of our proposed NODE-MOE, we conduct experiments across seven widely used datasets. These include four homophilic datasets: Cora, CiteSeer, Pubmed (Sen et al., 2008), and ogbn-arxiv (Hu et al., 2020); along with four heterophilic datasets: Chameleon, Squirrel (Pei et al., 2020), Penn94 and pokec Lim et al. (2021a). For Cora, CiteSeer, and Pubmed, we generate ten random splits, distributing nodes into 60% training, 20% validation, and 20% testing partitions. For the heterophilic datasets, we utilize the ten fixed splits as specified in Pei et al. (2020) and Lim et al. (2021a). The ogbn-arxiv dataset is evaluated using its standard split (Hu et al., 2020). We run the experiments 3 times for each split and report the average performance and standard deviation. More details about these datasets are shown in Appendix C.1.

Baselines. We compare our method with a diverse set of baselines, which can be divided into five categories: (1) Non-GNN methods like MLP and Label Propagation (LP) (Zhou et al., 2003); (2) Homophilic GNNs utilizing fixed low-pass filters such as GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), APPNP (Gasteiger et al., 2018), and GCNII (Chen et al., 2020); (3) Heterophilic GNNs including AutoGCN (Wu et al., 2022), WRGCN (Suresh et al., 2021), PC-Conv (Li et al., 2024), ACM-GCN (Luan et al., 2022), ASGAT (Li et al., 2021) and LinkX (Lim et al., 2021a); (4) GNNs with learnable filters like GPRGNN (Chien et al., 2020) and ChebNetII (He et al., 2022); (5) MoE-based GNNs such as GMoE (Wang et al., 2024) and Mowst (Zeng et al., 2023).

NODE-MOE settings. The proposed NODE-MOE framework is highly flexible, allowing for a wide range of choices in both gating and expert models. In this work, we employ the GIN (Xu et al., 2018) as the gating model due to its exceptional expressive power and ability to leverage community properties as discussed in Section 2. As for the expert models, we utilize ChebNetII (He et al., 2022), known for its efficiency in learning filters. Specifically, we experiment with configurations of 2, 3, and 5 ChebNetII experts, each initialized with different filters. More details and parameter settings are in Appendix C.3.

4.2 PERFORMANCE COMPARISON ON BENCHMARK DATASETS

In this section, we evaluate the efficacy of the proposed NODE-MOE across both homophilic and heterophilic datasets. The results of node classification experiments are detailed in Table 1. From the results, we can have the following observations:

- The proposed NODE-MOE demonstrates robust performance across both homophilic and heterophilic datasets, outperforming the baselines in most cases. This indicates its effectiveness in handling diverse graph structures.

- The GNNs and methods like LP that use fixed low-pass filters generally do well on homophilic datasets but tend to underperform on heterophilic datasets. Conversely, specialized models like LinkX, designed for heterophilic graphs, do not perform as well on homophilic datasets.
- The GNNs equipped with learnable filters generally perform well on both types of datasets, as they can adapt their filters to the dataset's structural patterns. However, their performance is still not optimal. The proposed Node-MoE, which utilizes multiple ChebNetII as experts, significantly outperforms a single ChebNetII, especially on heterophilic datasets. This result validates the effectiveness of our node-wise filtering approach.
- We also compare the proposed NODE-MOE with two MoE methods, i.e., GMoE, which adapts the receptive field for each node but still applies traditional graph convolution with low-pass filters and Mowst, which selects MLP or GNN for prediction based on the confidence of GNN. We can find NODE-MOE consistently outperforms GMoE and Mowst across all datasets.

Table 1: Node classification accuracy (%) on benchmark datasets. OOM means out-of-memory. The bold and underline markers denote the best and second-best performance respectively. *indicates a t-test with $p < 0.05$.

Methods	Homophilic datasets				Heterophilc datasets			
	Cora	CiteSeer	PubMed	ogbn-arxiv	Chameleon	Squirrel	Penn94	Pokec
MLP	76.49 ± 1.13	73.15 ± 1.36	86.14 ± 0.64	55.68 ± 0.11	48.11 ± 2.23	31.68 ± 1.90	73.61±0.40	62.39 ± 0.06
LP	86.05 ± 1.35	69.39 ± 2.01	83.38 ± 0.64	68.14 ± 0.00	44.10 ± 4.10	31.92 ± 0.82	63.26 ± 0.41	53.28 ± 0.05
GCN	88.60 ± 1.19	76.88 ± 1.78	88.48 ± 0.46	71.91 ± 0.15	67.96 ± 1.82	54.47 ± 1.17	82.37 ± 0.24	75.43 ± 0.15
GAT	88.68 ± 1.13	76.70 ± 1.81	86.52 ± 0.56	71.92 ± 0.17	65.29 ± 2.54	49.46 ± 1.69	81.53 ± 0.55	71.77 ± 6.18
APPNP	88.49 ± 1.28	<u>77.42 ± 1.47</u>	87.56 ± 0.52	71.61 ± 0.30	54.32 ± 2.61	36.41 ± 1.94	74.33 ± 0.38	62.58 ± 0.08
GCNII	88.12 ± 1.05	77.30 ±1.58	**90.17 ± 0.57**	<u>72.74 ± 0.16</u>	55.54 ± 2.02	56.63 ± 1.17	82.92±0.59	78.94 ± 0.11
AutoGCN	87.59 ± 1.17	75.12 ± 1.94	89.13 ± 0.51	69.34 ± 0.63	65.21 ± 2.97	45.55 ± 1.54	81.02 ± 0.16	79.49 ± 0.33
WRGCN	88.06 ± 1.50	76.28 ± 1.98	86.39 ± 0.55	>24h	65.24 ± 0.87	48.85 ± 0.78	75.50 ± 0.09	>24h
PC-Conv	<u>88.85 ± 1.29</u>	77.30 ± 1.79	85.79 ± 0.64	67.21 ± 0.19	66.86 ± 1.97	44.75 ± 1.58	<u>85.36 ± 0.06</u>	77.86 ± 0.07
ACMGCN	88.01 ± 1.26	76.52 ± 1.72	89.51 ± 0.49	62.09 ± 1.29	69.62 ± 1.22	57.02 ± 0.79	83.02 ± 0.65	74.13 ± 0.14
ASGAT	86.63 ± 1.51	73.76 ± 1.17	OOM	OOM	66.50 ± 2.80	55.80 ± 3.20	OOM	OOM
LinkX	82.89 ± 1.27	70.05 ± 1.88	84.81 ± 0.65	66.54 ± 0.52	68.42 ± 1.38	<u>61.81 ± 1.80</u>	84.71 ± 0.52	<u>81.86 ± 0.21</u>
GPR-GNN	88.54 ± 0.67	76.44 ± 1.89	88.46 ± 0.31	71.78 ± 0.18	62.85 ± 2.90	54.35 ± 0.87	83.54 ± 0.32	80.74±0.22
ChebNetII	88.71 ± 0.93	76.93 ± 1.57	88.93 ± 0.29	72.32 ± 0.23	71.14 ± 2.13	57.12 ± 1.13	84.86 ± 0.33	81.16 ± 0.04
GMoE	87.27 ± 1.74	76.56 ± 1.57	88.14 ± 0.56	71.74 ± 0.29	<u>71.88 ± 1.60</u>	51.97 ± 3.16	75.76 ± 4.39	59.30 ± 1.92
Mowst	86.18 ± 1.45	75.27 ± 2.19	88.92 ± 0.61	70.37 ± 0.16	65.50 ± 1.86	52.14 ± 1.25	79.78 ± 0.26	77.05 ± 0.06
NODE-MOE	**89.38 ± 1.26***	**77.78 ± 1.36**	<u>89.58 ± 0.60</u>	**73.19 ± 0.22***	**73.64 ± 1.80***	**62.31 ± 1.98***	**85.37 ± 0.31**	**82.94 ± 0.06***

