Beyond Low-frequency Information in Graph Convolutional Networks

Deyu Bo[1], **Xiao Wang**[1], **Chuan Shi**[1] *, **Huawei Shen**[2]
[1]Beijing University of Posts and Telecommunications
[2]CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
{bodeyu, xiaowang, shichuan}@bupt.edu.cn, shenhuawei@ict.ac.cn

Abstract

Graph neural networks (GNNs) have been proven to be effective in various network-related tasks. Most existing GNNs usually exploit the low-frequency signals of node features, which gives rise to one fundamental question: is the low-frequency information all we need in the real world applications? In this paper, we first present an experimental investigation assessing the roles of low-frequency and high-frequency signals, where the results clearly show that exploring low-frequency signal only is distant from learning an effective node representation in different scenarios. How can we adaptively learn more information beyond low-frequency information in GNNs? A well-informed answer can help GNNs enhance the adaptability. We tackle this challenge and propose a novel Frequency Adaptation Graph Convolutional Networks (FAGCN) with a self-gating mechanism, which can adaptively integrate different signals in the process of message passing. For a deeper understanding, we theoretically analyze the roles of low-frequency signals and high-frequency signals on learning node representations, which further explains why FAGCN can perform well on different types of networks. Extensive experiments on six real-world networks validate that FAGCN not only alleviates the over-smoothing problem, but also has advantages over the state-of-the-arts.

Introduction

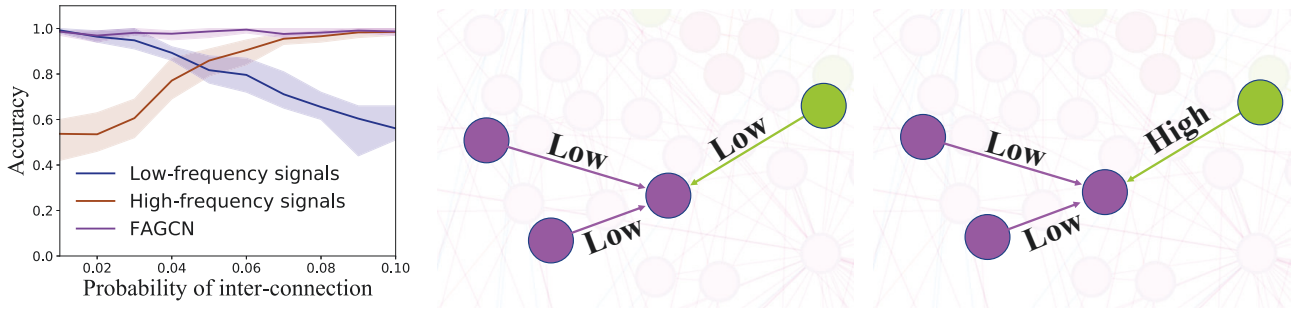
Networks, such as social networks, citation networks and molecular networks, are ubiquitous in the real world. Recently, the emerging graph neural networks (GNNs) have demonstrated powerful ability to learn node representations by jointly encoding network structures and node features (Wu et al. 2020; Zhang, Cui, and Zhu 2020; Wang et al. 2020a). This strategy has been proven to be effective in various tasks, including link prediction (Zhang and Chen 2018), node classification (Kipf and Welling 2017; Velickovic et al. 2018) and graph classification (Errica et al. 2020).

In general, GNNs update node representations by aggregating information from neighbors, which can be seen as a special form of low-pass filter (Wu et al. 2019; Li et al. 2019). Some recent studies (NT and Maehara 2019; Xu et al. 2019a) show that the smoothness of signals, i.e., low-frequency information, are the key to the success of GNNs. However, is the low-frequency information all we need and what roles do other information play in GNNs? This is a fundamental question which motivates us to rethink whether GNNs comprehensively exploit the information in node features when learning node representation.

Firstly, the low-pass filter in GNNs mainly retains the commonality of node features, which inevitably ignores the difference, so that the learned representations of connected nodes become similar. Thanks to the smoothness of low-frequency information, this mechanism may work well for assortative networks, i.e., similar nodes tend to connect with each other (Xu et al. 2019a). However, the real-world networks are not always assortative, but sometimes disassortative, i.e., nodes from different classes tend to connect with each other (Newman 2003). For example, the chemical interactions in proteins often occur between different types of amino acids (Zhu et al. 2020). If we force the representation of connected proteins to be similar by employing low-pass filter, obviously, the performance will be largely hindered. The low-frequency information here is insufficient to support the inference in such networks. Under the circumstances, the high-frequency information, capturing the difference between nodes, may be more suitable. Even the raw features, containing both low- and high-frequency information, are alternative solution (Wang et al. 2020b). Secondly, it is well established that the node representation will becomes indistinguishable when we always utilize low-pass filter, causing over-smoothing (Oono and Suzuki 2020). This reminds us that low-pass filter of current GNNs is distant from optimal for real world scenarios.

To provide more evidence for the above analysis, we focus on low-frequency and high-frequency signals as an example, and present experiments to assess their roles (details can be seen in Section). The results clearly show that both of them are helpful for learning node representations. Specifically, we find that when a network exhibits disassortativity, high-frequency signals perform much better than low-frequency signals. This implies that the high-frequency information, which is largely eliminated by the current GNNs, is not always useless, and the low-frequency information is not always optimal for the complex networks. Once the weakness of low-frequency information in GNNs is identified, a natural question is *how to use signals of different frequencies in GNNs and, at the same time, makes GNNs suitable for different type of networks?*

*Corresponding author



(a) Classification accuracy (b) Existing GNNs (c) FAGCN

Figure 1: (a) Classification accuracy of low-frequency signals, high-frequency signals and our model FAGCN. X-axis denotes probability of inter-connection q. (b) Existing GNNs aggregate the low-frequency signals of neighbors. (c) FAGCN aggregates the low-frequency signals of neighbors within the same class and high-frequency signals of neighbors from different classes, where the color indicates the node label.

