Difference Residual Graph Neural Networks

Liang Yang*
Weihang Peng*
Wenmiao Zhou*
yangliang@vip.qq.com
zhou_wen_miao@163.com
str_mail_id@163.com
Hebei University of Technology
Tianjin, China

Bingxin Niu
Junhua Gu
niubingxin666@163.com
jhgu_hebut@163.com
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China

Chuan Wang
wangchuan@iie.ac.cn
State Key Laboratory of Information Security
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China

Yuanfang Guo†
andyguo@buaa.edu.cn
School of Computer Science and Engineering
Beihang University
Beijing, China

Dongxiao He
hedongxiao@tju.edu.cn
College of Intelligence and Computing
Tianjin University
Tianjin, China

Xiaochun Cao
caoxiaochun@iie.ac.cn
School of Cyber Science and Technology, Shenzhen Campus
Sun Yat-sen University
Shenzhen, China

ABSTRACT

Graph Neural Networks have been widely employed for multimodal fusion and embedding. To overcome over-smoothing issue, residual connections, which are designed for alleviating vanishing gradient problem in NNs, are adopted in Graph Neural Networks (GNNs) to incorporate local node information. However, these simple residual connections are ineffective on networks with heterophily, since the roles of both convolutional operations and residual connections in GNNs are significantly different from those in classic NNs. By considering the specific smoothing characteristic of graph convolutional operation, deep layers in GNNs are expected to focus on the data which can't be properly handled in shallow layers. To this end, a novel and universal Difference Residual Connections (DRC), which feed the difference of the output and input of previous layer as the input of the next layer, is proposed. Essentially, Difference Residual Connections is equivalent to inserting layers with opposite effect (e.g., sharpening) into the network to prevent the excessive effect (e.g., over-smoothing issue) induced by too many layers with the similar role (e.g., smoothing) in GNNs. From the perspective of optimization, DRC is the gradient descent method to minimize an objective function with both smoothing and sharpening terms. The analytic solution to this objective function is determined by both graph topology and node attributes, which theoretically proves that DRC can prevent over-smoothing issue. Extensive experiments demonstrate the superiority of DRC on real networks with both homophily and heterophily, and show that DRC can automatically determine the model depth and be adaptive to both shallow and deep models with two complementary components.

*Three authors contributed equally to this research.
†Corresponding author.

ACM MM '22, October 10–14, 2022, Lisbon, Portugal
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Computing methodologies → Neural networks**;

KEYWORDS

Graph neural networks, residual, smoothing, sharpening

ACM Reference Format:
Liang Yang, Weihang Peng, Wenmiao Zhou, Bingxin Niu, Junhua Gu, Chuan Wang, Yuanfang Guo, Dongxiao He, and Xiaochun Cao. 2018. Difference Residual Graph Neural Networks. In *Proceedings of (ACM MM '22).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph Neural Networks (GNNs) are powerful tools to analyze irregular data by leveraging expressive power of deep learning [35, 41]. They have been widely employed by natural language processing [4], computer vision [5], information retrieval [19] and multimedia [8] for multimodal fusion and embedding [20]. Originated from the graph signal processing in spectral graph theory [28], GNNs are designed from the perspective of either spectral Fourier analysis [13] or spatial message passing [9], and recent progresses attempt to bridge the gap between these two perspectives [2] or unify them from the perspective of numerical optimization [23, 38, 43]. Most GNNs follow the feed-forward structure as classic deep neural networks (NNs) by stacking multiple graph convolutional layers (GCLs).

However, there exist many differences between classic NNs and GNNs. In classic deep NNs, shallow layers extract low-level information from input, while deep layers extract high-level semantic information based on the low-level extracted information. Thus, to improve the expressive power, classic NNs are getting deeper and deeper, from AlexNet [16] and VGG [29], to GoogleNet [30] and ResNet [11]. To overcome the vanishing gradient problem, which

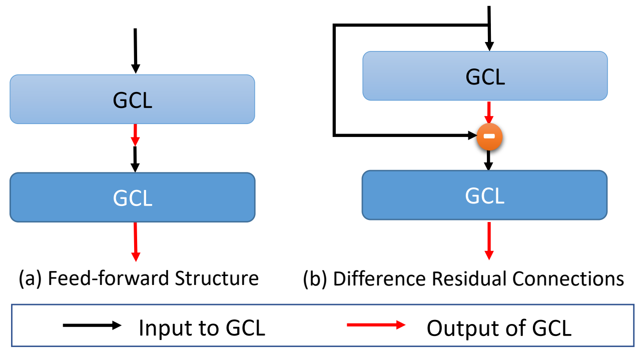

GCL
GCL
GCL
GCL
(a) Feed-forward Structure
(b) Difference Residual Connections
Input to GCL
Output of GCL

Figure 1: Comparison between GNN architectures following the feed-forward strategy and the proposed difference residual connection. GCL denotes graph convolutional layer.

hinders the efficient training of deep model as the depth in classic NNs increases, residual connection is proposed [11, 12]. Different from classic NNs, in GNNs, shallow layers explore local information, while deep layers explore global information for node representation. Due to the smoothing characteristic of graph convolutional operations in GCLs, GNNs often possess serious over-smoothing issue, which leads to the loss of expressive power, by stacking multiple layers [18, 21, 24]. To overcome this issue, residual connections are adopted in GNNs to incorporate local node information, e.g., original node attributes [1, 6, 14, 15, 22, 36, 39]. Therefore, the roles of both layer and residual connections are remarkably different in NNs and GNNs.

