A Supplemental Material

A.1 Notations

We first summarize the symbols in the main body, which will remain in use in the supplementary materials:

Table 1: Major symbols and definitions.

Symbols	Definitions
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	graph $\mathcal{G}$ with nodeset $\mathcal{V}$, edgeset $\mathcal{E}$
$\mathbf{A}$	$n \times n$ adjacency matrix of $\mathcal{G}$
$\mathbf{L}$	normalized graph Laplacian matrix
$\tilde{\mathbf{L}}$	$\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}$
$\mathbf{x}$	n-dimensional signal defined on the given graph $\mathcal{G}$
$\mathbf{z}$	n-dimensional filtered signal
$\mathcal{Y}$	set of class labels
y_v	class label for node $v \in \mathcal{V}$
$d_{\mathcal{G}}(v, m)$	the shortest path distance between two nodes v and m on graph $\mathcal{G}$
$N(v)$	neighbors of node v in $\mathcal{G}$
$N_i(v)$	i-hop neighbors of node v, $N_i(v) = \{m : m \in \mathcal{V} \wedge d_{\mathcal{G}}(v, m) = i\}$
Θ	the parameters of the MLP used for feature transformation
$\mathbf{W}$	the weight matrix of the feature propagation
Λ	the trainable coefficient of the feature propagation
σ	the non-linear activation function of the feature propagation

A.2 Detailed Analysis of Proposition 3.1

***Proof*.** For node v, following the assumption that its neighbors' class labels $\{y_m : m \in N(v)\}$ are conditionally independent when y_v is given, and $P(y_m = y_v | y_v) = \alpha$, $P(y_m = y | y_v) = \frac{1-\alpha}{|\mathcal{Y}|-1}, \forall y \neq y_v$, we have $P(y_m = i | y_v = i) = \alpha$ and $P(y_m = j | y_v = i) = \frac{1-\alpha}{|\mathcal{Y}|-1}, \forall i, j \in \mathcal{Y}$ and $j \neq i$, where $m \in N(v)$. The 2-hop neighborhood $N_2(v)$ of a node v will be expectedly heterophily-preferred when the following inequality holds:

$$P(y_o = i | y_v = i) - P(y_o \neq i | y_v = i) \leq 0, \ \forall o \in N_2(v) \tag{1}$$

Consider node $o \in N_2(v)$, we have:

$$P(y_o = k | y_v = i) = \sum_{j \in \mathcal{Y}} P(y_o = k | y_m = j) P(y_m = j | y_v = i) \tag{2}$$

Let $\tau = \frac{(1-\alpha)^2}{|\mathcal{Y}|-1}$ and $(1-\alpha)^2 = (|\mathcal{Y}| - 1)\tau$. From Eq. 2, we have:

$$P(y_o = i | y_v = i) = \alpha^2 + \tau \tag{3}$$

and for $j \in \mathcal{Y}$,

$$\sum_{j \neq i} P(y_o = j | y_v = i) = P(y_o \neq i | y_v = i) = 2\alpha(1-\alpha) + (|\mathcal{Y}| - 2)\tau \tag{4}$$

Hence, defining the difference between $P(y_o = i | y_v = i)$ and $P(y_o \neq i | y_v = i)$ as ϵ, we have:

$$\epsilon = P(y_o = i | y_v = i) - P(y_o \neq i | y_v = i) = \alpha^2 + \tau - 2\alpha(1-\alpha) - (|\mathcal{Y}| - 2)\tau \tag{5}$$

Eq. 5 can be simplified as :

$$\epsilon = (2\alpha^2 - 1) + 2\tau \tag{6}$$

Applying $\frac{(1-\alpha)^2}{|\mathcal{Y}|-1} = \tau$, we have

$$\epsilon = \frac{2\alpha(\alpha|\mathcal{Y}| - 2) + (3 - |\mathcal{Y}|)}{|\mathcal{Y}| - 1} \tag{7}$$

