Interpreting and Unifying Graph Neural Networks with An Optimization Framework

Meiqi Zhu
Beijing University of Posts and Telecommunications
zhumeiqi@bupt.edu.cn

Xiao Wang*
Beijing University of Posts and Telecommunications
xiaowang@bupt.edu.cn

Chuan Shi*
Beijing University of Posts and Telecommunications
shichuan@bupt.edu.cn

Houye Ji
Beijing University of Posts and Telecommunications
jhy1993@bupt.edu.cn

Peng Cui
Tsinghua University
cuip@tsinghua.edu.cn

ABSTRACT

Graph Neural Networks (GNNs) have received considerable attention on graph-structured data learning for a wide variety of tasks. The well-designed propagation mechanism which has been demonstrated effective is the most fundamental part of GNNs. Although most of GNNs basically follow a message passing manner, litter effort has been made to discover and analyze their essential relations. In this paper, we establish a surprising connection between different propagation mechanisms with a unified optimization problem, showing that despite the proliferation of various GNNs, in fact, their proposed propagation mechanisms are the optimal solution optimizing a feature fitting function over a wide class of graph kernels with a graph regularization term. Our proposed unified optimization framework, summarizing the commonalities between several of the most representative GNNs, not only provides a macroscopic view on surveying the relations between different GNNs, but also further opens up new opportunities for flexibly designing new GNNs. With the proposed framework, we discover that existing works usually utilize naïve graph convolutional kernels for feature fitting function, and we further develop two novel objective functions considering adjustable graph kernels showing low-pass or high-pass filtering capabilities respectively. Moreover, we provide the convergence proofs and expressive power comparisons for the proposed models. Extensive experiments on benchmark datasets clearly show that the proposed GNNs not only outperform the state-of-the-art methods but also have good ability to alleviate over-smoothing, and further verify the feasibility for designing GNNs with our unified optimization framework.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Networks** → **Network algorithms**.

*Corresponding authors

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449953

KEYWORDS

Graph neural networks, network representation learning, deep learning

ACM Reference Format:
Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449953

1 INTRODUCTION

Network is a ubiquitous structure for real-world data, such as social networks, citation networks and financial networks. Recently, Graph Neural Networks (GNNs) have gained great popularity in tackling the analytics tasks [6, 13, 31] on network-structured data. Moreover, GNNs have also been successfully applied to a wide range of application tasks, including recommendation [7, 28], natural language processing [8, 47] and computer vision [27, 29].

The classical graph neural networks can be generally divided into two types: spectral-based GNNs and spatial-based GNNs. Spectral-based methods [5, 36] mainly focus on defining spectral graph filters via graph convolution theorem; spatial-based methods [9, 37] usually follow a message passing manner, where the most essential part is the feature propagation process along network topology. To date, many representative GNNs have been proposed by designing different feature propagation mechanisms, e.g., attention mechanism [30], personalized pagerank [14] and jump connection [38]. The well-designed propagation mechanism which has been demonstrated effective is the most fundamental part of GNNs. Although there are various propagation mechanisms, they basically utilize network topology and node features through aggregating node features along network topology. In view of this, one question naturally arises: *Albeit with different propagation strategies, is there a unified mathematical guideline that essentially governs the propagation mechanisms of different GNNs? If so, what is it?* A well informed answer to this question can provide a macroscopic view on surveying the relationships and differences between different GNNs in a principled way. Such mathematical guideline, once discovered, is able to help us identify the weakness of current GNNs, and further motivates more novel GNNs to be proposed.

As the first contribution of our work, we analyze the propagation process of several representative GNNs (e.g., GCN [13] and PPNP

[14]), and abstract their commonalities. Surprisingly, we discover that they can be fundamentally summarized to a unified optimization framework with flexible graph convolutional kernels. The learned representation after propagation can be viewed as the optimal solution of the corresponding optimization objective implicitly. This unified framework consists of two terms: feature fitting term and graph Laplacian regularization term. The feature fitting term, building the relationship between node representation and original node features, is usually designed to meet different needs of specific GNNs. Graph Laplacian regularization term, playing the role of feature smoothing with topology, is shared by all these GNNs. For example, the propagation of GCN can be interpreted only by the graph Laplacian regularization term while PPNP needs another fitting term to constrain the similarity of the node representation and the original features.

Thanks to the macroscopic view on different GNNs provided by the proposed unified framework, the weakness of current GNNs is easy to be identified. As a consequence, the unified framework opens up new opportunities for designing novel GNNs. Traditionally, when we propose a new GNN model, we usually focus on designing specific spectral graph filter or aggregation strategy. Now, the unified framework provides another new path to achieve this, i.e., the new GNN can be derived by optimizing an objective function. In this way, we clearly know the optimization objective behind the propagation process, making the new GNN more interpretable and more reliable. Here, with the proposed framework, we discover that existing works usually utilize naïve graph convolutional kernels for feature fitting function, and then develop two novel flexible objective functions with adjustable kernels showing low-pass and high-pass filtering capabilities. We show that two corresponding graph neural networks with flexible graph convolutional kernels can be easily derived. Moreover, we also give the convergence ability analysis and expressive power comparisons for these two GNNs. The main contributions are summarized as follows:

- We propose a unified objective optimization framework with a feature fitting function and a graph regularization term, and theoretically prove that this framework is able to unify a series of GNNs propagation mechanisms, providing a macroscopic perspective on understanding GNNs and bringing new insight for designing novel GNNs.
- Within the proposed optimization framework, we design two novel deep GNNs with flexible low-frequency and high-frequency filters which can well alleviate over-smoothing. The theoretical analysis on both of their convergence and excellent expressive power is provided.
- Our extensive experiments on series of benchmark datasets clearly show that the proposed two GNNs outperform the state-of-the-art methods. This further verifies the feasibility for designing GNNs under the unified framework.

2 RELATED WORK

Graph Neural Networks. The current graph neural networks can be broadly divided into two categories: spectral-based GNNs and spatial-based GNNs. Spectral-based GNNs define graph convolutional operations in Fourier domain by designing spectral graph filters. [3] generalizes CNNs to graph signal based on the spectrum of graph Laplacian. ChebNet [5] uses Chebyshev polynomials to approximate the K-order localized graph filters. GCN [13] employs the 1-order simplification of the Chebyshev filter. GWNN [36] leverages sparse and localized graph wavelet transform to design spectral GNNs. Spatial-based GNNs directly design aggregation strategies along network topology, i.e., feature propagation mechanisms. GCN [13] directly aggregates one-hop neighbors along topology. GAT [30] utilizes attention mechanisms to adaptively learn aggregation weights. GraphSAGE [9] uses mean/max/LSTM pooling for aggregation. MixHop [1] aggregates neighbors at various distances to capture mixing relationships. GIN [37] uses a simple but expressive injective multiset function for neighbor aggregation. Policy-GNN [15] uses a meta-policy framework to adaptively learn the aggregation policy. Furthermore, there are some advanced topics have been studied in GNNs. For example, non-Euclidean space graph neural networks [2, 41]; heterogeneous graph neural networks [33, 43]; explanations for graph neural networks [40, 42]; pre-training graph neural networks [11, 25]; robust graph neural networks [12, 46]. For more details, please find in [35, 44, 45] survey papers.

Analysis and understanding on GNNs. Many works on understanding GNNs have been provided recently, which point out ways for designing and improving graph neural networks. Existing theoretical analysis works on GNNs are three-fold: 1) *The spectral filtering characteristic analysis*: *Li et al.* [16] show that the graph convolutional operation is a special form of Laplacian smoothing, and also point out the over-smoothing problem under many layers of graph convolutions; *Wu et al.* [34] make a simplification on GCN and theoretically analyze the resulting linear model acts as a fixed low-pass filter from spectral domain; *NT et al.* [21] also show that the graph convolutional operation is a simplified low-pass filter on original feature vectors and do not have the non-linear manifold learning property from the view of graph signal processing. 2) *The over-smoothing problem analysis*: *Xu et al.* [38] analyze the same over-smoothing problem by establishing the relationship between graph neural networks and random walk. And they show that GCN converges to the limit distribution of random walk as the number of layers increases; *Chen et al.* [4] provide theoretical analysis and imply that nodes with high degrees are more likely to suffer from over-smoothing in a multi-layer graph convolutional model. 3) *The capability of GNNs analysis*: *Hou et al.* [10] work on understanding how much performance GNNs actually gain from graph data and design two smoothness metrics to measure the quantity and quality of obtained information; *Loukas et al.* [18] show that GNNs are Turing universal under sufficient conditions on their depth, width and restricted depth and width may lose a significant portion of their power; *Oono et al.* [22] investigate the expressive power of GNNs as the layer size tends to infinity and show that deep GNNs can only preserve information of node degrees and connected components. However, these works do not theoretically analyze the intrinsic connections about the propagation mechanisms for GNNs. [19], a contemporary work to ours, studies a unified GNN framework from the graph denoising view.

3 A UNIFIED OPTIMIZATION FRAMEWORK

Notations. We consider graph convolutional operations on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$, $n = |\mathcal{V}|$ is the number of nodes. The nodes are described by the feature

matrix $\mathbf{X} \in \mathbb{R}^{n \times f}$, where f is the dimension of node feature. Graph structure of $\mathcal{G}$ can be described by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ where $\mathbf{A}_{i,j} = 1$ if there is an edge between nodes i and j, otherwise 0. The diagonal degree matrix is denoted as $\mathbf{D} = diag(d_1, \cdots, d_n)$, where $d_i = \sum_j \mathbf{A}_{i,j}$. We use $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ to represent the adjacency matrix with added self-loop and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$. Then the normalized adjacency matrix is $\hat{\tilde{\mathbf{A}}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. Correspondingly, $\tilde{\mathbf{L}} = \mathbf{I} - \hat{\tilde{\mathbf{A}}}$ is the normalized symmetric positive semi-definite graph Laplacian matrix.

3.1 The Unified Framework

The well-designed propagation mechanisms of different GNNs basically follow similar propagating steps, i.e, node features aggregate and transform along network topology for a certain depth. Here, we summarize the K-layer propagation mechanisms mainly as the following two forms.