These significant differences are the essential reasons of many serious issues in GNNs. 1) The direct feed-forward stacking scheme, i.e., the output of previous layer is directly adopted as the input of next layer, causes the over-smoothing issue, due to the smoothing characteristic of most GCLs. 2) Although some simple residual connections alleviate over-smoothing issue, they fail to perform well on networks with heterophily, where connected nodes possess different attributes and smoothing operation tends to be invalid. Thus, the investigation of the layer stacking scheme in GNNs is motivated.

To overcome the issues mentioned above, a novel universal Difference Residual Connection (DRC) for GNNs is proposed as shown in Figure 1. The key intuition of DRC is to let the deep GCLs process the information which has not been properly handled in shallow GCLs. Specifically, DRC feeds the difference of the input and output of previous layer as the input of the next layer, and combines the outputs from all the layers with learnable weighting parameters. The theoretical analysis to DRC are conducted from two perspectives. First, by analyzing the output of each GCL with DRC, DRC is equivalent to inserting layers with the opposite effect (e.g., sharpening) into the network to prevent the excessive effect (e.g., over-smoothing issue) induced by too many layers with similar role (e.g., smoothing) in GNNs. Second, from the perspective of optimization, DRC is the gradient descent method to minimize an objective function with both smoothing and sharpening terms. The analytic solution to this objective function is determined by both graph topology and node attributes, which theoretically proves that DRC can prevent over-smoothing issue. Besides, experimental evaluations show that DRC possesses two attractive properties: 1) automatic determination of the model depth, 2) adaptive to both shallow and deep models with two complementary components, i.e., difference residual connection and output weights

The main contributions of this paper are summarized as follows:

- We propose a novel and universal difference residual connection (DRC) for graph neural networks.
- We theoretically prove that DRC can overcome over-smoothing via iterative smoothing and sharpening.
- We experimentally show that DRC can automatically determine the model depth and be adaptive to both shallow and deep models with two complementary components.

2 PRELIMINARIES

In this section, the notations are given. Then, several existing GNNs, which are mentioned in this paper, are reviewed. Finally, Graph Representation Learning Framework, which can be used to explain some existing GNNs from optimization perspective, is provided.

2.1 Notations

A graph is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set with $|\mathcal{V}| = N$ and $\mathcal{E}$ is the edge set. Let $A \in \mathbb{R}^{N\times N}$ and $D = diag(d_1, d_2, ..., d_N)$ denote the adjacency matrix and diagonal degree matrix of $\mathcal{G}$, respectively. The graph Laplacian matrix and its normalized form are defined as $L = D-A$ and $\tilde{L} = I-D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, respectively. We define $\hat{L} = I - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$,where $\tilde{A} = A + I$and $\tilde{D} = D + I$. Let $X \in \mathbb{R}^{N\times D}$ denotes the node attribute matrix, where each node v_i is associated with a D-dimensional attribute vector X_i

2.2 Graph Neural Networks

Graph Convolutional Network (GCN) [13] is a simple and widely used model exploiting message passing algorithm. The representation of $(t + 1)^{th}$ layer can be formulated as

$$H^{(t+1)} = \sigma(\hat{A}H^{(t)}W), H^{(0)} = X \quad (1)$$

GCN naturally combines graph structure and node attribute in the convolution. Unfortunately, the most serious issue of GCN is the over-smoothing issue, which is caused by the multiple propagations via stacking multiple graph convolution layers.

JKNet [36] is a deep graph neural network which exploits information from neighborhoods of differing locality and the final representation can be formulated as

$$H = \sum_{t=1}^{T} \alpha_t \tilde{A}^t XW, \quad (2)$$

APPNP [14] is proposed to alleviate the over-smoothing issue in GCN by adding initial residual to each graph convolutional layer and can be formulated as

$$H^{(t+1)} = (1 - \alpha)\tilde{A}H^{(t)} + \alpha X, H^{(0)} = X \quad (3)$$

2.3 Graph Representation Learning Framework

The Graph Representation Learning Framework [43] can be formulated as

$$O = \min_{H} \|\mathbf{H} - \mathbf{X}\|_F^2 + \lambda tr(\mathbf{H}^T \tilde{\mathbf{L}} \mathbf{H}) \quad (4)$$

The first term is a unary constraint, which describes the representation of node v_i should be similar with its original attribute x_i, while the second term is a pairwise constraint, which depicts connected nodes should possess the similar representation.

Furthermore, recent researches have proved that many existing GNNs can be derived from this Graph Representation Learning Framework [38, 43]. Specifically, these GNNs can be induced by numerical optimization methods based on this framework.

3 DIFFERENCE RESIDUAL CONNECTIONS

3.1 Motivations

In this subsection, the rationality of the feed-forward structure adopted in GNNs are investigated. Figure 1(a) shows the data flow of the classic GNNs, such as GCN [13] and GAT [33]. Similar to classic deep feed-forward neural networks, the output of previous GCL (red arrows) is directly fed as the input of next GCL (black arrows). This process can be formulated as

$$H^{(t)} = f_A^t(H^{(t-1)}), \quad (5)$$

where $H^{(t)}$ stands for the output representations of the t^{th} GCL and the initial representation is the original node attribute, i.e., $H^{(0)} = X$. $f_A^t(\cdot)$ represents the t^{th} GCL, which processes the input according to graph topology A. In this process, the output of the $(t-1)^{th}$ GCL $f_A^{t-1}(\cdot)$, i.e. $H^{(t-1)}$ is directly fed as the input to the t^{th} GCL $f_A^t(\cdot)$. In classic deep neural networks, shallow layers extract low-level information, while deep layers extract high-level semantic information based on the low-level information. Different from classic neural networks, in graph neural networks, shallow layers explore local information, while deep layers explore global information for node representation. Thus, it is not appropriate to adopt global information directly for node representation. Furthermore, it may induce the loss of expressive power during feed-forward process, since many GNNs actually learn node representations by smoothing over neighbourhoods. Therefore, it tends to be not rational to directly feed the output of previous GCL to the next.