To answer this question, two challenges need to be solved: (1) Both the low-frequency and high-frequency signals are the parts of the raw features. Traditional filter is specifically designed for one certain signal, and cannot well extract different frequency signals simultaneously. (2) Even we can extract different information, however, the assortativity of real-world networks is usually agnostic and varies greatly, moreover, the correlation between task and different information is very complex, so it is difficult to decide what kind of signals should be used: raw features, low-frequency signals, high-frequency signals or their combination.

In this paper, we design a general frequency adaptation graph convolutional networks called FAGCN, to adaptively aggregate different signals from neighbors or itself. We first employ the theory of graph signal processing to formally define an enhanced low-pass and high-pass filter to separate the low-frequency and high-frequency signals from the raw features. Then we design a self-gating mechanism to adaptively integrate the low-frequency signals, high-frequency signals and raw features, without knowing the assortativity of network. Theoretical analysis proves that FAGCN is a generalization of most existing GNNs and it has a capability to freely shorten or enlarge the distance between node representations, which further explains why FAGCN can perform well on different types of networks.

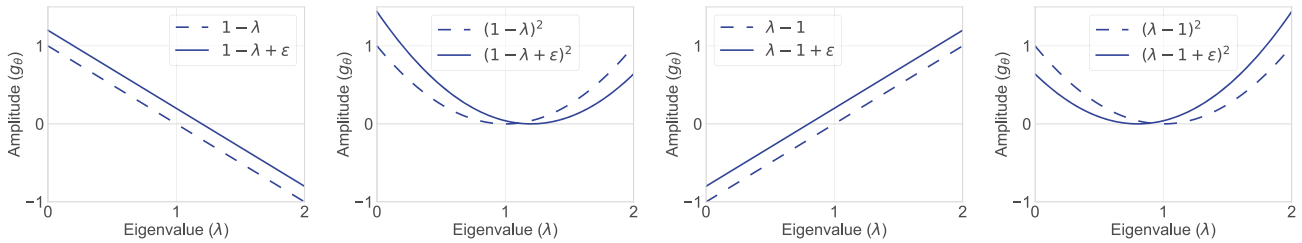
The contribution of this paper is summarized as follows:

- We study the roles of both low-frequency and high-frequency signals in GNNs and verify that high-frequency signals are useful for disassortative networks.
- We propose a novel graph convolutional networks FAGCN, which can adaptively change the proportion of low-frequency and high-frequency signals without knowing the types of networks.
- We theoretically prove that the expressive power of FAGCN is greater than other GNNs. Moreover, our proposed FAGCN is able to alleviate the over-smoothing problem. Extensive experiments on six real-world networks validate that FAGCN has advantages over state-of-the-arts.

An Experimental Investigation

In this section, taking the low-frequency and high-frequency signals as an example, we analyze their roles in learning node representations. Specifically, we test their performance of node classification on a series of synthetic networks. The main idea is to gradually increase the disassortativity of the synthetic networks, and observe how the performance of these two signals changes. We generate a network with 200 nodes and randomly divide them into 2 classes. For each node in class one, we sample a 20-dimensional feature vector from Gaussian distribution $\mathcal{N}(0.5, 1)$, while for the nodes in class two, the distribution is $\mathcal{N}(-0.5, 1)$. Besides, the connections in the same class are generated from a Bernoulli distribution with probability $p = 0.05$, and the probability of connections between two classes q varies from 0.01 to 0.1. When q is small, the network exhibits assortativity; As q increases, the network gradually exhibits disassortativity. We then apply the low-pass and high-pass filters, described in Section , to node classification task. Half of the nodes are used for training and the remains are used for testing.

Figure 1(a) illustrates that with the increase of inter-connection q, the accuracy of low-frequency signals decreases, while the accuracy of high-frequency signals increases gradually. This proves that both the low-frequency and high-frequency signals are helpful in learning node representations. The reason why existing GNNs fail when q increases is that, as shown in Figure 1(b), they only aggregate low-frequency signals from neighbors, i.e., making the node representations become similar, regardless of whether nodes belong to the same class, thereby losing the discrimination. When the network becomes disassortative, the effectiveness of high-frequency signals appears, but as shown in Figure 1(a), a single filter cannot achieve optimal results in all cases. Our proposed FAGCN, which combines the advantages of both low-pass and high-pass filters, can aggregate the low-frequency signals of neighbors within the same class and high-frequency signals of neighbors from different classes, as shown in Figure 1(c), thereby showing the best performance on every synthetic network.



(a) $\mathcal{F}_L$ (b) $\mathcal{F}_L^2$ (c) $\mathcal{F}_H$ (d) $\mathcal{F}_H^2$

Figure 2: The relations between eigenvalues and amplitudes in different filters.

Our Proposed Model: FAGCN

Consider an undirected graph $G = (V, E)$ with adjacency matrix $A \in \mathbb{R}^{N\times N}$, where V is a set of nodes with $|V| = N$ and E is a set of edges. The normalized graph Laplacian matrix is defined as $L = I_n - D^{-1/2}AD^{-1/2}$, where $D \in \mathbb{R}^{N\times N}$ is a diagonal degree matrix with $D_{i,i} = \sum_j A_{i,j}$ and I_n denotes the identity matrix. Because L is a real symmetric matrix, it has a complete set of orthonormal eigenvectors $\{u_l\}_{l=1}^n \in \mathbb{R}^n$, each of which has a corresponding eigenvalue $\lambda_l \in [0, 2]$ (Chung and Graham 1997). Through the eigenvalues and eigenvectors, we have $L = U\Lambda U^\top$, where $\Lambda = diag([\lambda_1, \lambda_2, \cdots, \lambda_n])$.