4.3 ANALYSIS OF NODE-MOE

In this section, we delve into an in-depth analysis of the behaviors of NODE-MOE to demonstrate its rationality and effectiveness. We aim to uncover several key aspects of how NODE-MOE operates and performs: What specific types of filters does Node-MoE learn? Are nodes appropriately assigned to these diverse filters by the gating model? Finally, which types of nodes benefit the most from the proposed NODE-MOE for different datasets? We conduct experiments on both CiteSeer and Chameleon datasets using configurations with 2 experts. The results for the Chameleon dataset are presented below. For more results and analysis, please refer to Appendix D.

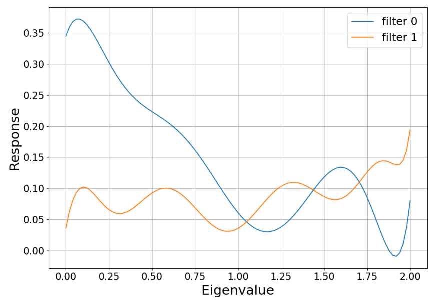


Figure 5: Learned 2 filters by NODE-MOE on Chameleon.

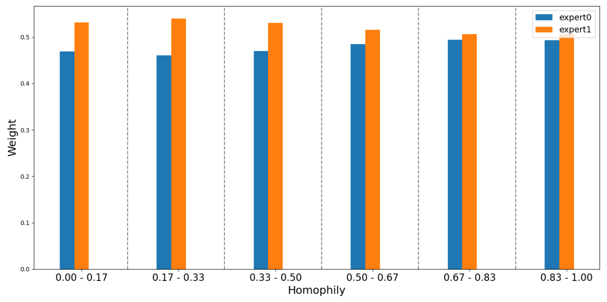
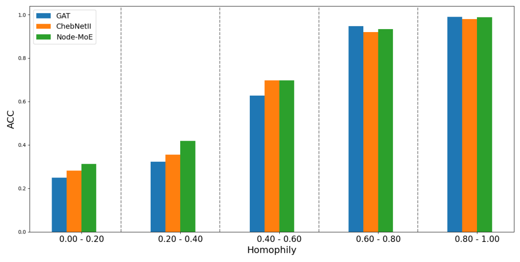


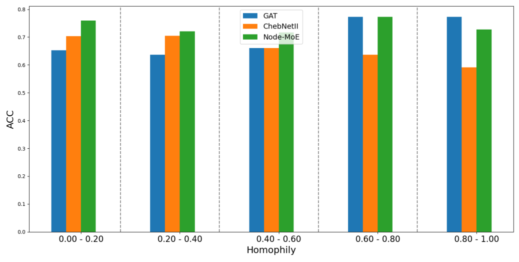
Figure 6: The average weight generated by the gating model for nodes in different homophily groups on Chameleon.

Figure 5 showcases the two filters learned by NODE-MOE on the Chameleon dataset, where filter 0 functions as a low-pass filter and filter 1 as a high-pass filter. To analyze the behavior of the gating model in NODE-MOE, we split nodes into different groups based on their homophily levels. Figure 6 displays the weights assigned by the gating model to these two experts. The results reveal that nodes with lower homophily levels predominantly receive higher weights for the high-pass filter (filter 1), and as the homophily level increases, the weight for this filter correspondingly decreases. This pattern confirms our design that nodes with varying structural patterns require different filters, demonstrating the effectiveness of the proposed gating model.

Figure 7 shows the performance of different models on node groups with varying levels of homophily. We observe that the proposed NODE-MOE improves the performance of low-homophilic nodes in the Cora dataset, while it enhances the performance of high-homophilic nodes in the Chameleon dataset, compared to the single-expert ChebNetII. Besides, NODE-MOE outperforms GAT on low-homophilic nodes in both datasets. These observations further demonstrate the effectiveness of our node-wise filtering method.



(a) Cora



(b) Chameleon

Figure 7: The performance of different models on node groups with different homophily.

4.4 ABLATION STUDIES

In this section, we conduct ablation studies to further investigate the effectiveness of two key components within the Node-MoE framework: the gating model and the filter smoothing loss. For the gating model, we explore two variants: a traditional MLP-based gating mechanism that utilizes the input features $\mathbf{X}$, and the Top-K gating approach as detailed in Section 3.4. Specifically, we choose $K = 1$ to ensure the proposed NODE-MOE has similar efficiency with the single expert. Figure 8 presents the results on CiteSeer, ogbn-arxiv, Chameleon, and Squirrel datasets. We observe two findings: (1) Traditional gating does not perform as well as the proposed gating in NODE-MOE and only achieves results comparable to an individual ChebNetII expert. (2) The Top-1 gating, which selects only one expert, can achieve similar results to those of the soft gating NODE-MOE that utilizes all experts. This indicates that the proposed NODE-MOE can effectively enhance performance while maintaining a complexity level comparable to that of an individual expert model.

We compared the average training time of the proposed NODE-MOE with Top-1 gating. Specifically, we select two large datasets, ogbn-arxiv and pokec, and compared the average training time of NODE-MOE with 3 and 5 experts, denoted as NODE-MOE-3 and NODE-MOE-5, respectively. As shown in Table 2, despite utilizing 3 or 5 experts, NODE-MOE 's training time remains comparable to that of the single-expert ChebNetII as the gating model only select Top-1 expert, demonstrating its efficiency.

Table 2: Average training time (s) per epoch.

Dataset	ChebNetII	NODE-MOE-3	NODE-MOE-5
ogbn-arxiv	1.57	1.77	1.93
pokec	15.58	16.6	16.79

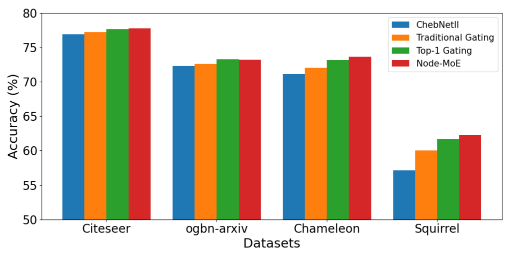


Figure 8: The performance comparison of different gating variants.