3.2 Architecture

To overcome the issues mentioned above, a novel Difference Residual Connection (DRC) for GNNs is proposed. The key intuition of DRC is to let the deep GCLs process the information which has not been properly handled in shallow GCLs. To this end, the feed-forward scheme in Eq. (5) is enhanced to

$$H^{(t)} = f_A^t(F^{(t)}), \quad (6)$$

where the input to t^{th} GCL, i.e. $F^{(t)}$, is not directly constrained to the output of previous GCL. Then, the difference between the input and output of previous GCL is utilized to represent the information, which has not been properly handled. Thus, $F^{(t)}$ is set as

$$F^{(t)} = \alpha^{(t)}(F^{(t-1)} - H^{(t-1)}) \quad t > 1, \quad (7)$$

where $F^{(t-1)}$ and $H^{(t-1)}$ denote the input and output of $(t-1)^{th}$ GCL respectively, and $F^{(t-1)} - H^{(t-1)}$ stands for the information which is not properly handled in $(t-1)^{th}$ GCL. The learnable parameter $\alpha^{(t)}$ denotes the importance of this information for the representation learning. The input to the first GCL is original node attribute, i.e., $F^{(1)} = X$. Finally, information handled by all GCLs, i.e., $H^{(t)}$'s, is combined as the final result $H = \sum_t H^{(t)} W^{(t)}$, where $W^{(t)}$ is the learnable mapping function. Note that the adoption of $\alpha^{(t)}$ can also be regarded as controlling the scale of $W^{(t)}$ in order to prevent overfitting as in GCNII [6]. However, different from fixed $\alpha^{(t)}$ in GCNII, $\alpha^{(t)}$ is learnable in DRC. This model is shown in Figure 3(e), where red and black arrows denote the output and input, respectively.

Universality: Following sections will mainly take the standard graph convolutional layer proposed in GCN [13], i.e., $\hat{A}X$, as the GCL and demonstrate the effect of difference residual connection. Essentially, the proposed DRC is a universal framework which can be employed by most existing GNNs.

3.3 Overcome Over-smoothing Issue

In this subsection, we show that Difference Residual Connection (DRC) can overcome the over-smoothing issue.

First, the over-smoothing in GCN is investigated. The essence of Graph Convolutional Network (GCN) [13] is the Laplacian Smoothing [18] or low-passing filtering [34]. From the perspective of Laplacian Smoothing, each Graph Convolutional Layer (GCL) in GCN is averaging node attribute among neighbourhoods. As formulated in [18, 25], GCL processes the input $H^{(t-1)}$ as

$$\begin{aligned} H^{(t)} &= (1-\gamma)H^{(t-1)} + \gamma \tilde{D}^{-1}\tilde{A}H^{(t-1)} \\ &= (I - \gamma\tilde{L})H^{(t-1)}, \end{aligned} \quad (8)$$

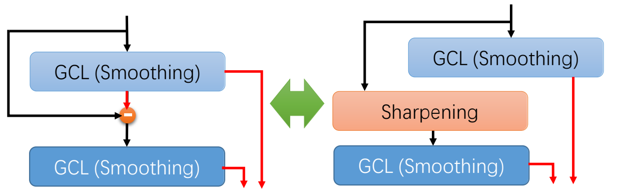
where $\tilde{D} = I + D$, $\tilde{A} = I + A$ and $\tilde{L} = I - \tilde{D}^{-1}\tilde{A}$ are the degree matrix, adjacent matrix and the Laplacian matrix (Random Walk version) with self-loop, respectively. Thus, by ignoring the nonlinear mapping function and collapsing weight matrices as in [34], the output after K GCLs is $H^{(K)} = (I - \gamma\tilde{L})^K X$. As $K \to \infty$, $(I - \gamma\tilde{L})^K X \to \mathbf{1}$ where $\mathbf{1}$ stands for matrix with all elements as 1 [18, 21]. This over-smoothing issue and the loss of expressive power are caused by that all the GCLs perform smoothing via $I - \gamma\tilde{L}$. Thus, to alleviate over-smoothing issue, we need to interfere the successive smoothing process.

By adopting our proposed Difference Residual Connection, Eq. (8) can be reformulated as

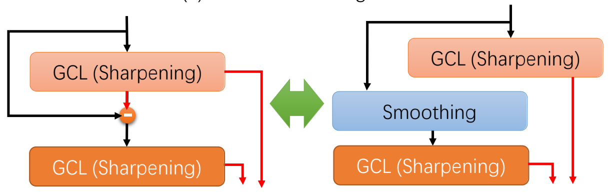
$$\begin{aligned} H^{(t)} &= (1-\gamma)F^{(t)} + \gamma \tilde{D}^{-1}\tilde{A}F^{(t)} \\ &= (I - \gamma\tilde{L})F^{(t)}, \end{aligned} \quad (9)$$

$$\begin{aligned} F^{(t)} &= \alpha^{(t)}(F^{(t-1)} - H^{(t-1)}) \\ &= \alpha^{(t)}(F^{(t-1)} - (I - \gamma\tilde{L})F^{(t-1)}) \\ &= \alpha^{(t)}\gamma\tilde{L}F^{(t-1)}, \end{aligned} \quad (10)$$

where $H^{(t)}$ and $F^{(t)}$ are the output and input of the t^{th} GCL.