Graph Fourier Transform. According to theory of graph signal processing (Shuman et al. 2013), we can treat the eigenvectors of normalized Laplacian matrix as the bases in graph Fourier transform. Given a signal $x \in \mathbb{R}^n$, the graph Fourier transform is defined as $\hat{x} = U^\top x$, and the inverse graph Fourier transform is $x = U\hat{x}$. Thus, the convolutional $*_G$ between the signal x and convolution kernel f is:

$$f *_G x = U\left(\left(U^\top f\right) \odot \left(U^\top x\right)\right) = Ug_\theta U^\top x, \quad (1)$$

where $\odot$ denotes the element-wise product of vectors and g_θ is a diagonal matrix, which represents the convolutional kernel in the spectral domain, replacing $U^\top f$. Spectral CNN (Bruna et al. 2014) uses a non-parametric convolutional kernel $g_\theta = diag(\{\theta_i\}_{i=1}^n)$. ChebNet (Defferrard, Bresson, and Vandergheynst 2016) parameterizes convolutional kernel with a polynomial expansion $g_\theta = \sum_{k=0}^{K-1} \alpha_k \Lambda^k$. GCN defines the convolutional kernel as $g_\theta = I - \Lambda$.

Separation

As discussed in Section , both the low-frequency and high-frequency signals are helpful for learning node representations. To make full use of them, we design a low-pass filter $\mathcal{F}_L$ and a high-pass filter $\mathcal{F}_H$ to separate the low-frequency and high-frequency signals from the node features:

$$\mathcal{F}_L = \varepsilon I + D^{-1/2}AD^{-1/2} = (\varepsilon + 1)I - L,$$
$$\mathcal{F}_H = \varepsilon I - D^{-1/2}AD^{-1/2} = (\varepsilon - 1)I + L, \quad (2)$$

where ε is a scaling hyper-parameter limited in $[0, 1]$. If we use $\mathcal{F}_L$ and $\mathcal{F}_H$ to replace the convolutional kernel f in Equation 1. The signal x is filtered by $\mathcal{F}_L$ and $\mathcal{F}_H$ as:

$$\mathcal{F}_L *_G x = U[(\varepsilon + 1)I - \Lambda]U^\top x = \mathcal{F}_L \cdot x,$$
$$\mathcal{F}_H *_G x = U[(\varepsilon - 1)I + \Lambda]U^\top x = \mathcal{F}_H \cdot x. \quad (3)$$

Therefore, the convolutional kernel of $\mathcal{F}_L$ is $g_\theta = (\varepsilon+1)I - \Lambda$, rewritten as $g_\theta(\lambda_i) = \varepsilon + 1 - \lambda_i$, shown in Figure 2(a). When $\lambda_i > 1 + \varepsilon$, $g_\theta(\lambda_i) < 0$, which gives a negative amplitude. To avoid this, we consider the second-order convolution kernel $\mathcal{F}_L^2$ with $g_\theta(\lambda_i) = (\varepsilon+1-\lambda_i)^2$, shown in Figure 2(b). When $\lambda_i = 0$, $g_\theta(\lambda_i) = (\varepsilon + 1)^2 > 1$ and when $\lambda_i = 2$, $g_\theta(\lambda_i) = (\varepsilon - 1)^2 < 1$, which amplifies the low-frequency signals and restrains the high-frequency signals.

Remark 1. *(Enhanced filters) As in Figure 2, compared with traditional low-pass filters, e.g., GCN and SGC (Wu et al. 2019), $\mathcal{F}_L$ is an enhanced low-pass filter. Convolutional kernel of second-order GCN is $g_\theta(\lambda_i) = (1 - \lambda_i)^2$. When $\lambda_i = 0$, the amplitude of GCN is $g_\theta(\lambda_i) = 1 < (1 + \varepsilon)^2$. Hence, the value of $\mathcal{F}_L$ is greater than GCN in low-pass filtering. Similarly, $\mathcal{F}_H$ is an enhanced high-pass filter, which provides a greater value for the high-frequency signals.*

Separating the low-frequency and high-frequency signals from the node features provides a feasible way to deal with different networks, e.g., low-frequency signals for assortative networks and high-frequency signals for disassortative networks. However, this way has two disadvantages: One is that selecting signals requires a priori knowledge, i.e., we actually do not know whether a network is assortative or disassortative beforehand. The other is that, as in Equation 3, it requires matrix multiplication, which is undesirable for large graphs (Hamilton, Ying, and Leskovec 2017). Therefore, an efficient method that can adaptively aggregate low-frequency and high-frequency signals is desired.

Remark 2. *(Concrete meaning of signals) In Equation 2, we have $\mathcal{F}_L = \varepsilon I + D^{-1/2}AD^{-1/2}$ and $\mathcal{F}_H = \varepsilon I - D^{-1/2}AD^{-1/2}$. Therefore, the concrete meaning of low-frequency signal $\mathcal{F}_L \cdot x$ is the sum of node features and neighborhood features in spatial domain, while high-frequency signal $\mathcal{F}_H \cdot x$ represents the difference between node features and neighborhood features in spatial domain.*

Aggregation

Before introducing the details, we first compare the aggregation process of existing GNNs and FAGCN in Figure 3. The left shows that existing GNNs consider learning the importance (α_{ij}) of each node in aggregation. The right is FAGCN that uses two coefficients (α_{ij}^L and α_{ij}^H) to aggregate low-frequency and high-frequency signals from the neighbors, respectively.