Notice that $|\mathcal{Y}| \geq 3$ for multi-class node classification task, it can easily obtain

$$\epsilon \leq \frac{2\alpha(\alpha|\mathcal{Y}| - 2)}{|\mathcal{Y}| - 1} \leq 0, \text{ if } \alpha \leq \frac{2}{|\mathcal{Y}|} \tag{8}$$

Clearly, the sufficient condition for $P(y_o = i|y_v = i) - P(y_o \neq i|y_v = i) \leq 0$ is $\alpha \leq \frac{2}{|\mathcal{Y}|}$. Thus, the 2-hop neighborhood $N_2(v)$ of a node v will always be expectedly heterophily-preferred if $\alpha \leq \frac{2}{|\mathcal{Y}|}$.

This completes the proof. ☐

According to Eq. 6 and Eq. 7, we also have remarks as follows:

Remark 1. *For a graph $\mathcal{G}$ with a class label set $\mathcal{Y}$, consider the class labels of the neighbors of node v in $\mathcal{G}$, $\{y_m : m \in N(v)\}$, are conditionally independent given y_v, and $P(y_m = y_v|y_v) = h$, $P(y_m = y|y_v) = \frac{1-h}{|\mathcal{Y}|-1}, \forall y \neq y_v$, we have:* ***1)*** *the 2-hop neighborhood $N_2(v)$ of a node v will always be expectedly homophily-preferred if $h \geq \sqrt{\frac{1}{2}}$.* ***2)*** *the 2-hop neighborhood $N_2(v)$ of a node v will always be expectedly homophily-preferred if $\mathcal{Y}$ is a binary class label set.*

A.3 Analysis of some existing GNNs from a Polynomial Filtering Perspective

ChebNet [3]. ChebNet first used polynomials to approximate filters, thus avoiding feature decomposition in spectral convolution. The Chebyshev polynomials $T_k(\cdot)$ is used:

$$\mathbf{z} = \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x} \tag{9}$$

Theoretically, the Chebyshev polynomials could approximate arbitrary filters if the order K is high enough. Experiments on node classification also empirically demonstrate that ChebNet filters more effectively than GCN on non-homophilic graphs.

GCN [6]. Kipf et al. simplifies ChebNet to 1-order polynomial approximation, and set $\theta = \theta_0 = -\theta_1$ [6]. Thus, the filtering operation can be seen as:

$$\mathbf{z} = \sum_{k=0}^{1} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x} = \theta_0(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x} \approx \theta\tilde{\mathbf{A}}\mathbf{x} \tag{10}$$

where $\tilde{A}$ is the symmetric normalized adjacency matrix with self-loops. Although GCN uses the renormalization trick and non-linear activation function, its filtering capability is still limited since the 1-order polynomial can only approximate a finite number of filters. Moreover, the predefined coefficients (θ_0, θ_1) render it equivalent to a low-pass filter. More important, analysis of GCN from the spectral filtering perspective can account for the over-smoothing of the GCN. We can derive from Eq. 10 that the frequency response of GCN layer is $\hat{g}(\lambda) = (1 - \tilde{\lambda)}$, where $\tilde{\lambda)}$ are the eigenvalure of $\tilde{\mathbf{A}}$. Hence, the frequency response of GCN is equivalent to $\hat{g}(\lambda) = (1 - \tilde{\lambda)}^K$ when stacking K such layers. As K tends to infinity, it will approximate an impulse low-pass filter, which means the filtered signal at each node is the same, i.e., over-smoothing. As a simplified version of the GCN, SGC [8] is consistent with it.