(a) GCL as Smoothing



(b) GCL as Sharpening

Figure 2: Equivalence.

Similarly, the output of the $(t+1)^{th}$ GCL is

$$\begin{aligned} H^{(t+1)} &= (I-\gamma\tilde{L})F^{(t+1)} \\ &= (I-\gamma\tilde{L})\alpha^{(t+1)}\gamma\tilde{L}F^{(t)} \\ &= \alpha^{(t+1)}\gamma\tilde{L}(I-\gamma\tilde{L})F^{(t)}. \end{aligned} \tag{11}$$

It can be observed that the effect of $\gamma\tilde{L}$ is opposite to the Laplacian smoothing $I-\gamma\tilde{L}$, which averages the node attribute with its neighbourhood by weighted summation. Actually, $\gamma\tilde{L} = \gamma(I-\tilde{D}^{-1}\tilde{A})$ is known as the Laplacian Sharpening, which removes some attributes from node to make each node different from its neighbourhoods, i.e. sharpening [25, 32]. Therefore, different from vanilla GCN in Eq (8) which successively performs smoothing in each GCL, our proposed DRC in Eq. (11) iteratively performs smoothing and sharpening to prevent over-smoothing. The equivalent process is shown in Figure 2(a). Analogously, if the GCL is designed as a sharpening process, the proposed DRC is equivalent to inserting smoothing layer into the GNNs to prevent over-sharpening as shown in Figure 2(b).

In brief, Difference Residual Connections essentially insert layers with the opposite effect into the GNNs to prevent the excessive effect induced by too many layers with the similar role (smoothing or sharpening). The learnable parameter $\alpha^{(t)}$ determines the importance of the output of t^{th} layer. Thus, DRC enhances the classic GNNs with the ability to handle networks with both homophily and heterophily. The relationship between overcoming over-smoothing issue and handling network with heterophily is also investigated in [37].

3.4 Optimization Perspective

Recently, some efforts have been paid to interprete GNNs from the perspective of numerical optimization [23, 38, 43]. Each graph convolutional layer in GCN [13] can be seen as the gradient descent to minimize the graph Laplacian regularization $tr(H^T\hat{L}H) = \frac{1}{2}\sum_{i,j}\|h_i - h_j\|_2^2$ with node attribute X as the start point, where $tr(\cdot)$ stands for the trace of the matrix and $\hat{L} = I - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Due to the smoothing effect of graph Laplacian regularization, all h_i's will converge to the same value, i.e., over-smoothing issue, when $tr(H^T\hat{L}H)$ achieves the minima 0. From this perspective, APPNP [14] and JKNet [36] actually are the gradient descent of the following two objective functions according to [38], respectively,

$$\begin{aligned} &\min \|H-X\|_F^2 + \beta tr(H^T\hat{L}H), \\ &\min \|H-\hat{A}X\|_F^2 + \beta tr(H^T\hat{L}H), \end{aligned} \tag{12}$$

which are equivalent to preventing over-smoothing issue induced by $tr(H^T\hat{L}H)$ via local information, i.e. original node attribute X in APPNP or low-frequency information AX in JKNet. Although both of them can alleviate over-smoothing issue, they are incapable to handle network with heterophily, because they only balance the local information and smoothing constraints but fail to prevent over-smoothing. To further understand the ability of DRC on both perventing over-smoothing and handling networks with heterophily, we provide the following theorem.

Theorem 3.1. *If the graph convolutional layer function is $f_A^t(F^{(t)}) = \hat{A}F^{(t)}$ (smoothing effect) or $f_A^t(F^{(t)}) = \hat{L}F^{(t)}$ (sharpening effect) where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and $\hat{L} = I - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ are the symmetric normalized adjacency and Laplacian matrices, the Difference Residual Connections in Eqs. (6) and (7) are the gradient descent methods to minimize the following two objective functions, respectively*

$$\begin{aligned} &\min_H \|H-\hat{A}X\|_F^2 + \beta tr(H^T\hat{A}H), \\ &\min_H \|H-\hat{L}X\|_F^2 + \beta tr(H^T\hat{L}H), \end{aligned} \tag{13}$$

with the analytic solution as

$$H = (I+\beta\hat{A})^{-1}\hat{A}X, \quad H = (I+\beta\hat{L})^{-1}\hat{L}X. \tag{14}$$

Proof. To show the equivalence between the Difference Residual Connections in Eqs. (6) (7) with the graph convolutional layer function as $f_A^t(F^{(t)}) = \hat{A}F^{(t)}$ and Eq. (13), we prove they possess the same solution. Let O denote the objective function in Eq. (13). We have

$$O = \min_H \|H-\hat{A}X\|_F^2 + \beta tr(H^T\hat{A}H), \tag{15}$$

with the gradient as

$$\frac{\partial O}{\partial H} = \frac{\partial\|H-\hat{A}X\|_F^2}{\partial H} + \beta\frac{\partial tr(H^T\hat{A}H)}{\partial H}.$$

To minimize Eq. (15), we let its derivative equal to zero, i.e., $\frac{\partial O}{\partial H} = 0$, then

$$\frac{\partial\|H-\hat{A}X\|_F^2}{\partial H} + \beta\frac{\partial tr(H^T\hat{A}H)}{\partial H} = 0. \tag{16}$$