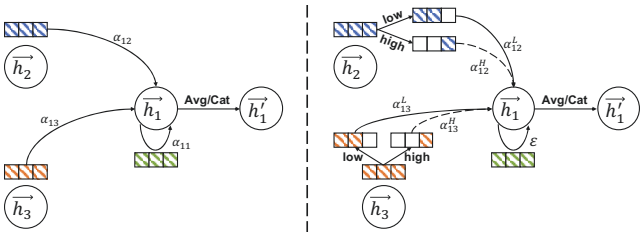


Figure 3: Left: The aggregation process of existing GNNs, and α_{ij} indicates the importance of node j to node i. Right: The aggregation process of FAGCN, and α_{ij}^L, α_{ij}^H denote the proportions of low-frequency and high-frequency signals of node j to node i, respectively.

The input of our model are the node features, $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N\} \in \mathbb{R}^{N\times F}$, where F is the dimension of the node features. For the purpose of frequency adaptation, a basic idea is to use the attention mechanism to learn the proportion of low-frequency and high-frequency signals:

$$\tilde{\mathbf{h}}_i = \alpha_{ij}^L(\mathcal{F}_L\cdot\mathbf{H})_i + \alpha_{ij}^H(\mathcal{F}_H\cdot\mathbf{H})_i = \varepsilon\mathbf{h}_i + \sum_{j\in\mathcal{N}_i}\frac{\alpha_{ij}^L - \alpha_{ij}^H}{\sqrt{d_i d_j}}\mathbf{h}_j, \tag{4}$$

where $\tilde{\mathbf{h}}_i$ is the aggregated representation of node i. $\mathcal{N}_i$ and d_i denote the neighbor set and degree of node i, respectively. α_{ij}^L and α_{ij}^H represent the proportions of node j's low-frequency and high-frequency signals to node i. We set $\alpha_{ij}^L + \alpha_{ij}^H = 1$ and $\alpha_{ij}^G = \alpha_{ij}^L - \alpha_{ij}^H$. In the following, we show that α_{ij}^G can be interpreted from two perspectives.

Remark 3. *(Two perspectives of α_{ij}^G) One is that α_{ij}^G indirectly represents the proportion of low-frequency and high-frequency signals. $\alpha_{ij}^G > 0$, i.e., $\alpha_{ij}^L > \alpha_{ij}^H$, means that low-frequency signals dominate the representations and vice versa. Based on α_{ij}^G, we can calculate the value of α_{ij}^L and α_{ij}^H, so as to achieve the proportions of signals. Another is that α_{ij}^G denotes the coefficients of neighbors in aggregation. $\alpha_{ij}^G > 0$ represents the sum of node features and neighborhood features, i.e., $\mathbf{h}_i + \mathbf{h}_j$, while $\alpha_{ij}^G < 0$ represents the difference between them, i.e., $\mathbf{h}_i - \mathbf{h}_j$, as explained in Remark 2. Besides, when $\alpha_{ij}^G \approx 0$, the contributions of neighbors will be limited, so the raw features will dominate the node representations.*

In order to learn the coefficients α_{ij}^G effectively, we need to consider the features of both the node itself and its neighbors. Therefore, we propose a shared *self-gating* mechanism $\mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$ to learn the coefficients:

$$\alpha_{ij}^G = \tanh\left(\mathbf{g}^\top[\mathbf{h}_i \parallel \mathbf{h}_j]\right), \tag{5}$$

where $\parallel$ denotes the concatenation operation, $\mathbf{g} \in \mathbb{R}^{2F}$ can be seen as a shared convolutional kernel (Velickovic et al. 2018) and $\tanh(\cdot)$ is the hyperbolic tangent function, which can naturally limits the value of α_{ij}^G in $[-1, 1]$. Besides, to make use of the structural information, we only calculate the coefficients among the node and its first-order neighbors $\mathcal{N}_i$.

After calculating α_{ij}^G, we can aggregate the representations of neighbors:

$$\mathbf{h}_i^{'} = \varepsilon\mathbf{h}_i + \sum_{j\in\mathcal{N}_i}\frac{\alpha_{ij}^G}{\sqrt{d_i d_j}}\mathbf{h}_j, \tag{6}$$

where $\mathbf{h}_i^{'}$ denotes the aggregated representation of node i. Note that when aggregating information from neighbors, the degrees are used to normalize the coefficients, thus preventing the aggregated representations from being too large.

The Whole Architecture of FAGCN

In the previous section, we introduce the message passing process of FAGCN. Here, we formally define the whole architecture of FAGCN. Some recent studies (Wu et al. 2019; Cui et al. 2020) emphasize that the entanglement of filters and weight matrices may be harmful to the performance and robustness of the model. Motivated by this, we first use a multilayer perceptron (MLP) to apply the non-linear transform to the raw features. Then we propagate the representations through Eq. 6. The mathematical expression of FAGCN is defined as:

$$\begin{aligned}
\mathbf{h}_i^{(0)} &= \phi(\mathbf{W}_1\mathbf{h}_i) && \in \mathbb{R}^{F'\times 1}\\
\mathbf{h}_i^{(l)} &= \varepsilon\mathbf{h}_i^{(0)} + \sum_{j\in\mathcal{N}_i}\frac{\alpha_{ij}^G}{\sqrt{d_i d_j}}\mathbf{h}_j^{(l-1)} && \in \mathbb{R}^{F'\times 1}\\
\mathbf{h}_{out} &= \mathbf{W}_2\mathbf{h}_i^{(L)} && \in \mathbb{R}^{K\times 1},
\end{aligned} \tag{7}$$

where $\mathbf{W}_1 \in \mathbb{R}^{F\times F'}$, $\mathbf{W}_2 \in \mathbb{R}^{F'\times K}$ are the weight matrices, ϕ is the activation function, F' denotes the dimension of hidden layers, l indicates the layers, ranging from 1 to L, and K represents the number of classes. The complexity of a single layer FAGCN is $\mathcal{O}((N + |E|) \times F')$, which is approximately linear with the number of edges and nodes.