APPNP [7]. An improved propagation scheme is derived from PageRank in [7], which is also equivalent to a polynomial filtering operation:

$$\mathbf{z} = (1 - \alpha)^K \tilde{\mathbf{A}}^K \mathbf{x} + \sum_{k=0}^{K-1} \alpha(1 - \alpha)^k \tilde{\mathbf{A}}^k \mathbf{x} \tag{11}$$

where $\alpha \in (0, 1]$ is a hyper-parameter to control the teleport (restart) probability. The second term is K-order monomial polynomial of $\tilde{\mathbf{A}}$ and the coefficient of the k-order is $\alpha(1 - \alpha)^k$. It can be seen that the fixed α restricts the expressibility of APPNP, so that only a specific set of filters can be approximated. Nevertheless, APPNP can approximate more filters than GCN. On the basis of APPNP, GPRGNN [2] and BernNet [5] improve APPNP via trainable parameters to learn the coefficients of the polynomial.

ARMA [1]. A recursive and distributed formulation is used to implement the auto-regressive moving average (ARMA) filter. If we remove the non-linear activation function, the ARMA layer is defined as follows:

$$\mathbf{z} = \frac{1}{B}\sum_{b=1}^{B} \bar{\mathbf{x}}_b^{(K)}, \bar{\mathbf{x}}_b^{(K)} = \mathbf{M}^K \mathbf{x} \mathbf{W}_b^K + \sum_{k=0}^{K} \mathbf{M}^k \mathbf{x} (\mathbf{V}_b \mathbf{W}_b^k) \tag{12}$$

where $\mathbf{M} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, ${\mathbf{W}_b}_{b=1}^{B}$ and ${\mathbf{V}_b}_{b=1}^{B}$ are the trainable weight matrics. It can be seen that the second term is a K-order polynomial of $\mathbf{M}$ and the coefficient of the k-order is integrated into the feature transformation matrix $\mathbf{V}_b \mathbf{W}_b^k$. Thus, ARMA is similar to ChebNet which can approximate arbitrary filters.

A.4 Details of Experimental Setup

The Statistics of Datasets. The statistics of the 10 datasets used in this work are summarized in Table 2.

Table 2: Statistics of the used datasets

	Cora	Cite.	PubMed	Comp.	Photo	Cham.	Squi.	Actor	Texas	Corn.
Nodes	2708	3327	19717	13752	7650	2277	5201	7600	183	183
Edges	5278	4552	44324	245861	119081	31371	198353	26659	279	277
Features	1433	3703	500	767	745	2325	2089	932	1703	1703
Classes	7	6	5	10	8	5	5	5	5	5
$\mathcal{H}_\mathcal{G}$	0.825	0.718	0.792	0.802	0.849	0.247	0.217	0.215	0.057	0.301

Hardwares. The experiments and models are performed on a workstation with an NVIDIA 2080Ti GPU (12GB GPU memory), an Intel Xeon E5-2680 CPU (8 cores), and 100GB memory.

Baseline Implemenation. For GPRGNN, BernNet, GeomGCN, and BMGCN, we directly use the open-source codes released by the original paper. For the others, we use the models that are provided by [2], which are implemented based on Pytorch Geometric library [4]. Specifically, we use 2 GCN layers with 64 hidden units for GCN implementation. We use 2 GAT layers for GAT implementation, where the attention heads are (8, 1), and the number of hidden units of each head is (8, 64). For ChebyNet, each layer is set to 2 propagation steps with 32 hidden units. For APPNP, we use a 2-layer MLP with 64 hidden units for feature transformation and 10 steps for feature propagation. For SGC, we use the default $K = 2$. The MLP in baselines is as same as the 2-layer MLP of APPNP. The URL of the officially released codes are listed as follows:

- **GPRGNN**: `https://github.com/jianhao2016/GPRGNN`
- **BernNet**: `https://github.com/ivam-he/BernNet`
- **GeomGCN**: `https://github.com/graphdml-uiuc-jlu/geom-gcn`
- **BMGCN**: `https://github.com/hedongxiao-tju/BM-GCN`