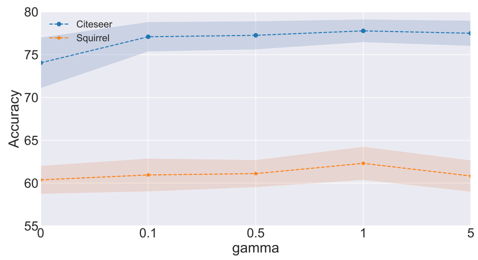


Figure 9: The performance with different weight parameters γ of the filter smoothing loss.

We also investigate the impact of the weight parameter, γ, of the filter smoothing loss on the overall performance. Specifically, we conduct experiments on the Citeseer and Squirrel datasets and the γ is chosen in $[0, 0.1, 0.5, 1, 5]$. As shown in Figure 9, incorporating the filter smoothing loss generally enhances performance, especially for the Citeseer dataset. The reason is that the filter smoothing loss can mitigate filter oscillation, which may lead to the model being hard to learn. For more detailed insights into the effects of the filter smoothing loss, please refer to Appendix B.

Additionally, we explore the effects of the number of experts and the value of K in Top-K gating. The results, shown in the Appendix D.2 and D.3, demonstrate that NODE-MOE achieves excellent performance with just a few experts (e.g., 2) and small K values (e.g., 1). We also evaluate NODE-MOE's performance under noisy feature settings. As shown in Appendix D.4, NODE-MOE still outperforms the single-expert model even at higher noise levels, though the performance gap tends to decrease as noise increases.

5 RELATED WORKS

Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Veličković et al., 2017) have achieved remarkable success across a wide range of tasks (Zhou et al., 2020). Most GNNs usually follow the message-passing mechanism (Gilmer et al., 2017), which can be regarded as low-pass graph filters (Nt & Maehara, 2019; Zhao & Akoglu, 2019). As a result, these GNNs are usually suitable for homophilic graphs. To address heterophilic graphs, specialized models like GloGNN (Li et al., 2022), LinkX (Lim et al., 2021a), MixHop (Abu-El-Haija et al., 2019) haven been developed. Additionally, models such as Bernnet (He et al., 2021), GPRGNN (Chien et al., 2020), and ChebNetII (He et al., 2022) feature learnable filters that adapt to various graph types. Recent studies have highlighted that real-world graphs often exhibit a mixture of structural patterns (Suresh et al., 2021; Li et al., 2022; Mao et al., 2024). Traditional GNNs typically apply the same global filter across all nodes, which can be suboptimal for such mixed scenarios. In response, our proposed NODE-MOE introduces a node-wise filtering approach, applying distinct filters to nodes based on their individual patterns, enhancing adaptability and performance. We note there are few methods, such as ACM-GNN (Luan et al., 2022), AutoGCN (Wu et al., 2022), PC-Conv (Li et al., 2024) and ASGAT (Li et al., 2021) also leverage multiple filters. **Our method is distinct from these methods:** these models typically use post-fusion, where all nodes are passed through all filters, and the resulting representations are then combined using mechanisms like attention. However, this post-fusion strategy increases computational cost, as all nodes must be processed by every filter. In contrast, the Top-K gating mechanism in NODE-MOE significantly reduces computational cost. Furthermore, the filtered representations in these methods may not accurately capture the importance of each filter, as indicated by the poor calibration of GNNs (Hsu et al., 2022). The experimental results in section 4.2 show NODE-MOE outperforms all these post-fusion methods. Additionally, while many of these methods use predefined filters, our results in Appendix D.5 show that learnable filters perform better than multiple fixed filters.

Mixture of Experts (MoE) (Jacobs et al., 1991; Jordan & Jacobs, 1994) architecture has been widely used in NLP (Du et al., 2022; Zhou et al., 2022) and Computer Vision (Riquelme et al., 2021) to improve efficiency of large models. In graph domain, GraphMETRO (Wu et al., 2023) leverage MoE to address the graph distribution shift issue. GMoE (Wang et al., 2024) utilizes MoE to adaptive select propagation hops for different nodes. Link-MoE (Ma et al., 2024) finds different node pairs require different heuristics to predict and different GNN4LP models have different abilities for different heuristics. They leverage MoE to use different GNN4LP models for different node pairs. Despite these advancements, these methods still face challenges in handling complex graph patterns. Another related work is Mowst (Zeng et al., 2023), which selects the prediction from either MLP or GNN based on the confidence of the GNN's prediction. However, this method still relies on post-fusion and uses fixed filters, limiting its flexibility. In contrast, the proposed NODE-MOE demonstrates both superior effectiveness and efficiency.

6 CONCLUSION

In this paper, we explored the complex structural patterns inherent in real-world graph datasets, which typically exhibit a mixture of homophilic and heterophilic patterns. Notably, these patterns exhibit significant variability across different communities within the same dataset, highlighting the intricate and diverse nature of graph structures. Our theoretical analysis reveals that the conventional single global filter, commonly used in many GNNs, is often inadequate for capturing such complex structural patterns. To address this limitation, we proposed the node-wise filtering method, NODE-MOE, a flexible and effective solution that adaptively selects appropriate filters for different nodes. Extensive experiments demonstrate the proposed NODE-MOE demonstrated excellent performance on both homophilic and heterophilic datasets. Further, our behavioral analysis and ablation studies validate the design and effectiveness of the proposed NODE-MOE.

7 Reproducibility Statement

The experimental setup, including hyperparameters settings and dataset details, along with a link to anonymously source code, can be found in Appendix C to ensure reproducibility.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pp. 21–29. PMLR, 2019.

Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. *arXiv preprint arXiv:2102.06966*, 2021.

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*, 44(7):3496–3507, 2021.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 3950–3957, 2021.

Joan Bruna and X Li. Community detection with graph neural networks. *stat*, 1050:27, 2017.

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

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding mixture of experts in deep learning. *arXiv preprint arXiv:2208.02813*, 2022.

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

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

Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. Contextual stochastic block models. *Advances in Neural Information Processing Systems*, 31, 2018.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.

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

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

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

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

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

Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? *Advances in Neural Information Processing Systems*, 35:13775–13786, 2022.

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

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Zhimeng Jiang, Xiaotian Han, Chao Fan, Zirui Liu, Xiao Huang, Na Zou, Ali Mostafavi, and Xia Hu. Topology matters in fair graph learning: a theoretical pilot study. 2023.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

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

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*, volume 38, pp. 13437–13445, 2024.

Shouheng Li, Dongwoo Kim, and Qing Wang. Beyond low-pass filters: Adaptive feature propagation on graphs. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pp. 450–465. Springer, 2021.