Theoretical Analysis

Connection to Existing GNNs

FAGCN is a generalization of most existing GNNs. Specifically, when we set the coefficients α_{ij}^G to 1, FAGCN acts like GCN and when we use softmax function to normalize α_{ij}^G, FAGCN becomes GAT. Therefore, as indicated in Remark 2, because the coefficients in GCN and GAT are both greater than zero, they prefer to aggregate the low-frequency signals. However, FAGCN can learn a coefficient that can be positive or negative, to adaptively aggregate low-frequency and high-frequency signals.

Expressive Power of FAGCN

We analyze the expressive power of FAGCN from the perspective of the distance between node representations. Assume that (u, v) is a pair of connected nodes, and $\mathbf{h}_u, \mathbf{h}_v$ are the node features. Let $\mathcal{D}, \mathcal{D}_L, \mathcal{D}_H$ be the distance of node features, low-frequency signals of node features and high-frequency signals of node features, respectively.

$$\begin{aligned}
\mathcal{D} &= \|\mathbf{h}_u - \mathbf{h}_v\|_2.\\
\mathcal{D}_L &= \|(\varepsilon\mathbf{h}_u + \mathbf{h}_v) - (\varepsilon\mathbf{h}_v + \mathbf{h}_u)\|_2 = |1-\varepsilon|\mathcal{D}.\\
\mathcal{D}_H &= \|(\varepsilon\mathbf{h}_u - \mathbf{h}_v) - (\varepsilon\mathbf{h}_v - \mathbf{h}_u)\|_2 = |1+\varepsilon|\mathcal{D}.
\end{aligned}$$

Dataset	Assortivity	Nodes	Edges	Classes	Features
Cora	0.771	2,708	5,429	7	1,433
Citeseer	0.671	3,327	4,732	6	3,703
Pubmed	0.686	19,717	44,338	3	500
Chameleon	0.180	2,277	36,101	3	2,325
Squirrel	0.018	5,201	217,073	3	2,089
Actor	0.003	7,600	33,544	5	931

Table 1: The statistics of datasets

Proposition 1. *Low-pass filtering makes the representations become similar, while high-pass filtering makes the representations become discriminative.*

Proof. It is easy to see that $\mathcal{D}_H > \mathcal{D} > \mathcal{D}_L$. This indicates that compared with the original distance $\mathcal{D}$, the distance $\mathcal{D}_L$ induced by low-frequency signals is smaller, implying that low-frequency signals can make the representations of connected nodes become similar. While the distance $\mathcal{D}_H$ induced by high-frequency signals is larger, implying that high-frequency signals can make the representations of connected nodes become discriminative. □

We have analyzed the roles of low-frequency and high-frequency signals in representation learning. Obviously, FAGCN can choose to shorten or enlarge the distance between node representations flexibly, while most existing GNNs cannot.

Proposition 2. *Most existing GNNs, e.g., GCN, only have the capability to make representations of nodes become similar.*

Proof. The filter used in GCN is: $(D+I)^{-1/2}(A+I)(D+I)^{-1/2}$. Hence, the distance of representations learned by GCN is: $\mathcal{D}_G \approx \|(\frac{1}{d_u}\mathbf{h}_u+\mathbf{h}_v)-(\frac{1}{d_v}\mathbf{h}_v+\mathbf{h}_u)\|_2 \approx |1-\frac{1}{d}|\mathcal{D} < \mathcal{D}$ (s.t. $d_u \approx d_v \approx d$). □

Experiments

Datasets

Assortative datasets. We choose the commonly used *citation networks*, e.g., Cora, Citeseer and Pubmed for assortative datasets. Edges in these networks represent the citation relationship between two papers (undirected), node features are the bag-of-words vector of the papers and labels are the fields of papers. In each network, we use 20 labeled nodes per class for training, 500 nodes for validation and 1000 nodes for testing. Details can be found in (Kipf and Welling 2017).
Disassortative datasets. We consider the *Wikipedia networks*[1] and *Actor co-occurrence network* (Tang et al. 2009) for disassortative datasets. Chameleon and Squirrel are two Wikipedia networks. Edges represent the hyperlinks between two pages, node features are some informative nouns in the pages and labels correspond to the traffic of the pages. In Actor co-occurrence network, each node represents an actor, and the edges denote the collaborations of them. Node features are the keywords in Wikipedia and labels are the types of actors. Since there is no standard division for these networks. To verify the effectiveness and robustness, we use 20% for validation, 20% for testing and change the training ratio from 10% to 60%.

More detailed characteristics of the datasets can be found in Table 1. Note that a higher value of the second column represents a more obvious assortativity (Newman 2003).