That is,

$$H - \hat{A}X + \beta\hat{A}H = 0. \tag{17}$$

The solution to Eq. (17) is

$$H = (I+\beta\hat{A})^{-1}\tilde{A}X. \tag{18}$$

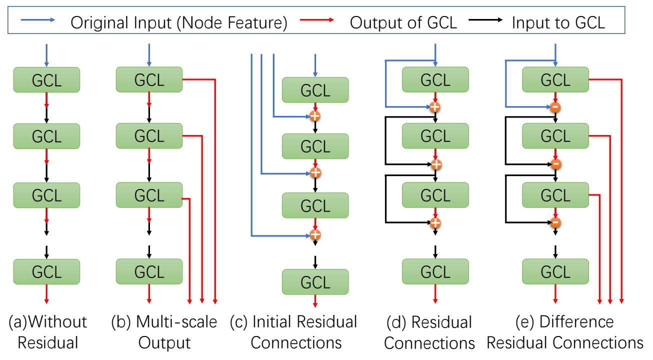

Original Input (Node Feature)
Output of GCL
Input to GCL
GCL
(a)Without Residual
(b) Multi-scale Output
(c) Initial Residual Connections
(d) Residual Connections
(e) Difference Residual Connections

Figure 3: Architecture comparison of different residual connections in GNNs.

This solution can be expressed as the following summation of series

$$\begin{aligned} H &= (I + \beta\hat{A})^{-1}\hat{A}X \\ &= ((1+\beta)I - \beta\hat{L})^{-1}\hat{A}X \\ &\approx \frac{1}{1+\beta}\hat{A}X + \frac{\beta}{(1+\beta)^2}\hat{L}\hat{A}X + \ldots + \frac{\beta^k}{(1+\beta)^{k+1}}\hat{L}^k\hat{A}X \\ &= \eta_1\hat{A}X + \eta_2\hat{L}\hat{A}X + \eta_3\hat{L}^2\hat{A}X + \ldots + \eta_{k+1}\hat{L}^k\hat{A}X \\ &= \sum_{i=1}^{k+1}\eta_i\hat{L}^{i-1}\hat{A}X. \end{aligned} \tag{19}$$

In the next, we show the Difference Residual Connections in Eqs. (6) (7) with the graph convolutional layer function as $f_A^t(F^{(t)}) = \hat{A}F^{(t)}$ possess the same expression as Eq. (19).

Similar with the relation between inputs of t^{th} and $(t+1)^{th}$ layers mentioned in Eq. (10), the relation of outputs of t^{th} and $(t+1)^{th}$ layers can be deduced as follows.

$$H^{(t+1)} = \alpha^{(t+1)}\gamma\hat{L}H^{(t)}. \tag{20}$$

Note that $H^{(t)}$ represents the output of t^{th} layer and node representation $H = H^{(1)} + H^{(2)} + \cdots + H^{(k+1)}$, which denotes the summation of each layer. With $\gamma = 1$ and $H^{(1)} = \eta_1\hat{A}X$, we can calculate the final result of node representation as follow.

$$H = \sum_{i=1}^{k}\eta_i\hat{L}^{i-1}\tilde{A}X. \tag{21}$$

It can be observed that the final node representations of DRC in Eq. (21) is the same as the solution of Eq. (19) to objective function Eq. (13). Thus, DRC is equivalent to the gradient methods of the objective function Eq. (13). □

Here, graph convolutional layer function with smoothing effect, i.e., $f_A^t(F^{(t)}) = \hat{A}F^{(t)}$ is taken for interpretation. The first term in the objective function is to make the node representation similar to the smoothed node attribute, i.e., $\hat{A}X$. The second term $tr(H^T\hat{A}H)$ possesses the opposite effect to Laplacian regularization $tr(H^T\hat{L}H)$, since $\hat{L} = I - \hat{A}$. Thus, the second term plays the role of Laplacian sharpening. By combining both terms, smoothing and sharpening effects are balanced. Note that Eq. (14) is the solution to both objective function in Eq. (13) and the Difference Residual Connection with graph convolutional layer function as $f_A^t(F^{(t)}) = \hat{A}F^{(t)}$. It means as the number of layers tends to infinity, the learned node representation $H = (I + \beta\hat{A})^{-1}\hat{A}X$ is determined by both topology A and original node attribute X. Therefore, Difference Residual Connection can prevent over-smoothing issue. Analogously, for graph convolutional layer function with sharpening effect, i.e., $f_A^t(F^{(t)}) = \hat{L}F^{(t)}$, Difference Residual Connection can prevent over-sharpening issue.

Table 1: Datasets statistics

Dataset	Nodes	Edges	Features	Classes	Homophily
Cora	2,708	5,429	1,433	7	0.83
Citeseer	3,327	4,732	3,703	6	0.71
Pubmed	19,717	44,338	500	3	0.79
Chameleon	2,277	36,101	2,325	5	0.25
Squirrel	5,201	217,073	2,089	5	0.22
Actor	7,600	33,544	931	5	0.24

3.5 Comparisons

In Figure 3, some representative architectures of GNNs are provided. These architectures focus on the flow of the input and output, but ignore the specific structure of Graph Convolutional Layer (GCL). In the figure, blue arrows denote the input node attributes, while black and red arrows represent the input and output of the GCL, respectively.