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

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

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

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

Li Ma, Haoyu Han, Juanhui Li, Harry Shomer, Hui Liu, Xiaofeng Gao, and Jiliang Tang. Mixture of link predictors. *arXiv preprint arXiv:2402.08583*, 2024.

Yao Ma and Jiliang Tang. *Deep learning on graphs*. Cambridge University Press, 2021.

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

Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *Advances in Neural Information Processing Systems*, 36, 2024.

Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.

Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

John Palowitch, Anton Tsitsulin, Bryan Perozzi, and Brandon A Mayer. Synthetic graph generation to benchmark graph learning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

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.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. *Advances in Neural Information Processing Systems*, 36, 2024.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

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

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*, 2019.

Jiaqi Sun, Lin Zhang, Shenglin Zhao, and Yujiu Yang. Improving your graph neural networks: a high-frequency booster. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 748–756. IEEE, 2022.

Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1541–1551, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Kompella, Zhangyang Wang, et al. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Shirley Wu, Kaidi Cao, Bruno Ribeiro, James Zou, and Jure Leskovec. Graphmetro: Mitigating complex distribution shifts in gnns via mixture of aligned experts. *arXiv preprint arXiv:2312.04693*, 2023.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Beyond low-pass filtering: Graph convolutional networks with automatic filtering. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6687–6697, 2022.

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

Hanqing Zeng, Hanjia Lyu, Diyi Hu, Yinglong Xia, and Jiebo Luo. Mixture of weak & strong experts on graphs. *arXiv preprint arXiv:2311.05185*, 2023.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

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

Appendix

A Proof of Theorem 1

In this section, we present the proof of Theorem 1. This theorem analyzes the separability when different filters are applied to graphs generated by a mixed CSBM model in Defination 1- $CSBM(n, \boldsymbol{\mu}, \boldsymbol{\nu}, (p_0, q_0), (p_1, q_1), P)$ using a linear classifier.

Notably, the following proof is derived based on Baranwal et al. (2021), which analyzes the linear separability of a single graph convolution under a single CSBM model with only one pattern - $CSBM(n, \boldsymbol{\mu}, \boldsymbol{\nu}, (p, q))$. We extend the analysis to graphs with mixed CSBM models. Besides, we analyze the scenarios in which different filters are applied to the same graph.

We follow the assumption 1 and 2 in Baranwal et al. (2021): The graph size n should be relatively large with $\omega(d \log d) \leq n \leq O(\text{poly}(d))$, and the graph is not too sparse with $p_0, q_0, p_1, q_1 = \omega\left(\log^2(n)/n\right)$.

A.1 Proof of part 1 of Theorem 1

Proof. For the low-pass filter, consider the filtered feature $\tilde{\mathbf{X}} = \mathbf{D}^{-1}\mathbf{A}\mathbf{X}$. Due to the normal distribution of node feature $\mathbf{X}$, the filtered feature of node i still follows the normal distribution. Specifically, the mean of nodes in different classes and partterns can be represented by:

$$m(i) = E(\tilde{\mathbf{X}}_i) = \begin{cases} \dfrac{p_0\boldsymbol{\mu} + q_0\boldsymbol{\nu}}{p_0 + q_0}(1 + o(1)) & \text{for } i \in C_0 \text{ and } i \in H_0 \\ \dfrac{q_0\boldsymbol{\mu} + p_0\boldsymbol{\nu}}{p_0 + q_0}(1 + o(1)) & \text{for } i \in C_1 \text{ and } i \in H_0 \\ \dfrac{p_1\boldsymbol{\mu} + q_1\boldsymbol{\nu}}{p_1 + q_1}(1 + o(1)) & \text{for } i \in C_0 \text{ and } i \in H_1 \\ \dfrac{q_1\boldsymbol{\mu} + p_1\boldsymbol{\nu}}{p_1 + q_1}(1 + o(1)) & \text{for } i \in C_1 \text{ and } i \in H_1 \end{cases},$$

where C_0 and C_1 represent the class 0 and class 1, respectively; H_0 and H_1 are the homophilic and heterophilic node sets, respectively. The covariance matrix can be represented by: $Cov(\tilde{\mathbf{X}}_i) = \frac{1}{d\mathbf{D}_{ii}}\mathbf{I}.$. Lemma 2 in Baranwal et al. (2021) demostrate that for any unit vector $\mathbf{w}$, we have: $\left|\left(\tilde{\mathbf{X}}_i - m(i)\right) \cdot \mathbf{w}\right| = O\left(\sqrt{\frac{\log n}{dn(p_0+q_0)}}\right)$.

If we only consider the nodes with homophilic patterns, i.e., $i \in H_0$, we can find the optimal linear classifier with $\mathbf{w}^* = R\frac{\boldsymbol{\nu}-\boldsymbol{\mu}}{\|\boldsymbol{\nu}-\boldsymbol{\mu}\|}$ and $\mathbf{b}^* = -\frac{1}{2}\langle\boldsymbol{\nu} + \boldsymbol{\mu}, \mathbf{w}^*\rangle$. We also have the assumption that the distance between $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are relatively large, with $\|\boldsymbol{\nu} - \boldsymbol{\mu}\| = \Omega\left(\frac{\log n}{dn(p_0+q_0)}\right)$.

Then, for $i \in C_0$ and $i \in H_0$, we have:

$$\begin{aligned} \langle\tilde{\mathbf{X}}_i, \mathbf{w}^*\rangle + b^* &= \frac{\langle p_0\boldsymbol{\mu} + q_0\boldsymbol{\nu}, \mathbf{w}^*\rangle}{p_0 + q_0}(1 + o(1)) + O\left(\|\mathbf{w}^*\|\sqrt{\frac{\log n}{dn(p+q)}}\right) - \frac{1}{2}\langle\boldsymbol{\nu} + \boldsymbol{\mu}, \mathbf{w}^*\rangle \\ &= \frac{\langle 2p_0\boldsymbol{\mu} + 2q_o\boldsymbol{\nu} - (p_0 + q_0)(\boldsymbol{\mu} + \boldsymbol{\nu}), \mathbf{w}^*\rangle}{p_0 + q_0}(1 + o(1)) + o(\|\mathbf{w}^*\|) \\ &= \frac{p_0 - q_0}{2(p_0 + q_0)}\langle\boldsymbol{\mu} - \boldsymbol{\nu}, \mathbf{w}*\rangle(1 + o(1)) + o(\|\mathbf{w}^*\|) \\ &= -\frac{R(p_0 - q_0)}{2(p_0 + q_0)}\|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1)) < 0 \end{aligned}$$

Similarly, for $i \in C_1$ and $i \in H_0$, we have:

$$\langle \tilde{\mathbf{X}}_i, \mathbf{w}^* \rangle + b^* = -\frac{R(q_0 - p_0)}{2(p_0 + q_0)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1)) > 0$$

Therefore, the linear classifier with w^* and b^* can separate class C_0.