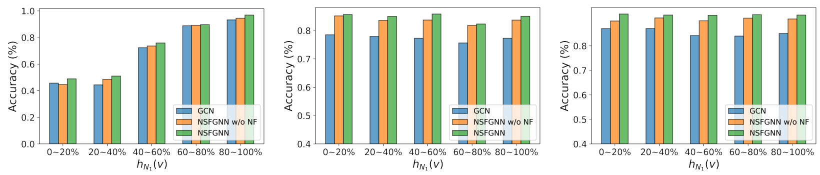
NFGNN Implementation and Hyperparameters Tuning. For our NFGNN, a 2-layer MLP is used for feature transformation, whose dropout rate dp_l is set to 0.5 for all datasets, and hidden units is set to $(f_h, |\mathcal{Y}|)$. Besides, we use lr_l to denote the learning rate of the MLP. lr_p, dp_p are used to denote the learning rate, dropout rate of the node-oriented filtering layer. The Adam optimizer with weight decay L_2 is used to optimize the model. For the parameter initialization of NFGNN, we use PPR [2] to initialize Λ and Xavier initialization for Θ and $\mathbf{W}$. The hyperparameter of PPR is denoted as β. For each dataset, we search the optimal lr_l within $\{0.01, 0.05\}$, lr_p within $\{0.001, 0.005, 0.01\}$, dp_p and β within $\{0, 0.1, 0.2, 0.5, 0.7, 0.8, 0.9\}$, f_h within $\{16, 32, 64\}$, and L_2 within $\{0.0001, 0.0005, 0.001\}$. The hyperparameters of NFGNN for each dataset are summarized in Table.3.

A.5 Additional Experimental results of Node-level Analysis

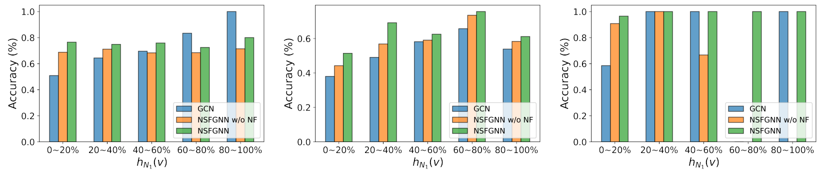
Fig. 1 shows the results of node-level analysis on six datasets. It can be observed that our NFGNN achieves great performance on each interval with high accuracy. Besides, GCN performs poorly for nodes with a low homophilic ratio, which is consistent with the analysis of the GCN.

Table 3: Hyperparameters for NFGNN on real-world datasets.

Datasets	lr_l	dp_l	lr_p	dp_p	β	f_h	L_2	K
Cora	0.01	0.5	0.001	0.9	0.9	32	0.001	10
Citeseer	0.05	0.5	0.001	0	0.9	64	0.001	10
Pubmed	0.05	0.5	0.001	0	0.2	32	0.001	10
Computers	0.05	0.5	0.005	0.7	0.5	64	0	10
Photo	0.05	0.5	0.01	0.8	0.8	64	0	10
Chameleon	0.01	0.5	0.01	0.5	0	64	0.0001	10
Squirrel	0.01	0.5	0.001	0.5	0	64	0	10
Actor	0.05	0.5	0.001	0	0.8	16	0.0005	10
Texas	0.05	0.5	0.001	0.1	0.8	32	0.0001	10
Cornell	0.05	0.5	0.001	0.5	0.9	64	0.001	10



(a) Pubmed (b) Computers (c) Photo



(d) Chameleon (e) Squirrel (f) Texas

Figure 1: Mean classification accuracy of nodes range by homophily ratio $h_{N_1}(v)$ on each dataset.

A.6 Broader Impact

NFGNN is a general technical and theoretical contribution, which extends existing methods of global consistent spectral filtering and adaptive discrimination of the variations of local homophily patterns. We consider the proposed NFGNN might have a positive societal impact since it offers a new possibility for network analysis and node prediction to benefit the applications involving graph data. We hope that this paper could help researchers understand and develop universal GNNs from a spectral domain perspective, thus breaking through the long-held homophily assumptions of GNNs. Besides, there are no foreseeable potential negative social impacts of this paper.

References

[1] 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*, 2021.

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

[3] 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.

[4] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

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

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

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

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