- Figure 3(a) stands for the classic Graph Convolutional Networks without any branch connections, such as GCN [13] and GAT [33]. The over-smoothing issue is its fatal drawback.
- Figures 3(b) and (c) represent the architectures of GNNs which combine multi-scale topology information [1, 14, 15, 22, 36] and initial node attribute [6, 14], respectively. Their abilities on alleviating over-smoothing can be mainly attributed to the integration of the local information in shallow GCL or node attribute which is ignored by architecture in Figure 3(a). However, it is incapable to handle network with heterophily by just integrating local information.
- Figure 3(d) shows the GNNs with residual connections, such as DeepGCNs [17] which draws lesson from ResNet [11]. Note that different from the residual connections in ResNet, which aims at overcoming vanishing gradient problem in training deep neural network, the role of residual connections in GNNs is mainly the incorporation of local information. Although this architecture is the most similar to our proposed Difference Residual Connection in Figure 3(e), it is essentially an extension of architecture in Figure 3(c) and share the same advantages and disadvantages with initial residual connections.

4 EVALUATIONS

In this section, the performance of our proposed DRC is experimentally evaluated on real world datasets including both homophilic and heterophilic ones. Then, the universality and some attractive properties of DRC are validated.

Table 2: Mean Classification Accuracy (Bold indicates the best, underlined indicates the second best)

Models	Cora	Pubmed	Citeseer	Chameleon	Squirrel	Actor
GCN	85.77±0.25	88.13±0.28	73.68±0.31	28.18±0.23	23.96±0.26	26.86±0.23
GAT	86.37±0.30	87.62±0.26	74.32±0.27	42.93±0.28	30.03±0.25	28.45±0.23
GraphSAGE	87.77±1.04	88.42±0.50	71.09±1.30	49.24±1.68	36.28±1.73	35.28±1.37
MLP	74.82±2.22	63.76±0.78	70.94±0.39	49.67±0.78	37.04±0.46	34.10±0.25
Geom-GCN-I	85.19±1.13	**90.05±0.90**	**77.99±1.23**	60.31±1.77	33.32±1.59	29.09±0.86
Geom-GCN-P	84.93±0.51	88.09±1.37	75.14±1.50	60.90±1.13	38.14±1.23	31.63±0.98
Geom-GCN-S	85.27±1.48	84.75±1.39	74.71±1.17	59.96±2.03	36.24±1.05	30.30±1.20
GPRGNN	88.65±1.37	89.18±0.61	77.99±1.64	67.48±1.98	49.93±1.34	36.58±1.04
FAGCN	87.77±1.69	88.60±0.64	74.66±2.27	61.12±1.95	40.88±2.02	36.81±0.26
H2GCN-1	86.92±1.37	89.40±0.34	77.07±1.64	57.11±1.58	36.42±1.89	35.86±1.03
H2GCN-2	87.81±1.35	89.59±0.33	76.88±1.77	59.39±1.98	37.90±2.02	35.62±1.30
JKNet	**88.93±1.35**	87.68±0.30	74.37±1.53	62.31±2.76	44.24±2.11	36.47±0.51
APPNP	87.87±0.85	89.40±0.61	76.53±1.33	54.30±0.34	33.29±1.72	31.71±0.70
GCNII	88.49±2.78	89.57±1.56	77.08±1.21	60.61±2.00	37.85±2.76	36.18±0.61
DeepGCNs	83.58±1.21	84.87±0.77	70.64±1.89	39.22±0.89	28.89±1.55	30.16±0.73
DRC(ours)	87.14±0.91	88.96±0.48	76.38±1.08	**71.34±3.09**	**52.64±2.16**	**36.89±0.99**

4.1 Real world datasets

With various levels of homophily, 6 datasets: Cora, Citeseer, Pubmed, Actor, Chameleon and Squirrel, are adopted. Cora, Citeseer and Pubmed are three commonly used homophilic datasets, which are citation networks. Actor, Chameleon and Squirrel are datasets with low ratio of homophily. Actor is a graph representing actor co-occurrence in Wikipedia pages, based on the film-director-actor-writer network in [31]. Chameleon and Squirrel are two subgraphs of web pages in Wikipedia. Dataset statistics and the corresponding homophily ratios are summarized in Table 1. Homophily ratios provided in [26] are employed to show the proportion of node pairs with the same label in all the node pairs.

4.2 Baselines

To demonstrate the superiority of DRC, three classic GNN models are employed, including GCN [13], GAT [33] and GraphSAGE [10]. Besides, DRC is compared with other four recently proposed state-of-the-art methods designed for networks with heterophily, including Geom-GCN [26], H2GCN [42], GPRGNN [7] and FAGCN [3], and four models designed for tackling over-smoothing issue, including JKNet [36], APPNP [14], GCNII [6] and DeepGCNs [17]. All the results are obtained by running the code provided in the paper with default settings.

4.3 Node Classification Task

Settings. For all datasets listed in Table 1, we randomly split nodes of each class into 60%, 20% and 20% for training, validation and testing respectively, as in [26].

Experimental Results Analysis. Table 2 shows the mean classification accuracy and the standard error, where the bold and the underlined indicate the best and the second best performances, respectively. It can be observed that compared to recent methods designed for networks with heterophily including H2GCN, GPRGNN and FAGCN, the proposed DRC achieves new state-of-the-art results on Squirrel, Chameleon and Actor, and obtains competitive performance on other datasets, which demonstrate the superiority of DRC. Notably, the reason for performance improvement on networks with heterophily are two-fold. First, DRC meets the characteristic of heterophily. Specifically, heterophily means that most connected nodes possess different attributes and smoothing-based graph convolutional operation tends to be failed. Thus, by iteratively performing smoothing and sharpening as shown in Figure 2, DRC adaptively captures both homophily and heterophily characteristics in the networks. Second, the DRC possesses the ability to overcome over-smoothing issue.