However, if we apply this linear classifier to the heterophilic node set H_1, where $p_1 < q_1$, we have:

$$\langle \tilde{\mathbf{X}}_i, \mathbf{w}^* \rangle + b^* = \begin{cases} -\frac{R(p_1 - q_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1)) > 0 & \text{for } i \in C_0 \text{ and } i \in H_1 \\ -\frac{R(q_1 - p_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1)) < 0 & \text{for } i \in C_1 \text{ and } i \in H_1 \end{cases}$$

Therefore, all nodes in H_1 are misclassified. The binary cross-entropy over node set H_1 can be represented by:

$$\begin{aligned} L(H_1, \mathbf{w}^*, b^*) &= \frac{1}{|H_1|} \sum_{i \in H_1} -y_i \log\left(\sigma\left(\left\langle \tilde{\mathbf{X}}_i, \mathbf{w}^* \right\rangle + \tilde{b}\right)\right) - (1 - y_i) \log\left(1 - \sigma\left(\left\langle \tilde{\mathbf{X}}_i, \mathbf{w}^* \right\rangle + b^*\right)\right) \\ &= \frac{1}{|H_1|} \sum_{i \in H_1} \log\left(1 + \exp\left((1 - 2y_i)\left(\left\langle \tilde{X}_i, \tilde{\mathbf{w}} \right\rangle + b^*\right)\right)\right) \\ &= \log\left(1 + \exp\left(-\frac{R(p_1 - q_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1))\right)\right) \end{aligned}$$

As for $x = -\frac{R(p_1 - q_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\| > 0$, we have $e^x \geq x$. As a result, we have

$$L(H_1, \mathbf{w}^*, b^*) \geq \frac{R(q_1 - p_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1))$$

□

A.2 Proof of part 2 of Theorem 1

Proof. Suppose we apply a high-pass filter to the heterophilic nodes H_1 and the filtered features are $\tilde{\mathbf{X}} = -\mathbf{D}^{-1}\mathbf{A}\mathbf{X}$. For nodes in H_1,

$$m(i) = E(\tilde{\mathbf{X}}_i) = \begin{cases} -\frac{p_1 \boldsymbol{\mu} + q_1 \boldsymbol{\nu}}{p_1 + q_1}(1 + o(1)) & \text{for } i \in C_0 \text{ and } i \in H_1 \\ -\frac{q_1 \boldsymbol{\mu} + p_1 \boldsymbol{\nu}}{p_1 + q_1}(1 + o(1)) & \text{for } i \in C_1 \text{ and } i \in H_1 \end{cases}$$

Therefore, if we apply the same linear classifier with $\mathbf{w}^*$ and b^*, then we have:

$$\langle \tilde{\mathbf{X}}_i, \mathbf{w}^* \rangle + b^* = \begin{cases} \frac{R(p_1 - q_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1)) < 0 & \text{for } i \in C_0 \text{ and } i \in H_1 \\ \frac{R(q_1 - p_1)}{2(p_1 + q_1)} \|\boldsymbol{\mu} - \boldsymbol{\nu}\|(1 + o(1)) > 0 & \text{for } i \in C_1 \text{ and } i \in H_1 \end{cases}$$

As a result, the same linear classifier can separate both the homophilic set H_0 and heterophilic set H_1.

□

A.3 Proof of part 2 of Theorem 1

Proof. Let $\mathbf{A}$ be the adjacency matrix of G, $\mathbf{D}$ be the diagonal degree matrix where D_{ii} is the degree of node i, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the feature matrix with $\mathbf{X}_i$ denoting the feature vector of node i. The filtered feature is defined as:

$$\tilde{\mathbf{X}} = \mathbf{D}^{-1}\mathbf{A}\mathbf{X},$$

where $\mathbf{D}^{-1}\mathbf{A}$ averages features across neighbors of each node.

We now analyze the squared feature change, $f_i^2 = (\tilde{\mathbf{X}}_i - \mathbf{X}_i)^2$, which represents the squared deviation of the aggregated feature from the original feature. For node i, this is:

$$f_i^2 = \left(\sum_{j \in \mathcal{N}(i)} \frac{\mathbf{X}_j}{D_{ii}} - \mathbf{X}_i \right)^2,$$

where $\mathcal{N}(i)$ is the neighborhood of i.

Nodes are divided into:

- H_0: Homophilic nodes where intra-class connections dominate.
- H_1: Heterophilic nodes where inter-class connections dominate.

Each node belongs to one of two classes C_0 or C_1, with the class means $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, respectively.

If $f_i = \tilde{\mathbf{X}}_i - \mathbf{X}_i \sim N(\mu_{f_i}, \sigma^2_{f_i})$, then f_i^2 follows a scaled Chi-squared distribution:

$$f_i^2 \sim \frac{\sigma^2_{f_i}}{\sigma^2} \chi^2(1, \lambda_i),$$

where:

- $\chi^2(1, \lambda_i)$ is a non-central Chi-squared distribution with 1 degree of freedom and non-centrality parameter $\lambda_i = \frac{\mu^2_{f_i}}{\sigma^2_{f_i}}$.
- $\sigma^2_{f_i} = \frac{\sigma^2}{d}\left(1 + \frac{1}{D_{ii}}\right)$ for node i.
- The mean μ_{f_i} depends on the node type (homophilic or heterophilic):

$$\mu_{f_i} = \begin{cases} \frac{q_0}{p_0+q_0}(\nu - \mu), & \text{if } i \in H_0, C_0, \\ -\frac{q_0}{p_0+q_0}(\nu - \mu), & \text{if } i \in H_0, C_1, \\ \frac{q_1}{p_1+q_1}(\nu - \mu), & \text{if } i \in H_1, C_0. \\ -\frac{q_1}{p_1+q_1}(\nu - \mu), & \text{if } i \in H_1, C_1. \end{cases}$$

For each node i, the expected squared feature change f_i^2 is:

$$\mathbb{E}[f_i^2] = \mu^2_{f_i} + \sigma^2_{f_i},$$

and the variance of f_i^2 is:

$$\text{Var}(f_i^2) = 2\sigma^4_{f_i} + 4\mu^2_{f_i}\sigma^2_{f_i}.$$

Misclassification occurs when the squared feature changes of nodes $i \in H_0$ and $j \in H_1$ overlap. Define the difference in squared feature changes:

$$DF_{ij} = f_i^2 - f_j^2.$$

The expectation of DF_{ij} is:

$$\mathbb{E}[DF_{ij}] = \mathbb{E}[f_i^2] - \mathbb{E}[f_j^2].$$

Substituting $\mathbb{E}[f_i^2] = \mu^2_{f_i} + \sigma^2_{f_i}$, we get:

$$\mathbb{E}[DF_{ij}] = (\mu^2_{f_i} - \mu^2_{f_j}) + (\sigma^2_{f_i} - \sigma^2_{f_j}),$$

where:

- The difference in means $\mu_{f_i}^2 - \mu_{f_j}^2$ is:

$$\mu_{f_i}^2 - \mu_{f_j}^2 = \left(\frac{q_0}{p_0 + q_0} - \frac{q_1}{p_1 + q_1}\right)^2 (\nu - \mu)^2 = \Delta^2(\nu - \mu)^2.$$

  Here, $\Delta = \frac{q_0}{p_0+q_0} - \frac{q_1}{p_1+q_1}$ represents the normalized connection bias between classes.