[1] http://snap.stanford.edu/data/wikipedia-article-networks.html

Method	Cora	Citeseer	Pubmed
SGC	81.0%	71.9%	78.9%
GCN	81.5%	70.3%	79.0%
GWNN	82.8%	71.7%	79.1%
ChebNet	81.2%	69.8%	74.4%
GraphHeat	83.7%	72.5%	**80.5%**
GIN	77.6%	66.1%	77.0%
GAT	83.0%	72.5%	79.0%
MoNet	81.7%	-	78.8%
APPNP	83.7%	72.1%	79.2%
GraphSAGE	82.3%	71.2%	78.5%
FAGCN	**84.1±0.5%**	**72.7±0.8%**	79.4±0.3%

Table 2: Summary of node classification results (in percent).

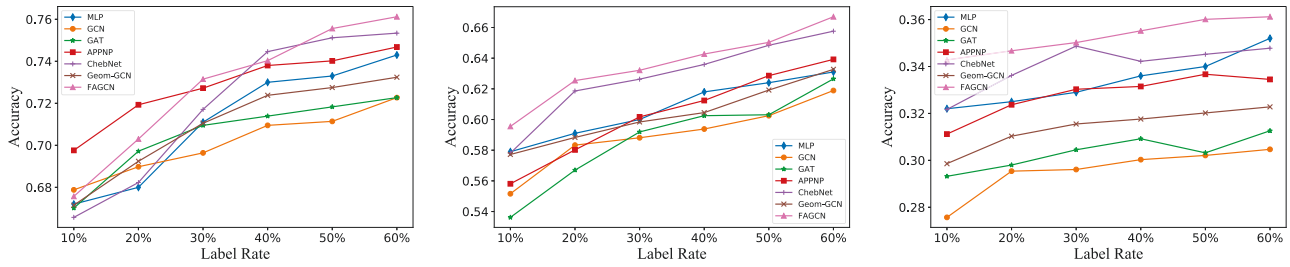
Experimental Setup

We compare FAGCN with two types of representative GNNs: Spectral-based methods, i.e., SGC (Wu et al. 2019), GCN (Kipf and Welling 2017), ChebNet (Defferrard, Bresson, and Vandergheynst 2016) and GWNN (Xu et al. 2019b); Spatial-based methods, i.e., GIN (Xu et al. 2019c), GAT (Velickovic et al. 2018), MoNet (Monti et al. 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017) and APPNP (Klicpera, Bojchevski, and Günnemann 2019). For disassortative networks, we add Geom-GCN (Pei et al. 2020) and MLP as new benchmarks. All methods were implemented in Pytorch with Adam optimizer (Kingma and Ba 2015). We run 10 times and report the mean values with standard deviation. The hidden unit is fixed at 16 in assortative networks and 32 in disassortative networks. The hyper-parameter search space is: learning rate in $\{0.01, 0.005\}$, dropout in $\{0.4, 0.5, 0.6\}$, weight decay in $\{1E\text{-}3, 5E\text{-}4, 5E\text{-}5\}$, number of layers in $\{1, 2, \cdots, 8\}$, ε in $\{0.1, \cdots, 1.0\}$.

In assortative datasets, we use hyper-parameters in previous literature for baselines. For FAGCN, the hyper-parameter setting is: learning rate = 0.01, dropout = 0.6, weight decay = $1E$-3, layers = 4. ε = 0.2, 0.3, 0.3 for Cora, Citeseer and Pubmed. The patience of early stop is set to 100. In disassortative datasets, the hyper-parameter for FAGCN is: learning rate = 0.01, dropout = 0.5, weight decay = $5E$-5, layers = 2. ε = 0.4, 0.3, 0.5 for Chameleon, Squirrel and Actor, respectively. Besides, we run 500 epochs and choose the model with highest validation accuracy for testing.

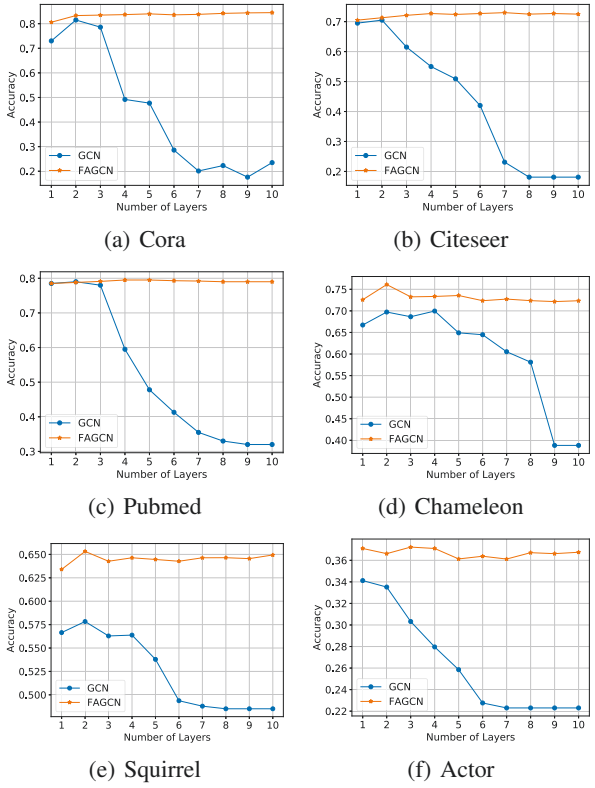
Classification on Different Types of Networks

The performance of different methods on assortative networks is summarized in Table 2. GraphHeat designs a low-pass filer through heat kernel, which can better capture the low-frequency information than GCN (Xu et al. 2019a).



(a) Chameleon (b) Squirrel (c) Actor

Figure 4: Classification accuracy of different methods under different label rates on disassortative networks.



(a) Cora (b) Citeseer

(c) Pubmed (d) Chameleon

(e) Squirrel (f) Actor

Figure 5: Classification accuracy with different model depth.