For GNNs with layer-wise feature transformation (e.g. GCN [13]), the K-layer propagation process can be represented as:

$$\mathbf{Z} = \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K) = \left\langle \boldsymbol{Trans}\Big(\boldsymbol{Agg}\big\{\mathcal{G}; \mathbf{Z}^{(k-1)}\big\}\Big)\right\rangle_K, \quad (1)$$

with $\mathbf{Z}^{(0)} = \mathbf{X}$ and $\mathbf{Z}$ is the output representation after the K-layer propagation. And $\langle\rangle_K$, usually depending on specific GNN models, represents the generalized combination operation after K convolutions. $\boldsymbol{Agg}\big\{\mathcal{G}; \mathbf{Z}^{(k-1)}\big\}$ means aggregating the $(k-1)$-layer output $\mathbf{Z}^{(k-1)}$ along graph $\mathcal{G}$ for the k-th convolutional operation, and $\boldsymbol{Trans}(\cdot)$ is the corresponding layer-wise feature transformation operation including non-linear activation function $ReLU()$ and layer-specific learnable weight matrix $\mathbf{W}$.

Some deep graph neural networks (e.g. APPNP [14], DAGNN [17]) decouple the layer-wise $\boldsymbol{Trans}(\cdot)$ and $\boldsymbol{Agg}\big\{\mathcal{G}; \mathbf{Z}^{(k-1)}\big\}$, and use a separated feature transformation before the consecutive aggregation steps:

$$\mathbf{Z} = \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K) = \left\langle \boldsymbol{Agg}\big\{\mathcal{G}; \mathbf{Z}^{(k-1)}\big\}\right\rangle_K, \quad (2)$$

with $\mathbf{Z}^{(0)} = \boldsymbol{Trans}(\mathbf{X})$ and $\boldsymbol{Trans}(\cdot)$ can be any linear or non-linear transformation operation on original feature matrix $\mathbf{X}$.

In addition, the combination operation $\langle\rangle_K$ is generally two-fold: for GNNs like GCN, SGC, and APPNP, $\langle\rangle_K$ directly utilizes the K-th layer output. And for GNNs using outputs from other layers, like JKNet and DAGNN, $\langle\rangle_K$ may represent pooling, concatenation or attention operations on the some (or all) outputs from K layers.

Actually, the propagation process including aggregation and transformation is the key core of GNNs. Network topology and node features are the two most essential sources of information improving the learned representation during propagation: network topology usually plays the role of low-pass filter on the input node signals, which smooths the features of two connected nodes [16]. In this way, the learned node representation is able to capture the homophily of graph structure. As for the node feature, itself contains complex information, e.g., low-frequency and high-frequency information. Node feature can be flexibly used to further restrain the learned node representation. For example, APPNP adds the original node feature to the representation learned by each layer, which well preserves the personalized information so as to alleviate over-smoothing.

The above analysis implies that despite various GNNs are proposed with different propagation mechanisms, in fact, they usually potentially aim at achieving two goals: encoding useful information from feature and utilizing the smoothing ability of topology, which can be formally formulated as the following optimization objective:

$$\mathcal{O} = \min_{\mathbf{Z}} \big\{ \underbrace{\zeta \big\|\mathbf{F}_1\mathbf{Z} - \mathbf{F}_2\mathbf{H}\big\|_F^2}_{O_{fit}} + \underbrace{\xi tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})}_{O_{reg}} \big\}. \quad (3)$$

Here, ξ is a non-negative coefficient, ζ is usually chosen from [0, 1], and $\mathbf{H}$ is the transformation on original input feature matrix $\mathbf{X}$. $\mathbf{F}_1$ and $\mathbf{F}_2$ are defined as arbitrary graph convolutional kernels. $\mathbf{Z}$ is the propagated representation and corresponds to the final propagation result when minimizing the objective $\mathcal{O}$.

In this unified framework, the first part O_{fit} is a fitting term which flexibly encodes the information in $\mathbf{H}$ to the learned representation $\mathbf{Z}$ through designing different graph convolutional kernels $\mathbf{F}_1$ and $\mathbf{F}_2$. Graph convolutional kernels $\mathbf{F}_1$ and $\mathbf{F}_2$ can be chosen from the $\mathbf{I}$, $\hat{\tilde{\mathbf{A}}}$, $\tilde{\mathbf{L}}$, showing the all-pass, low-pass, high-pass filtering capabilities respectively. The second term O_{reg} is a graph Laplacian regularization term constraining the learned representations of two connected nodes become similar, so that the homophily property can be captured, and O_{reg} comes from the following graph Laplacian regularization:

$$O_{reg} = \frac{\xi}{2}\sum_{i,j}^{n} \hat{\tilde{\mathbf{A}}}_{i,j}\big\|\mathbf{Z}_i - \mathbf{Z}_j\big\|^2 = \xi tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z}). \quad (4)$$

In the following, we theoretically prove that the propagation mechanisms of some typical GNNs are actually the special cases of our proposed unified framework as shown in Table 1. This unified framework builds the connection among some typical GNNs, enabling us to interpret the current GNNs in a global perspective.

3.2 Interpreting GCN and SGC

GCN [13]/**SGC** [34]. Graph Convolutional Network (GCN) has the following propagation mechanism which conducts linear transformation and nonlinearity activation repeatedly throughout K layers:

$$\begin{aligned}\mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K)_{gcn} \\ &= \hat{\tilde{\mathbf{A}}}\sigma(\cdots\sigma(\hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}^{(0)})\cdots)\mathbf{W}^{(K-1)}.\end{aligned} \quad (5)$$

Simplifying Graph Convolutional Network (SGC) reduces this excess complexity through removing nonlinearities and collapsing weight matrices between consecutive layers. The linear model exhibits comparable performance since SGC has the similar propagation mechanism with GCN as:

$$\begin{aligned}\mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K)_{sgc} \\ &= \hat{\tilde{\mathbf{A}}}\cdots\hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}^{(0)}\cdots\mathbf{W}^{(K-1)} = \hat{\tilde{\mathbf{A}}}^K\mathbf{X}\mathbf{W}^*,\end{aligned} \quad (6)$$

where $\mathbf{W}^* = \mathbf{W}^{(0)}\mathbf{W}^{(1)}\cdots\mathbf{W}^{(K-1)}$. We have the following interpretations on the propagation mode of SGC (GCN) under the proposed unified framework.

Table 1: The overall correspondences between propagation mechanisms and optimization objectives for GNNs.

Model	Characteristic	Propagation Mechanism	Corresponding Objective
GCN/SGC [13]	K-layer graph convolutions	$\mathbf{Z} = \hat{\tilde{\mathbf{A}}}^K\mathbf{X}\mathbf{W}^*$	$O = \min_{\mathbf{Z}} \{tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}, \mathbf{Z}^{(0)} = \mathbf{X}\mathbf{W}^*$
GC Operation [13]	1-layer graph convolution	$\mathbf{Z} = \hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}$	$O = \min_{\mathbf{Z}} \{\|\mathbf{Z}-\mathbf{H}\|_F^2 + tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}, \mathbf{H} = \mathbf{X}\mathbf{W}$, (*first-order*)
PPNP/APPNP [14]	Personalized pagerank	$\mathbf{H} = f_\theta(\mathbf{X}),$ $\begin{cases}\textbf{PPNP:} & \mathbf{Z} = \alpha(\mathbf{I}-(1-\alpha)\hat{\tilde{\mathbf{A}}})^{-1}\mathbf{H} \\ \textbf{APPNP:} & \mathbf{Z} = \left\langle (1-\alpha)\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \alpha\mathbf{H}\right\rangle_K, \mathbf{Z}^{(0)} = \mathbf{H}\end{cases}$	$O = \min_{\mathbf{Z}} \{\|\mathbf{Z}-\mathbf{H}\|_F^2 + (1/\alpha-1)tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}$
JKNet [38]	Jumping to the last layer	$\mathbf{Z} = \sum_{k=1}^{K} \alpha_k \hat{\tilde{\mathbf{A}}}^k\mathbf{X}\mathbf{W}^*$	$O = \min_{\mathbf{Z}} \{\|\mathbf{Z}-\hat{\tilde{\mathbf{A}}}\mathbf{H}\|_F^2 + \xi tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}, \mathbf{H} = \mathbf{X}\mathbf{W}^*$
DAGNN [17]	Adaptively incorporating different layers	$\mathbf{H} = f_\theta(\mathbf{X}), \quad \mathbf{Z} = \sum_{k=0}^{K} s_k \hat{\tilde{\mathbf{A}}}^k\mathbf{H}$	$O = \min_{\mathbf{Z}} \{\|\mathbf{Z}-\mathbf{H}\|_F^2 + \xi tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}$
GNN-LF (ours)	Flexible low-pass filtering kernel	$\mathbf{H} = f_\theta(\mathbf{X}),$ $\begin{cases}\textbf{closed:} & \mathbf{Z} = \{(\mu+1/\alpha-1)\mathbf{I} + \{2-\mu-1/\alpha\}\hat{\tilde{\mathbf{A}}}\}^{-1}\{\mu\mathbf{I}+(1-\mu)\hat{\tilde{\mathbf{A}}}\}\mathbf{H} \\ \textbf{iter:} & \mathbf{Z} = \left\langle \frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha\mu}{1+\alpha\mu-\alpha}\mathbf{H} + \frac{\alpha-\alpha\mu}{1+\alpha\mu-\alpha}\hat{\tilde{\mathbf{A}}}\mathbf{H}\right\rangle_K, \\ & \mathbf{Z}^{(0)} = \frac{\mu}{1+\alpha\mu-\alpha}\mathbf{H} + \frac{1-\mu}{1+\alpha\mu-\alpha}\hat{\tilde{\mathbf{A}}}\mathbf{H}\end{cases}$	$O = \min_{\mathbf{Z}} \{\|\{\mu\mathbf{I}+(1-\mu)\hat{\tilde{\mathbf{A}}}\}^{1/2}(\mathbf{Z}-\mathbf{H})\|_F^2 + (1/\alpha-1)tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}$
GNN-HF (ours)	Flexible high-pass filtering kernel	$\mathbf{H} = f_\theta(\mathbf{X}),$ $\begin{cases}\textbf{closed:} & \mathbf{Z} = \{(\beta+1/\alpha)\mathbf{I} + (1-\beta-1/\alpha)\hat{\tilde{\mathbf{A}}}\}^{-1}\{\mathbf{I}+\beta\tilde{\mathbf{L}}\}\mathbf{H} \\ \textbf{iter:} & \mathbf{Z} = \left\langle \frac{\alpha\beta-\alpha+1}{\alpha\beta+1}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha}{\alpha\beta+1}\mathbf{H} + \frac{\alpha\beta}{\alpha\beta+1}\tilde{\mathbf{L}}\mathbf{H}\right\rangle_K, \\ & \mathbf{Z}^{(0)} = \frac{1}{\alpha\beta+1}\mathbf{H} + \frac{\beta}{\alpha\beta+1}\tilde{\mathbf{L}}\mathbf{H}\end{cases}$	$O = \min_{\mathbf{Z}} \{\|\{\mathbf{I}+\beta\tilde{\mathbf{L}}\}^{1/2}(\mathbf{Z}-\mathbf{H})\|_F^2 + (1/\alpha-1)tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\}$

Theorem 3.1. *With $\zeta = 0$ and $\xi = 1$ in Eq. (3), the propagation process of SGC/GCN optimizes the following graph regularization term:*

$$O = \min_{\boldsymbol{Z}} \left\{tr(\boldsymbol{Z}^T\tilde{\boldsymbol{L}}\boldsymbol{Z})\right\}, \tag{7}$$

where $\boldsymbol{Z}$ is initialized as $\boldsymbol{X}\boldsymbol{W}^*$.