Comparison with multi-scale models. It can be observed form Table 2 that DRC outperforms all the multi-scale models (e.g., JKNet) on three heterophilic datasets, which demonstrates the advantage of difference residual connections. The reasons is blindingly obvious. By adopting multi-scale architecture [1, 14, 15, 22, 36] shown in Figure 3(b), GNNs can alleviate over-smoothing issue but fail on networks with heterophily, since these architectures only perform the smoothing operation.

4.4 Comparison with Deep Models

To validate whether the proposed DRC can alleviate over-smoothing issue, four strong baselines: GCNII, FAGCN, GPRGNN and JKNet, which perform well as deep models, are compared with the proposed DRC with varying number of layers on 6 networks mentioned in Table 1. The results are shown in Figure 4.

Alleviating over-smoothing issue. As shown in Figures 4(a)-(d), different from the basic model GCN, our proposed DRC keeps the performance quite stable or rise continuously as the number of layers increases. This quality is mainly caused by the design of the difference residual connection, which is equivalent to replacing the successive smoothing operation in existing GNNs with the iterative balance between smoothing and sharpening operation (Figure 2).

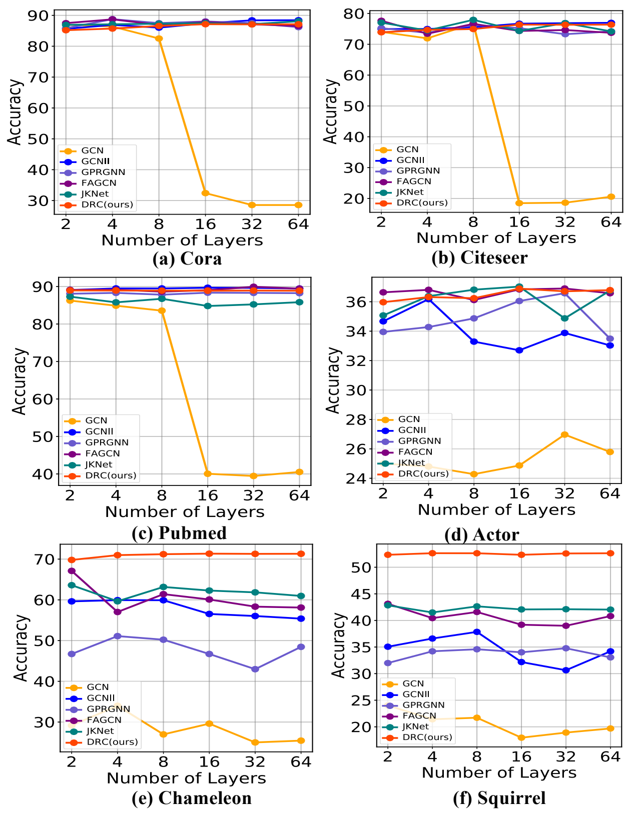


Figure 4: Classification accuracy results of deep GNN models with various depths.

Over-smoothing case on nodes with different degrees. The previous work has suggested that nodes with higher degrees are more likely to suffer from over-smoothing [27]. Figure 5 shows the mean classification accuracies of nodes with different degrees on Chameleon and Squirrel datasets, which demonstrates that DRC consistently enhances the performances of all nodes with both high and low degrees. Therefore, DRC can sort out the over-smoothing issue of nodes with higher degrees.

Comparison with other deep models on heterophilic networks. As shown in Figures 4(e)-(f), the proposed DRC significantly outperforms other SOTA deep models which are designed for networks with heterophily, on representative heterophilic networks Chameleon and Squirrel. Similar to the performance on homophilic networks, the performance on networks with heterophily are stable as the number of layer increases. This also shows the superior ability of DRC on balancing smoothing and sharpening effects, and demonstrates that the proposed DRC are more effective on capturing high-order information.

Efficiency analysis on real world benchmark datasets. The average running time per epoch and total running time of GCN, GCNII and DRC on three largest datasets are given in Table 3. All experiments are conducted on an NVIDIA Geforce 3090 GPU. It can be observed from Table 3 that indeed DRC has a running time similar to that of GCN and less than that of GCNII, which demonstrates that DRC is faster than other models even working as a deep model.

Table 3: Average running time per epoch/ total running time.

Dataset	Cora (2 layers)	Citeseer (16 layers)	Chameleon (16 layers)
GCN	14.23ms / 9.52s	14.30ms / 9.43s	15.18ms / 7.20s
GCNII	23.08ms / 18.01s	67.33ms / 21.81s	35.99ms / 12.72s
DRC	15.69ms / 13.10s	15.87ms / 11.69s	17.21ms / 8.35s

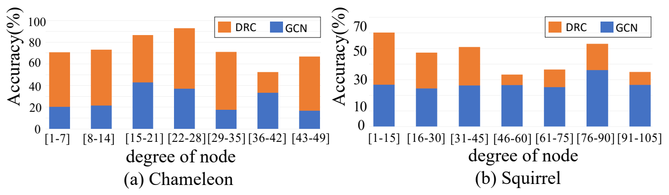


Figure 5: Mean accuracies of nodes with different degrees.