Therefore, it performs best in the baselines. But we can see that FAGCN exceed the benchmarks on most networks due to the enhanced low-pass filter, which validates the importance of low-pass filters in the assortative networks.

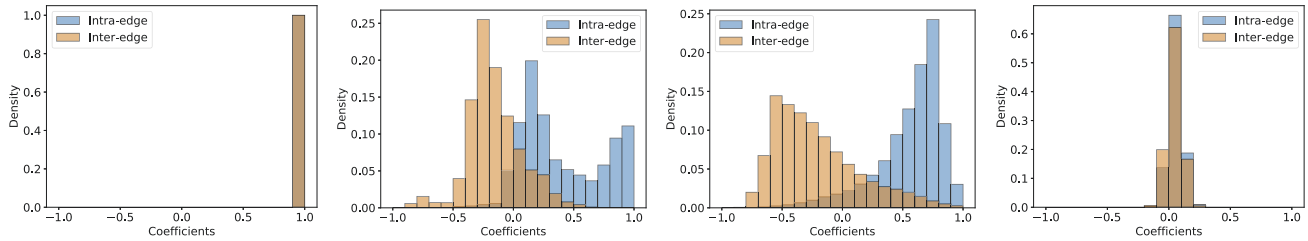
Besides, the performance on disassortative networks is illustrated in Fig. 4. Note that we do not choose all baselines because the methods focus on low-pass filtering have poor performance, and we use GCN and GAT as representatives. In addition, APPNP leverages residual connection to preserve the information of raw features, ChebNet uses ChebNet polynomials to approximate arbitrary convolution kernels and Geom-GCN is the state-of-the-art on disassortative networks. Therefore, comparing FAGCN with these baselines can reflect the superiority of FAGCN. From Fig. 4, we can see that GCN and GAT perform worse than other methods, which indicates that only using low-pass filters is not suitable for disassortative networks. APPNP and ChebNet perform better than GCN and GAT, which shows that the raw features and polynomials can preserve the high-frequency information to some extent. Finally, FAGCN performs best in most datasets and label rates, which reflects the superiority of our method.

Alleviating Over-smoothing Problem

To validate whether FAGCN can alleviate the over-smoothing problem, we compare the performance of GCN and FAGCN under different model depth. The results are shown in Fig. 5. It can be seen that GCN achieves the best performance at two layers. As the number of layers increases, the performance of GCN drops rapidly, which indicates that GCN suffers from over-smoothing seriously. Instead, the results of FAGCN are stable and significantly higher than GCN on different types of networks. The reasons are two-folds: one is that in Section we show that negative weights can prevents node representations from being too similar, which benefits deeper network architecture. Another is that we add the raw features, containing both low-frequency and high-frequency information, to each layer, which further keeps node representations from becoming indistinguishable. Through these two designs, when the model going deep, the performance of FAGCN is significantly better than GCN, which indicates that FAGCN has a good capability to alleviate over-smoothing.

Visualization of Edge Coefficients

In order to verify whether FAGCN can learn different edge coefficients to adapt to different networks, we visualize the coefficients α_{ij}^{G}, extracted from the last layer of FAGCN. Specifically, we divide the edges into intra-edges and inter-edges based on whether two connected nodes have the same label. It can be seen from Fig. 6(a) that in the networks with large assortativity, i.e., Cora, Citeseer and Pubmed, all edges are concentrated at the positive weights, which implies that the low-pass filter plays a major role in classification. However, in Fig. 6(b) and 6(c), a lot of inter-edges are distributed in negative weights, which shows that in the network with small assortativity, the high-frequency signal plays an important role in node classification. Moreover, there is an interesting phenomenon that in Fig. 6(d) the coefficients of



(a) Cora, Citeseer and Pubmed (b) Chameleon (c) Squirrel (d) Actor

Figure 6: Visualization of edge coefficients on different networks.

Method	This paper		(Pei et al. 2020)	
	Cham-3	**Squi-3**	**Cham-5**	**Squi-5**
FAGCN	**76.1%**	**66.7%**	**61.7%**	**39.7%**
Geom-GCN	73.2%	63.3%	60.9%	38.1%
GCN	72.3%	61.9%	**59.8%**	**36.9%**
MLP	**74.3%**	**63.1%**	46.4%	29.7%

Table 3: Classification accuracy with different label division.

edges are concentrated at zero. One possible reason is that the assortativity of Actor is quite small, which implies that the structures contributes less to the results of node classification, instead, the raw features dominate the classification results.

Details of Wikipedia Networks

In this section, we aim to give more details of Wikipedia networks. First of all, Chameleon and Squirrel were originally collected for regression task, i.e., traffic prediction (Rozemberczki, Allen, and Sarkar 2019). We divide the traffic into three categories: *less than 1000*, *between 1000 and 10000* and *more than 10000*, so that they can be applied to node classification task. Secondly, labels of Chameleon and Squirrel are different from those in Geom-GCN. The reason is that in the disassortative networks provided by (Pei et al. 2020), i.e., Cham-5 and Squi-5 in Table 3, we find that GCN performs much better than MLP. This is a strange phenomenon, because MLP uses raw features as input, which contains high-frequency information, so its performance should be better than GCN (Zhu et al. 2020). Therefore, we redivide the labels based on traffic, i.e., Cham-3 and Squi-3 in Table 3, where the performance of GCN and MLP is more reasonable. Besides, we can see that FAGCN perform best on all four datasets, so its effectiveness is still guaranteed across different datasets.