Proof. Set derivative of Eq. (7) with respect to $\mathbf{Z}$ to zero:

$$\frac{\partial tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})}{\partial \mathbf{Z}} = 0 \quad \Rightarrow \quad \tilde{\mathbf{L}}\mathbf{Z} = 0 \quad \Rightarrow \quad \mathbf{Z} = \hat{\tilde{\mathbf{A}}}\mathbf{Z}. \tag{8}$$

Eq. (8) can be explained as an limit distribution where $\mathbf{Z}_{lim} = \hat{\tilde{\mathbf{A}}}\mathbf{Z}_{lim}$. Then we use the following iterative form to approximate the limit $\mathbf{Z}_{lim}$ with $K \to \infty$:

$$\mathbf{Z}^{(K)} = \hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(K-1)}. \tag{9}$$

When SGC initializes input representation as $\mathbf{Z}^{(0)} = \mathbf{X}\mathbf{W}^*$, Eq. (9) becomes:

$$\mathbf{Z}^{(K)} = \hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(K-1)} = \hat{\tilde{\mathbf{A}}}^2\mathbf{Z}^{(K-2)} = \cdots = \hat{\tilde{\mathbf{A}}}^K\mathbf{Z}^{(0)} = \hat{\tilde{\mathbf{A}}}^K\mathbf{X}\mathbf{W}^*, \tag{10}$$

which matches the propagation mechanism of SGC. Since GCN can be simplified as SGC by ignoring the non-linear transformation, this conclusion also holds for GCN. □

Graph Convolutional Operation. The above analysis is for the consecutive K layers graph convolutional operations, here, we also pay attention to the one layer graph convolutional operation (**GC operation**) with the following propagation mechanism:

$$\mathbf{Z} = \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; 1)_{gc} = \hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}. \tag{11}$$

Theorem 3.2. *With $\boldsymbol{F}_1 = \boldsymbol{F}_2 = \boldsymbol{I}$, $\zeta = 1, \xi = 1$ in Eq. (3), the 1-layer GC operation optimizes the following objective under first-order approximation:*

$$O = \min_{\boldsymbol{Z}} \left\{\left\|\boldsymbol{Z}-\boldsymbol{H}\right\|_F^2 + tr(\boldsymbol{Z}^T\tilde{\boldsymbol{L}}\boldsymbol{Z})\right\}, \tag{12}$$

where $\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}$ is the linear transformation on feature, $\boldsymbol{W}$ is a trainable weight matrix.

Proof. Please refer to Appendix A.1 □

3.3 Interpreting PPNP and APPNP

PPNP [14] is a graph neural network which utilizes a propagation mechanism derived from personalized PageRank and separates the feature transformation from the aggregation process:

$$\begin{aligned}\mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; \infty)_{ppnp} \\ &= \alpha\big(\mathbf{I}-(1-\alpha)\hat{\tilde{\mathbf{A}}}\big)^{-1}\mathbf{H}, \quad \textit{and} \quad \mathbf{H} = f_\theta(\mathbf{X}),\end{aligned} \tag{13}$$

where $\alpha \in (0, 1]$ is the teleport probability, and $\mathbf{H}$ is the non-linear transformation result of the original feature $\mathbf{X}$ using an MLP network $f_\theta(\cdot)$.

Furthermore, due to the high complexity of calculating the inverse matrix, a power iterative version with linear computational complexity named APPNP is used for approximation. The propagation process of APPNP can be viewed as a layer-wise graph convolution with a residual connection to the initial transformed feature matrix $\mathbf{H}$:

$$\begin{aligned}\mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K)_{appnp} \\ &= \left\langle (1-\alpha)\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \alpha\mathbf{H}\right\rangle_K, \quad \textit{and} \quad \mathbf{Z}^{(0)} = \mathbf{H} = f_\theta(\mathbf{X}).\end{aligned} \tag{14}$$

Actually, it has been proved in [14] that APPNP converges to PPNP when $K \to \infty$, so we use one objective under the framework to explain both of them.

Theorem 3.3. *With $\boldsymbol{F}_1 = \boldsymbol{F}_2 = \boldsymbol{I}$, $\zeta = 1, \xi = 1/\alpha - 1, \alpha \in (0, 1]$ in Eq. (3), the propagation process of PPNP/APPNP optimizes the following objective:*

$$O = \min_{\boldsymbol{Z}} \left\{\left\|\boldsymbol{Z}-\boldsymbol{H}\right\|_F^2 + \xi tr(\boldsymbol{Z}^T\tilde{\boldsymbol{L}}\boldsymbol{Z})\right\}, \tag{15}$$

where $\boldsymbol{H} = f_\theta(\boldsymbol{X})$.

Proof. We can set the derivative of Eq. (15) with respect to $\mathbf{Z}$ to zero and get the optimal $\mathbf{Z}$ as:

$$\frac{\partial\left\{\|\mathbf{Z}-\mathbf{H}\|_F^2+\xi tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\right\}}{\partial\mathbf{Z}}=0 \quad \Rightarrow \quad \mathbf{Z}-\mathbf{H}+\xi\tilde{\mathbf{L}}\mathbf{Z}=0. \tag{16}$$

Note that a matrix $\mathbf{M}$ has an inverse matrix if and only if the determinant of matrix $\mathbf{det}(\mathbf{M})$ is not zero. Since the eigenvalues of the normalized Laplacian matrix $\lambda_i \in [0,2)$, and the eigenvalues of the matrix $\mathbf{I}+\xi\tilde{\mathbf{L}}$ are $(1+\xi\lambda_i)>0$. Then $\mathbf{det}(\mathbf{I}+\xi\tilde{\mathbf{L}})>0$ and $\mathbf{Z}$ in Eq. (16) can be rewritten as:

$$\mathbf{Z}=(\mathbf{I}+\xi\tilde{\mathbf{L}})^{-1}\mathbf{H}. \tag{17}$$

We use $\hat{\tilde{\mathbf{A}}}$ and α to rewrite Eq. (17):

$$\mathbf{Z}=\left\{\mathbf{I}+(1/\alpha-1)\big(\mathbf{I}-\hat{\tilde{\mathbf{A}}}\big)\right\}^{-1}\mathbf{H}=\alpha\big(\mathbf{I}-(1-\alpha)\hat{\tilde{\mathbf{A}}}\big)^{-1}\mathbf{H}, \tag{18}$$

which exactly corresponds to the propagation mechanism of PPNP or the convergence propagation result of APPNP. □

3.4 Interpreting JKNet and DAGNN

JKNet [38] is a deep graph neural network which exploits information from neighborhoods of differing locality. This architecture selectively combines aggregations from different layers with Concatenation/Max-pooling/Attention at the output, i.e., the representations "jump" to the last layer.

For convenience, following [4, 34], we simplify the k-th ($k \in [1,K]$) layer graph convolutional operation in the similar way by ignoring the non-linear activation with $\sigma(x)=x$ and sharing $\mathbf{W}^*=\mathbf{W}^{(0)}\mathbf{W}^{(1)}\cdots\mathbf{W}^{(k-1)}$ for each layer. Then k-th layer temporary output is $\hat{\tilde{\mathbf{A}}}^k\mathbf{X}\mathbf{W}^*$. Using attention mechanism for combination at the last layer, the K-layer propagation result of JKNet can be written as:

$$\begin{aligned}&\mathbf{Z}=\mathbf{PROPAGATE}(\mathbf{X};\mathcal{G};K)_{JKNet}\\&=\alpha_1\hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}^*+\alpha_2\hat{\tilde{\mathbf{A}}}^2\mathbf{X}\mathbf{W}^*+\cdots+\alpha_K\hat{\tilde{\mathbf{A}}}^K\mathbf{X}\mathbf{W}^*=\sum_{k=1}^{K}\alpha_k\hat{\tilde{\mathbf{A}}}^k\mathbf{X}\mathbf{W}^*,\end{aligned} \tag{19}$$

where $\alpha_1,\alpha_2,...,\alpha_K$ are the learnable fusion weights with $\sum_{k=1}^{K}\alpha_k=1$, and for convenient analysis, we assume that all nodes of the k-th layer share one common weight α_k.

Theorem 3.4. *With $\boldsymbol{F}_1=\boldsymbol{I}$, $\boldsymbol{F}_2=\hat{\tilde{\boldsymbol{A}}}$, $\zeta=1$, and $\xi\in(0,\infty)$ in Eq. (3), the propagation process of JKNet optimizes the following objective:*

$$O=\min_{\boldsymbol{Z}}\left\{\left\|\boldsymbol{Z}-\hat{\tilde{\boldsymbol{A}}}\boldsymbol{H}\right\|_F^2+\xi tr(\boldsymbol{Z}^T\tilde{\boldsymbol{L}}\boldsymbol{Z})\right\}, \tag{20}$$

here $\boldsymbol{H}=\boldsymbol{X}\boldsymbol{W}^*$ is the linear feature transformation after simplifications.