- The variance difference $\sigma_{f_i}^2 - \sigma_{f_j}^2$ is:

$$\sigma_{f_i}^2 - \sigma_{f_j}^2 = \frac{\sigma^2}{d}\left(\frac{1}{D_{ii}} - \frac{1}{D_{jj}}\right).$$

  As $d \to \infty$, this term vanishes.

Thus:

$$\mathbb{E}[DF_{ij}] = \Delta^2(\nu - \mu)^2 + \mathcal{O}\left(\frac{1}{d}\right).$$

The variance of DF_{ij} is:

$$\mathrm{Var}(DF_{ij}) = \mathrm{Var}(f_i^2) + \mathrm{Var}(f_j^2).$$

For each node:

$$\mathrm{Var}(f_i^2) = 2\sigma_{f_i}^4 + 4\mu_{f_i}^2\sigma_{f_i}^2.$$

Since $\sigma_{f_i}^2 = \frac{\sigma^2}{d}\left(1 + \frac{1}{D_{ii}}\right)$ and $\mu_{f_i}^2 \propto \frac{1}{d}$, we get:

$$\mathrm{Var}(f_i^2) \sim \mathcal{O}\left(\frac{1}{d}\right).$$

The misclassification probability $\mathbb{P}(DF_{ij} \leq \epsilon)$ can be bounded using the Chernoff inequality:

$$\mathbb{P}(DF_{ij} \leq \epsilon) \leq \exp\left(-\frac{(\mathbb{E}[DF_{ij}] - \epsilon)^2}{2\,\mathrm{Var}(DF_{ij})}\right).$$

Substituting the results:

$$\mathbb{E}[DF_{ij}] = \Delta^2(\nu - \mu)^2 + \mathcal{O}\left(\frac{1}{d}\right), \quad \mathrm{Var}(DF_{ij}) \sim \mathcal{O}\left(\frac{1}{d}\right).$$

Thus:

$$\frac{(\mathbb{E}[DF_{ij}] - \epsilon)^2}{2\,\mathrm{Var}(DF_{ij})} \sim \mathcal{O}(d),$$

implying that the exponential decay in the Chernoff bound becomes increasingly sharp as $d \to \infty$, making $\mathbb{P}(DF_{ij} \leq \epsilon)$ approach 1.

□

B THE IMPACT OF FILTER SMOOTHING LOSS

In this section, we explore the impact of the proposed filter smoothing loss on the behavior of the learned filters in our NODE-MOE framework. Figures 10 and 11 display the effects of the NODE-MOE framework without and with the application of filter smoothing loss, respectively. Without the filter smoothing loss, as shown in Figure 10, the learned filters exhibit significant oscillations, making it challenging to discern their specific functions. In contrast, with the filter smoothing loss applied, as illustrated in Figure 11, the behavior of the filters becomes more distinct: filter 0 clearly functions as a low-pass filter, and filter 1 as a high-pass filter.

Additionally, we assessed the training dynamics of the proposed Node-MoE framework by comparing performance with and without the filter smoothing loss, while keeping other hyperparameters constant.

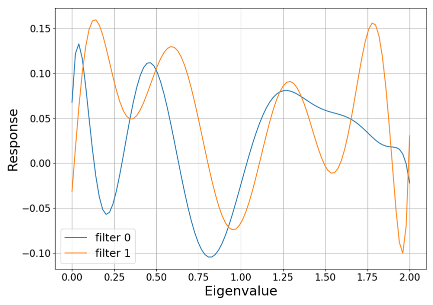


Figure 10: Learned 2 filters by NODE-MOE on Chameleon without filter smoothing loss.

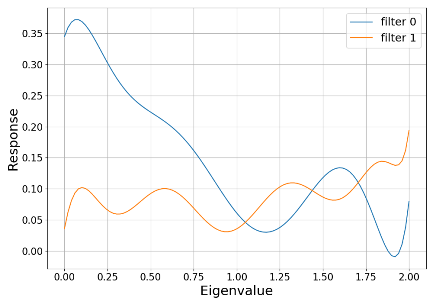


Figure 11: Learned 2 filters by NODE-MOE on Chameleon with filter smoothing loss.

For the Citeseer dataset, applying the filter smoothing loss resulted in a higher average training accuracy of 99.37 ± 0.17, compared to 93.51 ± 1.27 when the loss was not applied. A similar pattern was observed on the Squirrel dataset, where the training accuracy was 96.54 ± 1.42 with the filter smoothing loss, versus 95.54 ± 0.94 without it. These results suggest that oscillations in the filters without the smoothing loss can hinder the model's ability to fit the data effectively, resulting in suboptimal performance as shown in Section 4.4.

C DATASETS AND EXPERIMENTAL SETTINGS

In this section, we detail the datasets used and the experimental settings for both the baseline models and the proposed NODE-MOE framework.

C.1 DATASETS

We conduct experiments across seven widely recognized datasets, which encompass both homophilic and heterophilic types. The homophilic datasets include Cora, CiteSeer, and Pubmed (Sen et al., 2008), along with ogbn-arxiv (Hu et al., 2020); the heterophilic datasets comprise Chameleon, Squirrel (Pei et al., 2020), Penn94 and pokec Lim et al. (2021a). For Cora, CiteSeer, and Pubmed, we generate ten random splits, allocating nodes into training, validation, and testing sets with proportions of 60%, 20%, and 20%, respectively. For the heterophilic datasets, we adhere to the ten fixed splits as defined in Pei et al. (2020). The ogbn-arxiv dataset is assessed using its standard split as established by (Hu et al., 2020). Detailed statistics of these datasets are shown in Table 3.

Table 3: Statistics of datasets. The split ratio is for train/validation/test.