Related Work

Spectral Graph Neural Networks. Spectral GNNs aim to define the convolution kernel in spectral domain, by leveraging the theory of graph signal processing. Spectral CNN (Bruna et al. 2014) treats the convolution kernel as a trainable diagonal matrix and directly learns the amplitudes of signals. However, it requires the decomposition of Laplacian matrix, which is inefficient. To deal with this problem, ChebNet (Defferrard, Bresson, and Vandergheynst 2016) uses the polynomial of Laplacian matrix to approximate the convolution kernel and make better performance. GCN (Kipf and Welling 2017) is the first-order approximation of ChebNet with a self-loop mechanism. GraphHeat(Xu et al. 2019a) designs a more powerful low-pass filter through heat kernel. Besides, GWNN (Xu et al. 2019b) replaces the eigenvectors with wavelet bases so as to further improve the efficiency of the model. Generally, spectral methods have good interpretability for the signal processing on graph, but lack generalization (Hamilton, Ying, and Leskovec 2017).

Spatial Graph Neural Networks. Spatial GNNs focus on the design of aggregation function. GraphSAGE (Hamilton, Ying, and Leskovec 2017) designs a permutation-invariant aggregator for message passing; GAT (Velickovic et al. 2018) employs self-attention to calculate the coefficients of neighbors in aggregation; MoNet (Monti et al. 2017) provides a unified generalization of graph convolutional architectures in spatial domain; PPNP (Klicpera, Bojchevski, and Günnemann 2019) incorporates personalized PageRank to the aggregation function; Geom-GCN (Pei et al. 2020) utilizes the structural similarity to capture the long-range dependencies in disassortative graphs. H2GNN (Zhu et al. 2020) separates the raw features and aggregated features so as to preserve both high-frequency and low-frequency information, but it lacks adaptability; Non-Local GNN (Liu, Wang, and Ji 2020) designs an attention-guided sorting mechanism to transform the disassortative networks into assortative networks, which costs a lot of computations. Generally, spatial methods are more flexible and scalable, but lack interpretability. It is worth noting that FAGCN is a spatial method, but it still has good interpretability, which combines the advantages of both spectral and spatial methods.

Conclusion

In this paper, we make the attempt to study the roles of low-frequency and high-frequency signals in GNNs and show that both of them are helpful in learning node representations. Based on this observation, we design a novel frequency adaptation graph convolutional network to adaptively combine the low-frequency and high-frequency signals. Theoretical analysis shows that the expressive power of our model is greater than most existing GNNs. An important direction of future work is to use more signals with different frequencies, e.g., the intermediate frequency signals.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. U1936220, U1936104, 61772082, 61702296, 62002029, 61972442), Meituan-Dianping Group and BUPT Excellent Ph.D. Students Foundation (No. CX2020115). Huawei Shen is funded by Beijing Academy of Artificial Intelligence (BAAI).

References

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

Chung, F. R.; and Graham, F. C. 1997. *Spectral graph theory*. 92. American Mathematical Soc.

Cui, G.; Zhou, J.; Yang, C.; and Liu, Z. 2020. Adaptive Graph Encoder for Attributed Graph Embedding. In *KDD*, 976–985. ACM.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*, 3837–3845.

Errica, F.; Podda, M.; Bacciu, D.; and Micheli, A. 2020. A Fair Comparison of Graph Neural Networks for Graph Classification. In *ICLR*. OpenReview.net.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 1024–1034.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*. OpenReview.net.

Li, Q.; Wu, X.; Liu, H.; Zhang, X.; and Guan, Z. 2019. Label Efficient Semi-Supervised Learning via Graph Filtering. In *CVPR*, 9582–9591. Computer Vision Foundation / IEEE.

Liu, M.; Wang, Z.; and Ji, S. 2020. Non-Local Graph Neural Networks. *CoRR* abs/2005.14612.

Monti, F.; Boscaini, D.; Masci, J.; Rodolà, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*, 5425–5434. IEEE Computer Society.

Newman, M. E. 2003. Mixing patterns in networks. *Physical Review E* 67(2): 026126.

NT, H.; and Maehara, T. 2019. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters. *CoRR* abs/1905.09550.

Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*. OpenReview.net.

Pei, H.; Wei, B.; Chang, K. C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*. OpenReview.net.

Rozemberczki, B.; Allen, C.; and Sarkar, R. 2019. Multi-scale Attributed Node Embedding. *CoRR* abs/1909.13021.

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

Tang, J.; Sun, J.; Wang, C.; and Yang, Z. 2009. Social influence analysis in large-scale networks. In *KDD*, 807–816. ACM.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*. OpenReview.net.

Wang, X.; Bo, D.; Shi, C.; Fan, S.; Ye, Y.; and Yu, P. S. 2020a. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *CoRR* abs/2011.14867.

Wang, X.; Zhu, M.; Bo, D.; Cui, P.; Shi, C.; and Pei, J. 2020b. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In *KDD*, 1243–1253. ACM.

Wu, F.; Jr., A. H. S.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, 6861–6871. PMLR.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* .

Xu, B.; Shen, H.; Cao, Q.; Cen, K.; and Cheng, X. 2019a. Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning. In *IJCAI*, 1928–1934. ijcai.org.

Xu, B.; Shen, H.; Cao, Q.; Qiu, Y.; and Cheng, X. 2019b. Graph Wavelet Neural Network. In *ICLR*. OpenReview.net.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019c. How Powerful are Graph Neural Networks? In *ICLR*. OpenReview.net.

Zhang, M.; and Chen, Y. 2018. Link Prediction Based on Graph Neural Networks. In *NeurIPS*, 5171–5181.

Zhang, Z.; Cui, P.; and Zhu, W. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* .

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Advances in Neural Information Processing Systems* 33.