Proof. Similarly, we can set derivative of Eq. (20) with respect to $\mathbf{Z}$ to zero and get the optimal $\mathbf{Z}$ as:

$$\frac{\partial\left\{\|\mathbf{Z}-\hat{\tilde{\mathbf{A}}}\mathbf{H}\|_F^2+\xi tr(\mathbf{Z}^T\tilde{\mathbf{L}}\mathbf{Z})\right\}}{\partial\mathbf{Z}}=0 \quad \Rightarrow \quad \mathbf{Z}-\hat{\tilde{\mathbf{A}}}\mathbf{H}+\xi\tilde{\mathbf{L}}\mathbf{Z}=0. \tag{21}$$

Note that $\mathbf{det}(\mathbf{I}+\xi\tilde{\mathbf{L}})>0$, thus matrix $\{\mathbf{I}+\xi\tilde{\mathbf{L}}\}^{-1}$ exists. Then the corresponding closed-form solution can be written as:

$$\mathbf{Z}=\left\{(1+\xi)\mathbf{I}-\xi\hat{\tilde{\mathbf{A}}}\right\}^{-1}\hat{\tilde{\mathbf{A}}}\mathbf{H}. \tag{22}$$

Since $\frac{\xi}{1+\xi}<1$ for $\forall\xi>0$, and matrix $\hat{\tilde{\mathbf{A}}}$ has absolute eigenvalues bounded by 1, thus, all its positive powers have bounded operator norm, then the inverse matrix can be decomposed as follows with $K\to\infty$:

$$\begin{aligned}\mathbf{Z}&=\frac{1}{1+\xi}\left\{\mathbf{I}-\frac{\xi}{1+\xi}\hat{\tilde{\mathbf{A}}}\right\}^{-1}\hat{\tilde{\mathbf{A}}}\mathbf{H}\\&=\frac{1}{1+\xi}\left\{\mathbf{I}+\frac{\xi}{1+\xi}\hat{\tilde{\mathbf{A}}}+\cdots+\frac{\xi^{K-1}}{(1+\xi)^{K-1}}\hat{\tilde{\mathbf{A}}}^{K-1}+\cdots\right\}\hat{\tilde{\mathbf{A}}}\mathbf{H}.\end{aligned} \tag{23}$$

With $\mathbf{H}=\mathbf{X}\mathbf{W}^*$, we have the following expansion:

$$\mathbf{Z}=\frac{1}{1+\xi}\hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}^*+\frac{\xi}{(1+\xi)^2}\hat{\tilde{\mathbf{A}}}^2\mathbf{X}\mathbf{W}^*+\cdots+\frac{\xi^{K-1}}{(1+\xi)^K}\hat{\tilde{\mathbf{A}}}^K\mathbf{X}\mathbf{W}^*+\cdots. \tag{24}$$

Note that $\frac{1}{1+\xi}+\frac{\xi}{(1+\xi)^2}+\cdots+\frac{\xi^{K-1}}{(1+\xi)^K}+\cdots=1$ and we can change the coefficient $\xi\in(0,\infty)$ to fit fusion weights $\alpha_1,\alpha_2,\cdots,\alpha_K$. When the layer K is large enough, the propagation mechanism of JKNet in Eq. (19) approximately corresponds to the objective Eq. (20). □

DAGNN [17]. Deep Adaptive Graph Neural Networks (DAGNN) tries to adaptively incorporate information from large receptive fields. After decoupling representation transformation and propagation, the propagation mechanism of DAGNN is similar to that of JKNet:

$$\begin{aligned}\mathbf{Z}&=\mathbf{PROPAGATE}(\mathbf{X};\mathcal{G};K)_{DAGNN}\\&=s_0\mathbf{H}+s_1\hat{\tilde{\mathbf{A}}}\mathbf{H}+s_2\hat{\tilde{\mathbf{A}}}^2\mathbf{H}+\cdots+s_K\hat{\tilde{\mathbf{A}}}^K\mathbf{H}\\&=\sum_{k=0}^{K}s_k\hat{\tilde{\mathbf{A}}}^k\mathbf{H}, \quad \textit{and} \quad \mathbf{H}=f_\theta(\mathbf{X}).\end{aligned} \tag{25}$$

$\mathbf{H}=f_\theta(\mathbf{X})$ is the non-linear feature transformation using an MLP network, which is conducted before the propagation process, and $s_0,s_1,\cdots,s_K$ are the learnable retainment scores where $\sum_{k=0}^{K}s_k=1$ and we assume that all nodes of the k-th layer share one common weight s_k for convenience.

Theorem 3.5. *With $\boldsymbol{F}_1=\boldsymbol{F}_2=\boldsymbol{I}$, $\zeta=1$ and $\xi\in(0,\infty)$ in Eq. (3), the propagation process of DAGNN optimizes the following objective:*

$$O=\min_{\boldsymbol{Z}}\left\{\left\|\boldsymbol{Z}-\boldsymbol{H}\right\|_F^2+\xi tr(\boldsymbol{Z}^T\tilde{\boldsymbol{L}}\boldsymbol{Z})\right\}, \tag{26}$$

where $\boldsymbol{H}=f_\theta(\boldsymbol{X})$ is the non-linear transformation on feature matrix, the retainment scores $s_0,s_1,\cdots,s_K$ are approximated by $\xi\in(0,\infty)$.

Proof. We also set derivative of Eq. (26) with respect to $\mathbf{Z}$ to zero and get the closed-form solution as:

$$\mathbf{Z}=\left\{(1+\xi)\mathbf{I}-\xi\hat{\tilde{\mathbf{A}}}\right\}^{-1}\mathbf{H}. \tag{27}$$

Through the decomposition process similar to JKNet, we have the following expansion:

$$\mathbf{Z}=\frac{1}{1+\xi}\mathbf{H}+\frac{\xi}{(1+\xi)^2}\hat{\tilde{\mathbf{A}}}\mathbf{H}+\cdots+\frac{\xi^K}{(1+\xi)^{K+1}}\hat{\tilde{\mathbf{A}}}^K\mathbf{H}+\cdots. \tag{28}$$

Note that we can change $\xi\in(0,\infty)$ to fit the retainment scores where $\frac{1}{1+\xi}+\frac{\xi}{(1+\xi)^2}+\cdots+\frac{\xi^{K-1}}{(1+\xi)^K}+\cdots=1$. Then the propagation mechanism of DAGNN approximately corresponds to the objective Eq. (26). □

3.5 Discussion

For clarity, we conclude the overall relations between different GNNs and the corresponding objective functions in Table 1. It can be seen that our proposed framework abstracts the commonalities between different promising representative GNNs. Based on the framework, we can understand their relationships much easier. For example, the corresponding optimization objective for SGC in Theorem 3.1 only has a graph regularization term, while the objective for APPNP in Theorem 3.3 has both fitting term and graph regularization term. The explicit difference of objective function well explains deep APPNP (PPNP) outperforms SGC (GCN) on over-smoothing problem by additionally requiring the learned representation to encode the original features.

On the other hand, our proposed framework shows a big picture of GNNs by mathematically modelling the objective optimization function. Considering that different existing GNNs can be fit into this framework, novel variations of GNNs can also be easily come up. All we need is to design the variables within this framework (e.g., different graph convolutional kernels $\mathbf{F}_1$ and $\mathbf{F}_2$) based on the specific scenarios, the corresponding propagation can be easily derived, and new GNNs architecture can be naturally designed. With one targeted objective function, the newly designed model is more interpretable and more reliable.

4 GNN-LF/HF: OUR PROPOSED MODELS

Based on the unified framework, we find that most of the current GNNs simply set $\mathbf{F}_1$ and $\mathbf{F}_2$ as $\mathbf{I}$ in feature fitting term, implying that they require all original information in $\mathbf{H}$ to be encoded into $\mathbf{Z}$. However, in fact, the $\mathbf{H}$ may inevitably contain noise or uncertain information. We notice that JKNet has the propagation objective with $\mathbf{F}_2$ as $\hat{\tilde{\mathbf{A}}}$, which can encode the low-frequency information in $\mathbf{H}$ to $\mathbf{Z}$. While, in reality, the situation is more complex because it is hard to determine what information should be encoded, only considering one type of information cannot satisfy the needs of different downstream tasks, and sometimes high-frequency or all information is even also helpful. In this section, we focus on designing novel $\mathbf{F}_1$ and $\mathbf{F}_2$ to flexibly encode more comprehensive information under the framework.

4.1 GNN with Low-pass Filtering Kernel

4.1.1 Objective Function. We first consider building the relationship of $\mathbf{H}$ and $\mathbf{Z}$ in both original and low-pass filtering spaces.

Theorem 4.1. *With $\boldsymbol{F}_1 = \boldsymbol{F}_2 = \{\mu\boldsymbol{I} + (1-\mu)\hat{\tilde{\boldsymbol{A}}}\}^{1/2}$, $\mu \in [1/2, 1)$, $\zeta = 1$ and $\xi = 1/\alpha - 1$, $\alpha \in (0, 2/3)$ in Eq. (3), the propagation process considering flexible low-pass filtering kernel on feature is:*

$$O = min\Big\{\Big\|\{\mu\boldsymbol{I} + (1-\mu)\hat{\tilde{\boldsymbol{A}}}\}^{1/2}(\boldsymbol{Z} - \boldsymbol{H})\Big\|_F^2 + \xi tr(\boldsymbol{Z}^T\tilde{\boldsymbol{L}}\boldsymbol{Z})\Big\}, \quad (29)$$

where $\boldsymbol{H} = f_\theta(\boldsymbol{X})$.

Note that μ is a balance coefficient, and we set $\mu \in [1/2, 1)$ so that $\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$ is a symmetric and positive semi-definite matrix. Therefore, the matrix $\{\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}}\}^{1/2} = \boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^T$ has a filtering behavior similar to that of $\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}}$ in spectral domain. And we set $\alpha \in (0, 2/3)$ to ensure the iterative approximation solution in subsection 4.1.3 has positive coefficients. By adjusting the balance coefficient μ, the designed objective can flexibly constrain the similarity of $\mathbf{Z}$ and $\mathbf{H}$ in both original and low-pass filtering spaces, which is beneficial to meet the needs of different tasks.

4.1.2 Closed Solution. To minimize the objective function in Eq. (29), we set derivative of Eq. (29) with respect to $\mathbf{Z}$ to zero and derive the corresponding closed-form solution as follows:

$$\mathbf{Z} = \{\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}} + (1/\alpha - 1)\tilde{\mathbf{L}}\}^{-1}\{\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}}\}\mathbf{H}. \quad (30)$$

We can rewrite the Eq. (30) using $\hat{\tilde{\mathbf{A}}}$ as:

$$\mathbf{Z} = \Big\{\{\mu + 1/\alpha - 1\}\mathbf{I} + \{2 - \mu - 1/\alpha\}\hat{\tilde{\mathbf{A}}}\Big\}^{-1}\{\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}}\}\mathbf{H}. \quad (31)$$

4.1.3 Iterative Approximation. Considering that the closed-form solution is computationally inefficient because of the matrix inversion, we can use the following iterative approximation solution instead without constructing the dense inverse matrix:

$$\mathbf{Z}^{(k)} = \frac{1 + \alpha\mu - 2\alpha}{1 + \alpha\mu - \alpha}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha\mu}{1 + \alpha\mu - \alpha}\mathbf{H} + \frac{\alpha - \alpha\mu}{1 + \alpha\mu - \alpha}\hat{\tilde{\mathbf{A}}}\mathbf{H}, \quad (32)$$

which converge to the closed-form solution in Eq. (31) when $k \to \infty$, and with $\alpha \in (0, 2/3)$, all the coefficients are always positive.

4.1.4 Model Design. With the derived two propagation strategies in Eq. (31) and Eq. (32), we propose two new GNNs in both **closed** and **iter**ative forms. Note that we represent the proposed models as **GNN** with **L**ow-pass **F**iltering graph convolutional kernel (**GNN-LF**).