	Homophilic Datasets				Heterophilic Datasets			
	Cora	CiteSeer	PubMed	ogbn-arxiv	Chameleon	Squirrel	Penn94	pokec
#Nodes	2,708	3,327	19,717	169, 343	2,277	5,201	41,554	1,632,803
#Edges	5,429	4,732	44,338	1, 166, 243	31,421	198,493	1,362,229	30,622,564
#Classes	7	6	3	40	5	5	2	2
#Node Features	1,433	3,703	500	128	2,325	2,089	4814	65
#Split Ratio	60/20/20	60/20/20	60/20/20	54/18/28	48/32/20	48/32/20	50/25/25	50/25/25

C.2 ALGORITHM OF NODE-MOE

Algorithm 1: NODE-MOE

```
1 Input graph $\mathbf{A}$, Node feature $\mathbf{X}$, $m$ experts, i.e., $E_1, E_2, \ldots, E_m$, Gating model $g$, Top-K gating $k$
2 for $i = 1, 2, \ldots, m$ do
3     Initialize the filter $i$-th expert $E_i$
4 Calculate the gating input $\mathbf{GX} = [\mathbf{X}, |\mathbf{AX} - \mathbf{X}|, |\mathbf{A}^2\mathbf{X} - \mathbf{X}|]$
5 repeat
6     $\mathbf{G} = \text{Softmax}(\text{KeepTopK}(g(\mathbf{GX}), k))$
7     $\hat{y} = \sum_{o=0}^{m} \mathbf{G}_o E_o(\mathbf{A}, \mathbf{X})$
8     Update NODE-MOE weight by gradient descent on L
9 until Model converges;
```

The algorithm of NODE-MOE is shown in Algorithm 1. Lines 2-3 initialize the filters of experts based on the setting in Section C.3. Line 4 calculates the input for the gating model. Lines 6-7 calculate the prediction with top-k gating. Line 8 update the model based on the loss in Section 3.3.

C.3 EXPERIMENTAL SETTINGS

For the baseline models, we adopt the same parameter setting in their original paper. For the proposed NODE-MOE, we adopt GCNII as the experts. Specifically, for smaller datasets, we use GIN as the gating model, while for larger datasets, such as Pokec, we use an MLP as the gating model. Notably, the GCNII model has different learning rates and weight decay for the filters and other parameters. All the hyperparameters are tuned based on the validation accuracy from the following search space:

- Gating Learning Rate: {0.0001, 0.001, 0.01 }
- Gating Dropout: {0, 0.5, 0.8}
- Gating Weight Decay: {0, 5e-5, 5e-4}
- Expert Learning Rate for Filters: {0.001, 0.01, 0.1}
- Expert Weight Decay for Filters: {0, 5e-5, 5e-3, 5e-2 }
- Expert Learning Rate: {0.001, 0.01, 0.1, 0.5}
- Expert Dropout: {0, 0.5, 0.8}
- Filter Smoothing loss weight: {0, 0.01, 0.1, 1}
- Load balancing weight for top-k gating: {0, 0.001, 0.01, 0.1, 1}
- Number of experts: {2, 3, 5}

For the initialization of filters in ChebNetII, which uses a K-order approximation, we employ a set of initialization strategies for the polynomial coefficients. These strategies include: decreasing powers $[\alpha^0, \alpha^1, \cdots, \alpha^K]$, increasing powers $[\alpha^K, \alpha^{K-1}, \cdots, \alpha^0]$, and uniform values $[1, 1, \cdots, 1]$. For configurations with 2 or 3 experts, we set $\alpha = 0.9$. When expanding to 5 experts, we use two values of α, setting them at 0.9 and 0.8, respectively, to diversify the response characteristics of the filters. The code of the proposed NODE-MOE can be found via: https://anonymous.4open.science/r/Node-MoE-A05D/.

We use a single GPU of NVIDIA RTX A5000 24Gb, to run the experiments.

D ANALYSIS OF THE PROPOSED NODE-MOE

In this section, we provide more analysis of the proposed NODE-MOE by comprehensive experiments.

D.1 THE BEHAVIOR OF NODE-MOE WITH 2 EXPERTS

The learned filters and the corresponding gating weights for nodes with different homophily levels are illustrated below. For the Chameleon dataset, these are displayed in Figure 12 for the filters and

Figure 13 for the gating weights. Similarly, for the Citeseer dataset, the filters are shown in Figure 14 and the gating weights in Figure 15.

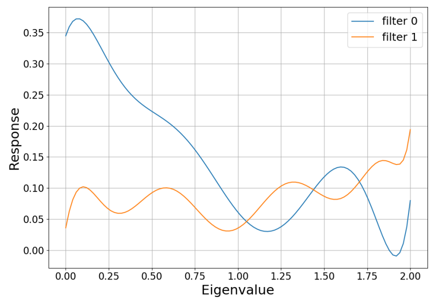


Figure 12: Learned 2 filters by NODE-MOE on Chameleon.

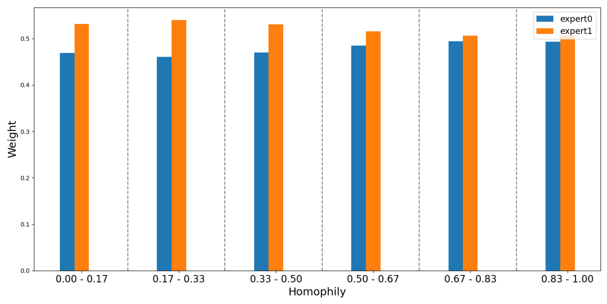


Figure 13: The average weight generated by the gating model for nodes in different homophily groups on Chameleon.

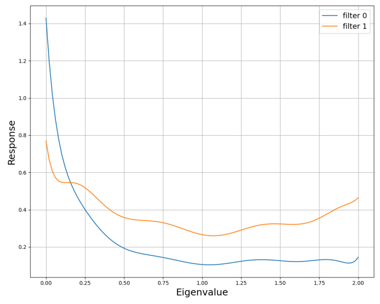


Figure 14: Learned 2 filters by NODE-MOE on Citeseer.

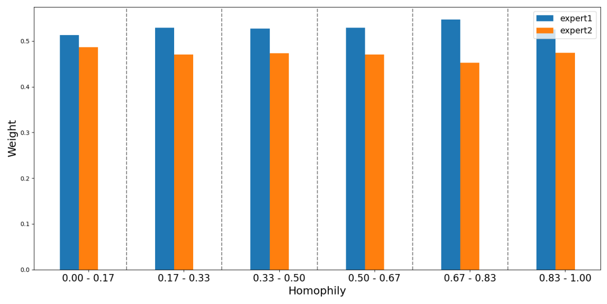


Figure 15: The average weight generated by the gating model for nodes in different homophily groups on Citeseer.

For both datasets, the learned filters demonstrate distinct characteristics: filter 0s function as low-pass filters, effectively smoothing signals, while filter 1s respond more strongly to high-frequency signals, characteristic of high-pass filters. Specifically, for the heterophilic dataset, such as Chameleon, the gating model generally assigns higher weights to filter 1, indicating a preference for high-pass filtering to accommodate the less homophilic nature of the dataset. Conversely, for the homophilic dataset, such as Citeseer, higher weights are typically assigned to filter 0, emphasizing low-pass filtering.

Moreover, within the Chameleon dataset, the weight assigned to the high-pass filter (filter 1) decreases as the homophily level increases. In contrast, in the Citeseer dataset, the weight to the low-pass filter (filter 0) increases with rising homophily levels. This pattern supports our initial hypothesis: nodes with lower homophily are better served by high-pass filters to capture the dissimilarity among neighbors, while nodes with higher homophily benefit from low-pass filters to reinforce the similarity among neighboring nodes.