GNN-LF-closed Using the closed-form propagation matrix in Eq. (31), we define the following propagation mechanism with $\mu \in [1/2, 1)$, $\alpha \in (0, 2/3)$ and $\mathbf{H} = f_\theta(\mathbf{X})$:

$$\begin{aligned}\mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; \infty)_{LF-closed} \\ &= \Big\{\{\mu + 1/\alpha - 1\}\mathbf{I} + \{2 - \mu - 1/\alpha\}\hat{\tilde{\mathbf{A}}}\Big\}^{-1}\{\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}}\}\mathbf{H}, \\ &\textit{and} \quad \mathbf{H} = f_\theta(\mathbf{X}).\end{aligned} \quad (33)$$

Here we first get a non-linear transformation result $\mathbf{H}$ on feature $\mathbf{X}$ with an MLP network $f_\theta(\cdot)$, and use the designed propagation matrix $\{\{\mu + 1/\alpha - 1\}\mathbf{I} + \{2 - \mu - 1/\alpha\}\hat{\tilde{\mathbf{A}}}\}^{-1}$ to propagate both $\mathbf{H}$ and $\mathbf{AH}$, then we can get the representation encoding feature information from both original and low-frequency spaces.

GNN-LF-iter Using the iter-form propagation mechanism in Eq. (32), we can design a deep and computationally efficient graph neural network with $\mu \in [1/2, 1)$, $\alpha \in (0, 2/3)$:

$$\begin{aligned}\mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K)_{LF-iter} \\ &= \left\langle \frac{1 + \alpha\mu - 2\alpha}{1 + \alpha\mu - \alpha}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha\mu}{1 + \alpha\mu - \alpha}\mathbf{H} + \frac{\alpha - \alpha\mu}{1 + \alpha\mu - \alpha}\hat{\tilde{\mathbf{A}}}\mathbf{H} \right\rangle_K, \\ \mathbf{Z}^{(0)} &= \frac{\mu}{1 + \alpha\mu - \alpha}\mathbf{H} + \frac{1-\mu}{1 + \alpha\mu - \alpha}\hat{\tilde{\mathbf{A}}}\mathbf{H}, \quad \textit{and} \quad \mathbf{H} = f_\theta(\mathbf{X}).\end{aligned} \quad (34)$$

We directly use the K-layer output as the propagation results. This iterative propagation mechanism can be viewed as layer-wise $\hat{\tilde{\mathbf{A}}}$ based neighborhood aggregation, with residual connection on feature matrix $\mathbf{H}$ and filtered feature matrix $\hat{\tilde{\mathbf{A}}}\mathbf{H}$. Note that we decouple the layer-wise transformation and aggregation process like [14, 17], which is beneficial to alleviate the over-smoothing problem. GNN-LF-iter and GNN-LF-closed have the following relation:

Theorem 4.2. *With $K \to \infty$, deep GNN-LF-iter converges to GNN-LF-closed with the same propagation result as Eq. (31).*

Proof. After the K-layer propagation using GNN-LF-iter, the corresponding output result $\mathbf{Z}$ can be written as:

$$\mathbf{Z} = \left\{ (\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha})^K \hat{\tilde{\mathbf{A}}}^K + \alpha \sum_{i=0}^{K-1} (\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha})^i \hat{\tilde{\mathbf{A}}}^i \right\} \left\{ \frac{\mu}{1+\alpha\mu-\alpha}\mathbf{H} + \frac{1-\mu}{1+\alpha\mu-\alpha}\hat{\tilde{\mathbf{A}}}\mathbf{H} \right\}, \tag{35}$$

where $\mu \in [1/2, 1)$, $\alpha \in (0, 2/3)$ and $|\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}| < 1$. When $K \to \infty$, the left term tends to 0 and the right term becomes a geometric series. The series converges since $\hat{\tilde{\mathbf{A}}}$ has absolute eigenvalues bounded by 1, then Eq. (35) can be rewritten as:

$$\mathbf{Z} = \left\{ \{\mu + 1/\alpha - 1\}\mathbf{I} + \{2 - \mu - 1/\alpha\}\hat{\tilde{\mathbf{A}}} \right\}^{-1} \{\mu\mathbf{I} + (1-\mu)\hat{\tilde{\mathbf{A}}}\}\mathbf{H}, \tag{36}$$

which exactly is the equation for calculating GNN-LF-closed. □

The training of GNN-LF is also the same with other GNNs. For example, it evaluates the cross-entropy loss over all labeled examples for semi-supervised multi-class node classification task.

4.2 GNN with High-pass Filtering Kernel

4.2.1 Objective Function. Similar with GNN-LF, we now consider preserving the similarity of $\mathbf{H}$ and $\mathbf{Z}$ in both original and high-pass filtering spaces. For neatness of the subsequent analysis, we choose the following objective:

Theorem 4.3. *With $\boldsymbol{F}_1 = \boldsymbol{F}_2 = \{\boldsymbol{I} + \beta\tilde{\boldsymbol{L}}\}^{1/2}$, $\beta \in (0, \infty)$, $\zeta = 1$ and $\xi = 1/\alpha - 1$, $\alpha \in (0, 1]$ in Eq. (3), the propagation process considering flexible high-pass convolutional kernel on feature is:*

$$O = \min_{\boldsymbol{Z}} \left\{ \left\| \{\boldsymbol{I} + \beta\tilde{\boldsymbol{L}}\}^{1/2}(\boldsymbol{Z} - \boldsymbol{H}) \right\|_F^2 + \xi tr(\boldsymbol{Z}^T \tilde{\boldsymbol{L}}\boldsymbol{Z}) \right\}, \tag{37}$$

where $\boldsymbol{H} = f_\theta(\boldsymbol{X})$.

Analogously, β is also a balance coefficient, and we set $\beta \in (0, \infty)$ so that $\mathbf{I} + \beta\tilde{\mathbf{L}} = \boldsymbol{V}^*\boldsymbol{\Lambda}^*\boldsymbol{V}^{*T}$ is also a symmetric and positive semi-definite matrix and the matrix $\{\mathbf{I} + \beta\tilde{\mathbf{L}}\}^{1/2} = \boldsymbol{V}^*\boldsymbol{\Lambda}^{*1/2}\boldsymbol{V}^{*T}$ has a filtering behavior similar to that of $\{\mathbf{I} + \beta\tilde{\mathbf{L}}\}$. As can be seen in Eq. (37), by adjusting the balance coefficient β, the designed objectives can flexibly constrain the similarity of $\mathbf{Z}$ and $\mathbf{H}$ in both original and high-frequency spaces.

4.2.2 Closed Solution. We calculate the closed-form solution as:

$$\mathbf{Z} = \left\{ \mathbf{I} + (\beta + 1/\alpha - 1)\tilde{\mathbf{L}} \right\}^{-1} \{\mathbf{I} + \beta\tilde{\mathbf{L}}\}\mathbf{H}, \tag{38}$$

it also can be rewritten as:

$$\mathbf{Z} = \left\{ (\beta + 1/\alpha)\mathbf{I} + (1 - \beta - 1/\alpha)\hat{\tilde{\mathbf{A}}} \right\}^{-1} \{\mathbf{I} + \beta\tilde{\mathbf{L}}\}\mathbf{H}. \tag{39}$$

4.2.3 Iterative Approximation. Considering it is inefficient to calculate the inverse matrix, we give the following iterative approximation solution without constructing the dense inverse matrix:

$$\mathbf{Z}^{(k)} = \frac{\alpha\beta - \alpha + 1}{\alpha\beta + 1}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha}{\alpha\beta + 1}\mathbf{H} + \frac{\alpha\beta}{\alpha\beta + 1}\tilde{\mathbf{L}}\mathbf{H}. \tag{40}$$

4.2.4 Model Design. With the derived two propagation strategies in Eq. (39) and in Eq. (40), we propose two new GNNs in both **closed** and **iter**ative forms. Similarly, we use **GNN-HF** to denote **GNN** with **H**igh-pass **F**iltering graph convolutional kernels.

GNN-HF-closed Using the closed-form propagation matrix in Eq. (39), we define the following new graph neural networks with closed-form propagation mechanism:

$$\begin{aligned} \mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; \infty)_{HF-closed} \\ &= \left\{ (\beta + 1/\alpha)\mathbf{I} + (1 - \beta - 1/\alpha)\hat{\tilde{\mathbf{A}}} \right\}^{-1} \{\mathbf{I} + \beta\tilde{\mathbf{L}}\}\mathbf{H}, \\ &\textit{and} \quad \mathbf{H} = f_\theta(\mathbf{X}). \end{aligned} \tag{41}$$

Note that $\beta \in (0, \infty)$ and $\alpha \in (0, 1]$. By applying the propagation matrix $\{(\beta + 1/\alpha)\mathbf{I} + (1 - \beta - 1/\alpha)\hat{\tilde{\mathbf{A}}}\}^{-1}$ directly on both $\mathbf{H}$ and $\tilde{\mathbf{L}}\mathbf{H}$ matrix, then we can get the representation encoding feature information from both original and high-frequency spaces.

GNN-HF-iter Using the iterative propagation mechanism in Section 4.2.3, we have a deep and computationally efficient graph neural networks with $\beta \in (0, \infty)$ and $\alpha \in (0, 1]$.

$$\begin{aligned} \mathbf{Z} &= \textbf{PROPAGATE}(\mathbf{X}; \mathcal{G}; K)_{HF-iter} \\ &= \left\langle \frac{\alpha\beta - \alpha + 1}{\alpha\beta + 1}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha}{\alpha\beta + 1}\mathbf{H} + \frac{\alpha\beta}{\alpha\beta + 1}\tilde{\mathbf{L}}\mathbf{H} \right\rangle_K, \\ \mathbf{Z}^{(0)} &= \frac{1}{\alpha\beta + 1}\mathbf{H} + \frac{\beta}{\alpha\beta + 1}\tilde{\mathbf{L}}\mathbf{H}, \quad \textit{and} \quad \mathbf{H} = f_\theta(\mathbf{X}). \end{aligned} \tag{42}$$

We directly use the K-layer output as the propagation results. Similarly, this iterative propagation mechanism can be viewed as layer-wise $\hat{\tilde{\mathbf{A}}}$ based neighborhood aggregation, and residual connection on both feature matrix $\mathbf{H}$ and high-frequency filtered feature matrix $\tilde{\mathbf{L}}\mathbf{H}$. And we also decouple the layer-wise transformation and aggregation process during propagation. GNN-HF-iter and GNN-HF-closed have the following relation:

Theorem 4.4. *When $K \to \infty$, deep GNN-HF-iter converges to GNN-HF-closed with the same propagation result as Eq. (39).*

Proof. Please refer to Appendix A.2. □

5 SPECTRAL EXPRESSIVE POWER ANALYSIS

In this section, we study several propagation mechanisms in graph spectral domain to examine their expressive power. A polynomial filter of order K on a graph signal $\mathbf{X} \in \mathbb{R}^{n \times f}$ with f input channels is defined as $\left(\sum_{k=0}^{K} \theta_k \tilde{\mathbf{L}}^k\right)\mathbf{X}$, where θ_k is the corresponding polynomial coefficients. Similar with [4], we also assume that the graph signal $\mathbf{X}$ is non-negative and $\mathbf{X}$ can be converted into the input signal $\mathbf{H}$ under linear transformation. We aim to compare the polynomial coefficients θ_k for different GNNs and show that GNN-LF/HF with K order flexible polynomial filter coefficients have better spectral expressive power. For concision, we mainly analyze the filter coefficients of GNN-LF, and the spectral analysis of SGC, PPNP, GNN-HF is in Appendix A.3.

5.1 Filter Coefficient Analysis

Analysis of GNN-LF. From the analysis in Theorem 4.2, we have the expanded propagation result of GNN-LF-iter in Eq. (35), which has been proved to converge to the propagation result of

GNN-LF-closed with $K \to \infty$. Taking this propagation result for analysis, we have the following filtering expression when $K \to \infty$:

$$\begin{aligned}\mathbf{Z} =& \Big\{\alpha \sum_{i=0}^{K-1} \big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^i \hat{\tilde{\mathbf{A}}}^i\Big\}\Big\{\frac{\mu}{1+\alpha\mu-\alpha}\mathbf{X} + \frac{1-\mu}{1+\alpha\mu-\alpha}\hat{\tilde{\mathbf{A}}}\mathbf{X}\Big\} \\ =& \frac{\alpha\mu}{1+\alpha\mu-\alpha}\Big\{\sum_{i=0}^{K-1}\big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^i \hat{\tilde{\mathbf{A}}}^i\Big\}\mathbf{X} + \frac{\alpha-\alpha\mu}{1+\alpha\mu-2\alpha}\cdot \\ & \Big\{\sum_{i=1}^{K}\big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^i \hat{\tilde{\mathbf{A}}}^i\Big\}\mathbf{X} \\ =& \frac{\alpha\mu}{1+\alpha\mu-\alpha}\Big\{\sum_{i=0}^{K-1}\big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^i (\mathbf{I}-\tilde{\mathbf{L}})^i\Big\}\mathbf{X} + \frac{\alpha-\alpha\mu}{1+\alpha\mu-2\alpha}\cdot \\ & \Big\{\sum_{i=1}^{K}\big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^i (\mathbf{I}-\tilde{\mathbf{L}})^i\Big\}\mathbf{X}.\end{aligned} \tag{43}$$

Expand the above equation, then the filter coefficients on $\tilde{\mathbf{L}}^k (k \in [0, K])$ can be summarized into the following forms:

1) *Filter coefficients for* $\tilde{\mathbf{L}}^0$:

$$\begin{aligned}\theta_0 &= \frac{\alpha\mu(1+\alpha\mu-2\alpha)}{(1+\alpha\mu-\alpha)^2} + \frac{(\alpha-\alpha\mu)(1+\alpha\mu-2\alpha)^{K-1}}{(1+\alpha\mu-\alpha)^K} + \sum_{j=1}^{K-1}\delta_j\binom{j}{0}, \\ \delta_j &= \Big\{\frac{\alpha\mu}{1+\alpha\mu-\alpha} + \frac{\alpha-\alpha\mu}{1+\alpha\mu-2\alpha}\Big\}\big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^j.\end{aligned} \tag{44}$$

2) *Filter coefficients for* $\tilde{\mathbf{L}}^k, k \in [1, K-1]$:

$$\begin{aligned}\theta_k &= \sum_{j=k}^{K}\delta_j(-1)^k\binom{j}{k}, \\ \delta_j &= \Big\{\frac{\alpha\mu}{1+\alpha\mu-\alpha} + \frac{\alpha-\alpha\mu}{1+\alpha\mu-2\alpha}\Big\}\big(\frac{1+\alpha\mu-2\alpha}{1+\alpha\mu-\alpha}\big)^j.\end{aligned} \tag{45}$$

3) *Filter coefficients for* $\tilde{\mathbf{L}}^K$:

$$\theta_K = \frac{(\alpha-\alpha\mu)(1+\alpha\mu-2\alpha)^{K-1}}{(1+\alpha\mu-\alpha)^K}(-1)^K\binom{K}{K}. \tag{46}$$

From the above analysis result on GNN-LF, we find the expression forms of filter coefficients depend on different k and are determined by two adjustable factors α and μ, which improve the expressive power of the spectral filters and further alleviate the over-smoothing problem.

Analysis of SGC/PPNP/GNN-HF. Note that the analysis of GNN-HF is similar with that of GNN-LF, for concision, we show them in Appendix A.3.

5.2 Discussion on Expressive Power

As [17, 38] point out, the reason for the over-smoothing problem is that typical GCN converges to the limit distribution of random walk which is isolated from the input feature and makes node representations inseparable as the number of layer increases. [4] also gives another understanding from the view of polynomial filtering coefficient and points out that flexible and arbitrary filter coefficients are essential for preventing over-smoothing.

Table 2: The statistics of the datasets

Dataset	Classes	Nodes	Edges	Features	Train/Val/Test
Cora	7	2708	5429	1433	140/500/1000
Citeseer	6	3327	4732	3703	120/500/1000
Pubmed	3	19717	44338	500	60/500/1000
ACM	3	3025	13128	1870	60/500/1000
Wiki-CS	10	11701	216123	300	200/500/1000
MS Academic	15	18333	81894	6805	300/500/1000

From the filter coefficients shown in Section 5.1 and Appendix A.3, we can find that: 1) SGC or K-layer graph convolutional operations have fixed constant filtering coefficients, which limit the expressive power and further lead to over-smoothing. 2) PPNP has a better filtering expressive ability against SGC (GCN) since the filter coefficients of the order k is changeable along with the factor α. 3) Comparing with PPNP and SGC (GCN), GNN-LF/HF are more expressive under the influence of adjustable factors α, μ or β, which increase the ability to fit arbitrary coefficients of polynomial filter, and help GNN-LF/HF to alleviate the over-smoothing problem.

From the limit distributions of PPNP [14], GNN-LF in Eq. (36), GNN-HF in Eq. (53), we can also find that all of them converge to a distribution carrying information from both input feature and network structure. This property additionally helps to reduce the effects of over-smoothing on PPNP/GNN-LF/GNN-HF even if the number of layers goes to infinity.

6 EXPERIMENTS

6.1 Experimental Setup

Dataset. To evaluate the effectiveness of our proposed GNN-LF/HF, we conduct experiments on six benchmark datasets in Table 2. 1) **Cora, Citeseer, Pubmed** [13]: Three standard citation networks where nodes represent documents, edges are citation links and features are the bag-of-words representation of the document. 2) **ACM** [33]: Nodes represent papers and there is an edge if two paper have same authors. Features are the bag-of-words representations of paper keywords. The three classes are *Database*, *Wireless Communication*, *DataMining*. 3) **Wiki-CS** [20]: A dataset derived from Wikipedia, in which nodes represent CS articles, edges are hyperlinks and different classes mean different branches of the files. 4) **MS Academic** [14]: A co-authorship Microsoft Academic Graph, where nodes represent authors, edges are co-authorships and node features represent keywords from authors' papers.

Baselines. We evaluate the performance of GNN-LF/HF by comparing it with several baselines. 1) Traditional graph learning methods: MLP [23], LP [48]. 2) Spectral methods: ChebNet [5], GCN [13]; 3) Spatial methods: SGC [34], GAT [30], GraphSAGE [9], PPNP [14]. 4) Deep GNN methods: JKNet [38], APPNP [14], IncepGCN [26].

Settings. We implement GNN-LF/HF based on Pytorch [24]. To ensure fair comparisons, we fix the hidden size as 64 for all models. We apply L_2 regularization on the first layer parameter weights, with coefficients of 5e-3 on all datasets except 5e-4 for Wiki-CS. We set the learning rate $lr = 0.01$ for the other datasets except $lr = 0.03$ for Wiki-CS, and set dropout rate $d = 0.5$. We

Table 3: Node classification results (%). We show the average accuracy with uncertainties showing the 95% confidence level calculated by boot-strapping. Bold and underline are used to show the best and the runner-up results.

Model	Dataset					
	Cora	Citeseer	Pubmed	ACM	Wiki-CS	MS Academic
MLP	57.79±0.11	61.20±0.08	73.23±0.05	77.39±0.11	65.66±0.20	87.79±0.42
LP	71.50±0.00	50.80±0.00	72.70±0.00	63.30±0.00	34.90±0.00	74.10±0.00
ChebNet	79.92±0.18	70.90±0.37	76.98±0.16	79.53±1.24	63.24±1.43	90.76±0.73
GAT	82.48±0.31	72.08±0.41	79.08±0.22	88.24±0.38	74.27±0.63	91.58±0.25
GraphSAGE	82.14±0.25	71.80±0.36	79.20±0.27	87.57±0.65	73.17±0.41	91.53±0.15
IncepGCN	81.94±0.94	69.66±0.29	78.88±0.35	87.75±0.61	60.54±1.06	75.45±0.49
GCN	82.41±0.25	70.72±0.36	79.40±0.15	88.38±0.51	71.97±0.51	92.17±0.11
SGC	81.90±0.23	<u>72.21±0.22</u>	78.30±0.14	87.56±0.34	72.43±0.28	88.35±0.36
PPNP	83.34±0.20	71.73±0.30	80.06±0.20	89.12±0.17	74.53±0.36	92.27±0.23
APPNP	83.32±0.42	71.67±0.48	80.05±0.27	89.04±0.21	74.30±0.50	92.25±0.18
JKNet	81.19±0.49	70.69±0.88	78.60±0.25	88.11±0.36	60.90±0.92	87.26±0.23
GNN-LF-closed	83.70±0.14	71.98±0.33	80.34±0.18	89.43±0.20	**75.50±0.56**	**92.79±0.15**
GNN-LF-iter	83.53±0.24	71.92±0.24	80.33±0.20	89.37±0.40	<u>75.35±0.24</u>	<u>92.69±0.20</u>
GNN-HF-closed	**83.96±0.22**	**72.30±0.28**	<u>80.41±0.25</u>	<u>89.46±0.30</u>	74.92±0.45	92.47±0.23
GNN-HF-iter	<u>83.79±0.29</u>	72.03±0.36	**80.54±0.25**	**89.59±0.31**	74.90±0.37	92.51±0.16

use the validation set for early stopping with a patience of 100 epochs. We fix 10 propagation depth for the two iterative version of GNN-LF/HF. Note that for APPNP and PPNP, all the settings are consistent with the above descriptions. As for ChebNet, GCN, GAT, SGC and GraphSAGE, we use the implementations of DGL [32][1]. For JKNet and IncepGCN, we use the implementation in [26][2]. We try to turn all hyperparameters reasonably to get the best performance, some models achieve better results than original reports. For JKNet, IncepGCN and SGC, we choose the best results of them with no more than 10 propagation depth. We conduct 10 runs on all datasets with the fixed training/validation/test split, where 20 nodes per class are used for training and 500/1000 nodes are used for val/test. For cora/citeseer/pubmed datasets, we follow the dataset splits in [39].