D.2 EFFECT OF THE NUMBER OF EXPERTS IN NODE-MOE

In this section, we analyze the impact of using different numbers of experts in NODE-MOE with soft gating. Specifically, we experiment with 1, 2, 3, and 5 experts on the Cora and Chameleon datasets, following the same settings as outlined in Section 4.1. The experimental results, shown in Table 4, indicate that NODE-MOE achieves excellent performance with only a few experts. Notably, even with just 2 experts, it outperforms the baseline models.

D.3 EFFECT OF THE NUMBER OF SELECTED EXPERTS IN TOP-K GATING

In this section, we explore the impact of the number of selected experts K in NODE-MOE with Top-K gating. Specifically, we use 3 experts in the MoE and vary K in [1, 2, 3] in the the Top-K gating. The

Table 4: The performance of Node-MoE with different number of experts.

Experts	1	2	3	5
Cora	88.71 ± 0.93	89.19 ± 1.53	89.38 ± 1.26	89.47 ± 0.85
Chameleon	71.14 ± 2.13	73.55 ± 1.74	73.64 ± 1.80	73.42 ± 1.43

results in Table 5 demonstrate that even with Top-1 gating, Node-MoE achieves superior performance, highlighting its effectiveness and maintaining good efficiency.

Table 5: The performance of Top-K gating for Node-MoE with 3 experts.

K	Single Expert	1	2	3
Cora	88.71 ± 0.93	89.58 ± 1.44	89.56 ± 1.39	89.38 ± 1.26
Chameleon	71.14 ± 2.13	73.18 ± 1.45	73.37 ± 1.86	73.64 ± 1.80

D.4 Noise Feature

The effectiveness of the gating model in Node-MoE depends on the quality of node features, and noisy features can hinder its ability to accurately classify node patterns. In this section, we investigate the impact of noisy features on Node-MoE. Specifically, we add varying levels of Gaussian noise to the features in Cora and Chameleon dataset, i.e., $X = X + \epsilon\mathcal{N}(0,1)$ with $\epsilon \in [0, 0.01, 0.03, 0.05]$, where $\mathcal{N}(0,1)$ is the standard normal distribution.

The results on Cora and Chameleon dataset are shown in Table 6. As the noise level increases, the performance gap between Node-MoE and the single-expert ChebyNetII decreases. However, Node-MoE consistently outperforms the single expert, even with higher noise levels. The reason is that when noise is too high, the gating model may randomly assign nodes to different experts, making the learned filters converge to a performance similar to the single-expert model.

Table 6: Node classification performance with different levels of noise in the node features.

Dataset	Cora				Chameleon			
Noise	0	0.01	0.03	0.05	0	0.01	0.03	0.05
MLP	76.49 ± 1.13	59.08 ± 2.36	31.05 ± 1.23	29.58 ± 1.39	48.11 ± 2.23	30.31 ± 1.74	23.71 ± 1.79	21.80 ± 1.76
GAT	88.68 ± 1.13	87.78 ± 0.98	85.35 ± 0.98	84.61 ± 1.29	65.29 ± 2.54	64.47 ± 2.77	63.73 ± 2.19	62.92 ± 2.42
ChebyNetII	88.71±0.93	87.90 ± 1.41	86.25 ± 1.62	85.85 ± 1.09	71.14 ± 2.13	71.45 ± 1.87	71.54 ± 1.48	71.62 ± 1.56
Node-MoE	89.38 ± 1.26	87.98 ± 1.51	86.45 ± 1.35	86.05 ± 1.19	73.64 ± 1.80	73.16 ± 1.18	72.13 ± 1.62	71.95 ± 1.66

D.5 Effect of Learnable Filters in Node-MoE

The propose Node-MoE leverages ChebNetII as the experts, which automatically learn the filters. In contrast, a few prior works use multiple fixed filters. To evaluate the effect of learnable filters, we conducted experiments with fixed filters. Specifically, we used 3 experts in Node-MoE and fixed the filters in each expert to predefined types (low-pass, high-pass, and all-pass), referred to as Node-MoE-fixed.

The performance comparison between Node-MoE and Node-MoE-fixed is shown in Table 7. Node-MoE with learnable filters achieves better performance than with fixed filters across all datasets. This suggests that learnable filters are better suited for capturing the complex patterns present in real-world graphs.

Table 7: Comparison between the fixed filters and learnable filters.

Method	Cora	CiteSeer	Chameleon	squirrel
Node-MoE-Fixed	87.26 ± 1.79	76.15 ± 1.99	71.78 ± 3.37	56.96 ± 1.51
Node-MoE	89.38 ± 1.26	77.78 ± 1.36	73.64 ± 1.80	62.31 ± 1.98

D.6 Duplicates in the Chameleon and Squirrel Datasets

Platonov et al. (2023) identified that some nodes in the Chameleon and Squirrel datasets are duplicated, sharing identical neighbors, which may affect the performance of GNNs. To further validate the effectiveness of the proposed method, we also tested it on filtered versions of the Chameleon and Squirrel datasets, where the duplicate nodes were removed, referred to as Chameleon-filtered and Squirrel-filtered.

As shown in Table 8, Node-MoE continues to perform well on these filtered datasets, demonstrating its robustness.

Table 8: Node classification performance with filtered datasets.

Method	Chameleon	Squirrel	Chameleon-filterd	Squirrel-filtered
PCNet	41.23 ± 1.42	26.28 ± 0.32	34.51 ± 1.86	33.08 ± 0.20
ASGAT	66.50 ± 2.80	55.80 ± 3.20	37.4 ± 6.40	35.10 ± 1.30
ACM-GCN	69.62 ± 1.22	57.02 ± 0.79	37.78 ± 2.28	36.59 ± 1.75
Mowst	65.50 ± 1.86	52.14 ± 1.25	43.45 ± 3.90	38.04 ± 2.14
Node-MoE	73.64 ± 1.80	62.31 ± 1.98	43.32 ± 3.56	42.37 ± 1.98

D.7 Node Classification with Low Labeling Rate

We also evaluate the semi-supervised node classification performance under a low labeling rate. Specifically, we randomly select 20 training nodes per class for the Cora dataset and use 20% of the training nodes for the Chameleon dataset. The results are shown in Table 9. Even under these low-labeling rate conditions, the proposed Node-MoE continues to outperform the baseline models.

Table 9: Semi-supervised Node classification performance with low labeling rate.

	Cora 20	Chameleon 20%
GCN	79.41 ± 1.30	56.71 ± 1.72
LinkX	52.93 ± 3.04	60.62 ± 1.93
GMoE	76.00 ± 1.14	65.18 ± 1.45
ChebNetII	81.20 ± 1.04	64.66 ± 1.86
Node-MoE	82.12 ± 1.19	68.81 ± 1.96