6.2 Node Classification

We evaluate the effectiveness of GNN-LF/HF against several state-of-the-art baselines on semi-supervised node classification task. We use accuracy (ACC) metric for evaluation, and report the average ACC with uncertainties showing the 95% confidence level calculated by bootstrapping in Table 3. We have the following observations:

1) GNN-LF and GNN-HF consistently outperform all the state-of-the-art baselines on all datasets. The best and the runner-up results are always achieved by GNN-LF/HF, which demonstrates the effectiveness of our proposed model. From the perspective of the unified objective framework, it is easy to check that GNN-LF/HF not only keep the representation same with the original features, but also consider capturing their similarities based on low-frequency or high-frequency information. These two relations are balanced so as to extract more meaningful signals and thus perform better.

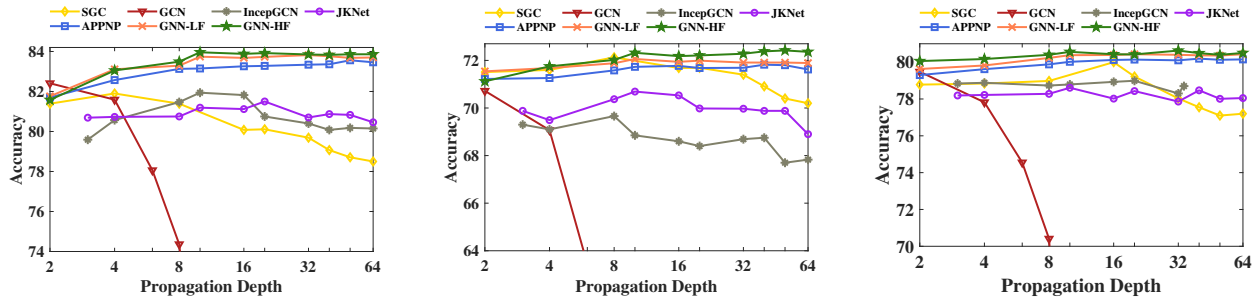
2) From the results of the closed and iterative versions of GNN-LF/HF, we can see that using 10 propagation depth for GNN-LF-iter/GNN-HF-iter is able to effectively approximate the GNN-LF-closed/GNN-HF-closed. As for performance comparisons between GNN-LF and GNN-HF, we find that it is hard to determine which is the best, since which filter works better may depend on the characteristic of different datasets. But in summary, flexibly and comprehensively considering multiple information in a GNN model can always achieve satisfactory results on different networks.

3) In addition, PPNP/APPNP always perform better than GCN/SGC since their objective also considers a fitting term to help find important information from features during propagation. On the other hand, APPNP outperforms JKNet mainly because that its propagation process takes full advantage of the original features and APPNP even decouples the layer-wise non-linear transformation operations [17] without suffering from performance degradation. Actually, the above differences of models and explanations for results can be easily drawn from our unified framework.

6.3 Propagation Depth Analysis

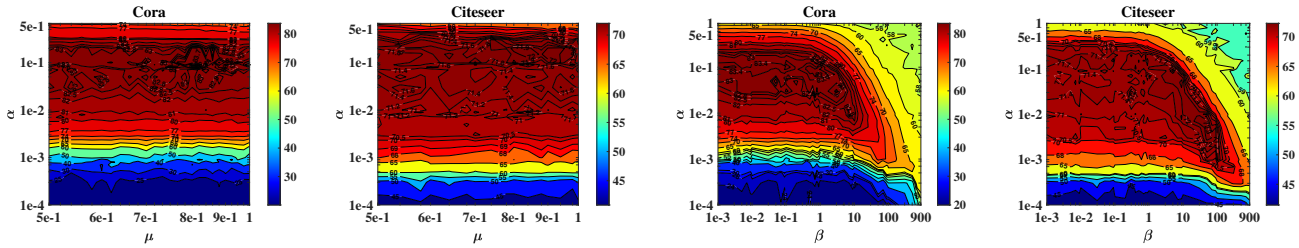
Because our proposed GNN-LF/HF flexibly consider extra filtering information during propagation and have high expressive power, here we further conduct experiments on GNN-LF/HF and other shallow/deep models with different propagation depths using three datasets. For all the models, we set the hidden size as 64 and tune

[1] https://github.com/dmlc/dgl
[2] https://github.com/DropEdge/DropEdge



(a) **Cora** (b) **Citeseer** (c) **Pubmed**

Figure 1: Analysis of propagation depth.



(a) **GNN-LF** (b) **GNN-HF**

Figure 2: Model analysis of GNN-LF/HF on Cora and Citeseer.

hyperparameters reasonably to get the best performance. Note that because of the specific architecture of JKNet and IncepGCN in [26], they are analyzed from 3 layers and IncepGCN on pubmed dataset faces the out of memory problem when the depth is bigger than 34. Figure 1 shows the accuracy with different propagation depths, and we have the following observations:

GNN-LF/HF and APPNP greatly alleviate the over-smoothing problem, since the performance does not drop when the depth increases. Furthermore, GNN-LF/HF are more flexible and more expressive with higher results under the influence of extra graph filters and three adjustable impact factors α, μ and β. As analyzed before, the polynomial filter coefficients of GNN-LF/HF are further more expressive and flexible than APPNP, GCN or SGC, which is useful for mitigating over-smoothing problem. Accuracy breaks down seriously for GCN while it drops a little bit slowly for SGC, but both GCN and SGC face the over-smoothing problem since the fixed polynomial filter coefficients limit their expressive power. As for JKNet/IncepGCN, they are deep GNNs to alleviate over-smoothing problem but still have to face performance degradation when the propagation depth increases.

6.4 Model Analysis

In this section, we analyze the performance of GNN-LF/HF with different impact factors: teleport probability α, balance coefficient μ and balance coefficient β. In general, α adjusts the regularization term weight and has an effect on structural information during propagation; μ and β focus on adjusting the balance weights between different filters and have effects on feature information during propagation.

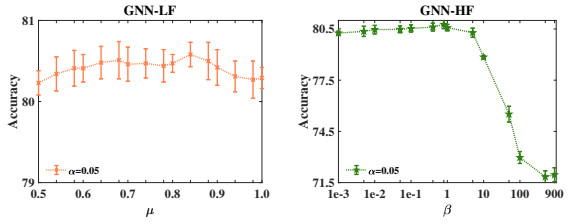


Figure 3: Impact factor analysis with fixed α on Pubmed.

We carefully adjust the value of three impact factors with GNN-LF-closed/GNN-HF-closed models on cora and citeseer datasets, and draw the contour map for accuracy in Figure 2. As can be seen: 1) For GNN-LF, α plays a more dominant influence than μ. The classification accuracy exactly increases as α becomes larger, and with the continuous increase of α, the results begin to drop. Generally, the best performance can be achieved when $\alpha \in [1e-2, 5e-1]$. On the other hand, the accuracy is relatively stable with different μ, and generally speaking, μ is with a suitable range around [0.6, 0.9]. 2) For GNN-HF, α and β both play dominant influence, the suitable weight range for α is also [1e-2, 5e-1] and lager β may result in performance degradation. In general, our proposed GNN-LF/HF achieve stable and excellent performance within a wide changing range of these impact factors α and μ or β.

We then analyze the influence of balance coefficients μ and β with the fixed α on pubmed dataset, shown in Figure 3, and we have similar conclusions: For GNN-LF, the performance first increases stably as μ grows and then may show a slight drop after a certain threshold. The appropriate range for best performance is [0.6, 0.9].

For GNN-LF, the classification accuracy first increases and then drops with the rise of μ after a certain threshold.

7 CONCLUSION

The intrinsic relation for the propagation mechanisms of different GNNs is studied in this paper. We establish the connection between different GNNs and a flexible objective optimization framework. The proposed unified framework provides a global view on understanding and analyzing different GNNs, which further enables us to identify the weakness of current GNNs. Then we propose two novel GNNs with adjustable convolutional kernels showing low-pass and high-pass filtering capabilities, and their excellent expressive power is analyzed as well. Extensive experiments well demonstrate the superior performance of these two GNNs over the state-of-the-art models on real world datasets.

8 ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. U20B2045, U1936104, 61972442, 61772082, 62002029), and the National Key Research and Development Program of China (2018YFB1402600). Houye Ji's research is supported by the BUPT Excellent Ph.D. Students Foundation (No. CX2020311). All opinions, findings, conclusions and recommendations are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *ICML*. 21–29.
[2] Gregor Bachmann, Gary Becigneul, and Octavian Ganea. 2020. Constant Curvature Graph Convolutional Networks. In *ICML*. 486–496.
[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.
[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*. 1725–1735.
[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*. 3844–3852.
[6] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A Fair Comparison of Graph Neural Networks for Graph Classification. In *ICLR*.
[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW*. 417–426.
[8] Hongyang Gao, Yongjun Chen, and Shuiwang Ji. 2019. Learning Graph Pooling and Hybrid Convolutional Operations for Text Representations. In *WWW*. 2743–2749.
[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.
[10] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. 2020. Measuring and Improving the Use of Graph Information in Graph Neural Networks. In *ICLR*.
[11] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.
[12] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph Structure Learning for Robust Graph Neural Networks. In *KDD*. 66–74.
[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[14] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.
[15] Kwei Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. 2020. PolicyGNN: Aggregation Optimization for Graph Neural Networks. In *KDD*. 461–471.
[16] Qimai Li, Zhichao Han, and Xiaoming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. 3538–3545.
[17] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards Deeper Graph Neural Networks. In *KDD*. 338–348.
[18] Andreas Loukas. 2020. What graph neural networks cannot learn: depth vs width. In *ICLR*.
[19] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2020. A Unified View on Graph Neural Networks as Graph Signal Denoising. *arXiv preprint arXiv:2010.01777* (2020).
[20] Péter Mernyei and Catalina Cangea. 2020. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. *arXiv preprint arXiv:2007.02901* (2020).
[21] Hoang Nt and Takanori Maehara. 2019. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters. *arXiv preprint arXiv:1905.09550* (2019).
[22] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*.
[23] S.K. Pal and S. Mitra. 1992. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks* 3, 5 (1992), 683–697.
[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).
[25] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. 1150–1160.
[26] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.
[27] Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. 2019. An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition. In *CVPR*. 1227–1236.
[28] Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to Hash with Graph Neural Networks for Recommender Systems. In *WWW*. 1988–1998.
[29] Damien Teney, Lingqiao Liu, and Anton van den Hengel. 2017. Graph-Structured Representations for Visual Question Answering. In *CVPR*. 3233–3241.
[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *ICLR*.
[31] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: a Deep Attentional Embedding approach. In *IJCAI*. 3670–3676.
[32] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).
[33] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[34] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*. 6861–6871.
[35] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks* (2020), 1–21.
[36] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph Wavelet Neural Network. In *ICLR*.
[37] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks. In *ICLR*.
[38] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*. 5449–5458.
[39] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*. 40–48.
[40] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *NeurIPS*. 9240–9251.
[41] Zhitao Ying, Ines Chami, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic Graph Convolutional Neural Networks. In *NeurIPS*. 4869–4880.
[42] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. XGNN: Towards Model-Level Explanations of Graph Neural Networks. In *KDD*. 430–438.
[43] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *KDD*. 793–803.
[44] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.
[45] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. *arXiv: Learning* (2018).
[46] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *KDD*. 1399–1407.
[47] Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph Neural Networks with Generated Parameters for Relation Extraction. In *ACL*. 1331–1339.
[48] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. 2005. Semi-supervised learning with graphs. (2005).

A PROOFS AND ANALYSIS

A.1 Proof of Theorem 3.2

With the objective Eq. (12), we have the following closed-form solution:

$$\mathbf{Z} = (\mathbf{I} + \tilde{\mathbf{L}})^{-1}\mathbf{H}. \tag{47}$$

Similar to the analysis in [21], we can decompose the matrix $(\mathbf{I}+\tilde{\mathbf{L}})^{-1}$ and get the first-order truncated form as:

$$(\mathbf{I} + \tilde{\mathbf{L}})^{-1} \approx \mathbf{I} - \tilde{\mathbf{L}} = \hat{\tilde{\mathbf{A}}}. \tag{48}$$

In this way, we have the first-order approximation :

$$\mathbf{Z} = \hat{\tilde{\mathbf{A}}}\mathbf{H} = \hat{\tilde{\mathbf{A}}}\mathbf{X}\mathbf{W}. \tag{49}$$

At this point, we provide another explanation on operation using first-order approximation based on the framework.

A.2 Proof of Theorem 4.4

GNN-HF-iter uses the following iteration equation:

$$\begin{aligned}\mathbf{Z}^{(0)} &= \frac{1}{\alpha\beta+1}\mathbf{H} + \frac{\beta}{\alpha\beta+1}\tilde{\mathbf{L}}\mathbf{H},\\ \mathbf{Z}^{(k)} &= \frac{\alpha\beta-\alpha+1}{\alpha\beta+1}\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)} + \frac{\alpha}{\alpha\beta+1}\mathbf{H} + \frac{\alpha\beta}{\alpha\beta+1}\tilde{\mathbf{L}}\mathbf{H}.\end{aligned} \tag{50}$$

The closed form of GNN-HF-closed with the same $\mathbf{H} = f_\theta(\mathbf{X})$ is:

$$\mathbf{Z} = \left\{(\beta+1/\alpha)\mathbf{I} + (1-\beta-1/\alpha)\hat{\tilde{\mathbf{A}}}\right\}^{-1}\{\mathbf{I}+\beta\tilde{\mathbf{L}}\}\mathbf{H}. \tag{51}$$

After the K-layer propagation using GNN-HF-iter, the corresponding output result $\mathbf{Z}$ can be written as:

$$\begin{aligned}\mathbf{Z} =& \left\{(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^K\hat{\tilde{\mathbf{A}}}^K + \alpha\sum_{i=0}^{K-1}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^i\hat{\tilde{\mathbf{A}}}^i\right\}\left\{\frac{1}{\alpha\beta+1}\mathbf{H}\right.\\ &\left.+\frac{\beta}{\alpha\beta+1}\tilde{\mathbf{L}}\mathbf{H}\right\},\end{aligned} \tag{52}$$

where $\beta \in (0,\infty)$, $\alpha \in (0,1]$ and $|\frac{\alpha\beta-\alpha+1}{\alpha\beta+1}| < 1$. Similar with the proof process of GNN-LF-iter in Theorem 4.2, we have the converging result as:

$$\mathbf{Z} = \left\{(\beta+1/\alpha)\mathbf{I} + (1-\beta-1/\alpha)\hat{\tilde{\mathbf{A}}}\right\}^{-1}\{\mathbf{I}+\beta\tilde{\mathbf{L}}\}\mathbf{H}, \tag{53}$$

which exactly is the Eq. (51) for calculating GNN-HF-closed.

A.3 Expressive Power Analysis

Analysis of SGC. As [34] points out, the K-layer graph convolutional operations or simplified graph convolutional operations act as the spectral polynomial filter of order K with fixed coefficients. [4] proves that such fixed coefficients limit the expressive power of GCN and thus leads to over-smoothing. The K-order polynomial filter on graph signal $\mathbf{X}$ is:

$$\mathbf{Z}^{(K)} = \hat{\tilde{\mathbf{A}}}^K\mathbf{X} = \left(\mathbf{I}-\tilde{\mathbf{L}}\right)^K\mathbf{X}. \tag{54}$$

By calculating the expansion of Eq. (54), we can conclude the k-th polynomial filtering term, denoted by $\theta_k\tilde{\mathbf{L}}^k$.Then for the filter coefficients on $\tilde{\mathbf{L}}^k, k \in [0,K]$, we have $\theta_k = (-1)^k\binom{K}{k}$, which is a fixed constant for any k.

Analysis of PPNP. As proved in [14], PPNP or K-order APPNP ($K \to \infty$) has the following expressive power:

$$\mathbf{Z}^{(K)} = \left\{(1-\alpha)^K\hat{\tilde{\mathbf{A}}}^K + \alpha\sum_{i=0}^{K-1}(1-\alpha)^i\hat{\tilde{\mathbf{A}}}^i\right\}\mathbf{X}. \tag{55}$$

Since $(1-\alpha)^\infty \to 0$, we can rewrite it using the normalized graph Laplacian $\tilde{\mathbf{L}}$ as:

$$\mathbf{Z}^{(K)} = \alpha\sum_{i=0}^{K-1}\left(1-\alpha\right)^i\hat{\tilde{\mathbf{A}}}^i\mathbf{X} = \alpha\sum_{i=0}^{K-1}\left(1-\alpha\right)^i\left(\mathbf{I}-\tilde{\mathbf{L}}\right)^i\mathbf{X}. \tag{56}$$

Then we can calculate the expansion of Eq. (56) and conclude the k-th polynomial filtering term, denoted by $\theta_k\tilde{\mathbf{L}}^k$. Then for the filter coefficients on $\tilde{\mathbf{L}}^k, k \in [0,K-1]$, we have:

$$\theta_k = \alpha\sum_{i=k}^{K-1}(1-\alpha)^i(-1)^k\binom{i}{k}. \tag{57}$$

Analysis of GNN-HF. Similarly, taking the propagation result in Eq. (52) with $K \to \infty$, we can also have the corresponding filtering expression of output $\mathbf{Z}$:

$$\begin{aligned}\mathbf{Z} =& \left\{\alpha\sum_{i=0}^{K-1}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^i\hat{\tilde{\mathbf{A}}}^i\right\}\left\{\frac{1}{\alpha\beta+1}\mathbf{X} + \frac{\beta}{\alpha\beta+1}\tilde{\mathbf{L}}\mathbf{X}\right\}\\ =& \frac{\alpha(\beta+1)}{\alpha\beta+1}\left\{\sum_{i=0}^{K-1}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^i\hat{\tilde{\mathbf{A}}}^i\right\}\mathbf{X} - \frac{\alpha\beta}{\alpha\beta-\alpha+1}\cdot\\ &\left\{\sum_{i=1}^{K}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^i\hat{\tilde{\mathbf{A}}}^i\right\}\mathbf{X}\\ =& \frac{\alpha(\beta+1)}{\alpha\beta+1}\left\{\sum_{i=0}^{K-1}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^i\left(\mathbf{I}-\tilde{\mathbf{L}}\right)^i\right\}\mathbf{X} - \frac{\alpha\beta}{\alpha\beta-\alpha+1}\cdot\\ &\left\{\sum_{i=1}^{K}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^i\left(\mathbf{I}-\tilde{\mathbf{L}}\right)^i\right\}\mathbf{X}.\end{aligned} \tag{58}$$

Then for the filter coefficients on $\tilde{\mathbf{L}}^k, k \in [0,K]$, we have the following conclusions:

1) *Filter coefficients for* $\tilde{\mathbf{L}}^0$:

$$\begin{aligned}\theta_0 &= \frac{\alpha(\beta+1)(\alpha\beta-\alpha+1)}{(\alpha\beta+1)^2} - \frac{\alpha\beta(\alpha\beta-\alpha+1)^{K-1}}{(\alpha\beta+1)^K} + \sum_{j=1}^{K-1}\delta_j\binom{j}{0},\\ \delta_j &= \{\frac{\alpha(\beta+1)}{\alpha\beta+1} - \frac{\alpha\beta}{\alpha\beta-\alpha+1}\}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^j.\end{aligned} \tag{59}$$

2) *Filter coefficients for* $\tilde{\mathbf{L}}^k, k \in [1,K-1]$:

$$\begin{aligned}\theta_0 &= \sum_{j=i}^{K}\delta_j(-1)^i\binom{j}{i},\\ \delta_j &= \{\frac{\alpha(\beta+1)}{\alpha\beta+1} - \frac{\alpha\beta}{\alpha\beta-\alpha+1}\}(\frac{\alpha\beta-\alpha+1}{\alpha\beta+1})^j.\end{aligned} \tag{60}$$

3) *Filter coefficients for* $\tilde{\mathbf{L}}^K$:

$$\theta_K = -\frac{\alpha\beta(\alpha\beta-\alpha+1)^{K-1}}{(\alpha\beta+1)^K}(-1)^K\binom{K}{K}. \tag{61}$$