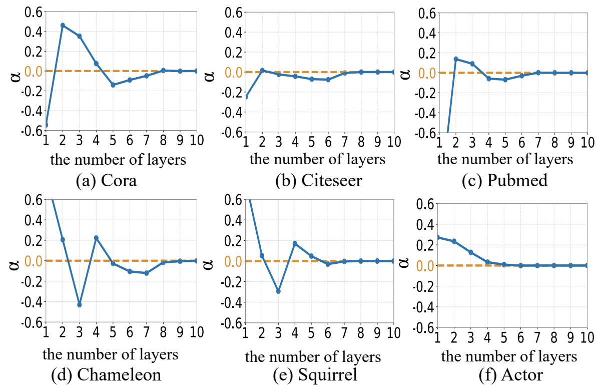


Figure 6: The trends of the learnable parameters α's with the increment of layer on datasets with both homophily and heterophily.

4.5 Automatic Determination of Mode Depth

Different from the most existing GNNs which pre-define the number of layers, the proposed DRC assigns learnable output weights α to all the layers. Figure 6 shows the learned α in different layers on networks with both homophily and heterophily. It can be observed that these output weights possess an attractive property that they decrease to zero as the number of layers increases. Thus, DRC can automatically determine the number of layers, i.e., model depth. This property guarantees that the DRC tends to converge to stable even we initialize a large number of layers. This property can also reveal the reason why DRC can overcome the over-smoothing issue from another perspective. Since the over-smoothing issue is often caused by the large number of graph convolutional layers which act as smoothing operation, DRC overcomes over-smoothing issue by employing only a small part of them even a large number of them are available. Therefore, the automatic determination of mode depth makes DRC overcome the over-smoothing issue.

Moreover, for node classification task, it is intuitive that the local information is more important than the global one, and the node too far away from the target one often plays a limited role. By assigning a learnable parameters α to the output of each layer instead of fixing the number of layers, the varying α verifies the

Table 4: DRC with different GNN backbones.

Dataset	Cora	Pubmed	Citeseer	Cham.	Squirrel	Actor
GraphSAGE	87.11	86.36	74.32	39.96	34.58	36.18
+DRC	**87.19**	**88.83**	**76.17**	**44.35**	**39.25**	**36.99**
GAT	86.00	84.51	74.92	37.72	25.84	25.46
+DRC	**86.92**	**88.72**	**76.35**	**39.21**	**32.19**	**35.13**
GCN	**86.37**	84.89	71.92	33.99	21.42	24.80
+DRC	85.72	**88.94**	**74.67**	**70.98**	**52.64**	**36.32**

Table 5: Results on synthetic datasets.

Homophily	**-1**	**-0.5**	**-0.25**	**0**	**0.25**	**0.5**	**1**
GPRGNN	96.25	93.75	86.50	85.75	86.75	92.25.	97.00
GCNII	50.50	57.50	73.50	77.00	91.00	91.00	96.75
DRC	98.50	96.50	96.00	96.50	96.00	96.50	97.00

above intuition and makes the model adaptive and efficient. We can observe from Figure 6 that there are widely differences between the trends of α with the increment of layers on datasets with homophily and heterophily. We will try to find a theoretical basis to support this attractive characteristic in future work.

4.6 DRC with Different GNN Backbones

In previous experiments which demonstrate the superiority of DRC, GCN [13] is adopted as the backbone, i.e. the GCL in Figures 2 and 3. Essentially, the proposed DRC is a general framework, where any existing GNNs can be employed as the backbones. To verify this, the backbone in the proposed DRC is replaced with GAT [33] and GraphSAGE [10], respectively. Table 4 shows the performance improvements of DRC with different GNNs as backbones. It is obvisous that DRC can significantly improve the performances of GNN backbones, especially on the networks with heterophily. Therefore, the proposed DRC is universal and effective framework to enhance the existing GNNs.

4.7 Evaluation on Synthetic Datasets

In order to fully test the ability of DRC with varying levels of homophily and heterophily, we propose to use cSBMs to generate synthetic graphs, following the generation emploited by GPRGNN. Note that parameter Φ controls homophily ratio of datasets and the corresponding homophily ratios are denoted by *Homophily* in Table 5. More specifically, $\Phi = 1$ corresponds to strongly homophilic graphs while $\Phi = -1$ corresponds to strongly heterophilic graphs. A detailed difference between the generation of synthetic datasets proposed by GPRGNN and this paper, is that we include 1,000 nodes and each node owns 1,500 features and approximately 10,000 edges in synthetic datasets. The performances of DRC in synthetic datasets with varying homophily ratios are shown in Table 5, which also includes the results of two state-of-the-art methods, i.e., GCNII [6] and GPRGNN [7] on the same setting. We can observe that our proposed DRC outperforms both GCNII and GPR, two outstanding models that can alleviate over-smoothing issue and be adaptively used on both homophilic and heterophilic datasets. It demonstrates the superiority of DRC on preventing over-smoothing and its applicability for both homophilic and heterophilic datasets.

Table 6: Ablation Study on residual connection and α's.

# Layers	Components	Chameleon	Squirrel	Actor
4 Layers	w/o weight α	66.47	51.16	34.72
	w/o residual	64.55	48.91	35.97
	DRC	**70.98**	**52.64**	**36.32**
16 Layers	w/o weight α	21.18	24.74	24.91
	w/o residual	66.37	49.45	35.83
	DRC	**71.34**	**52.34**	**36.89**

4.8 Ablation Study

This section performs the ablation study to provide an intuitive understanding to the two components, i.e. residual connection and output weight α. To this end, the performances of DRC with one component and different numbers of layers are given in Table 6. It can be observed that the performances of DRC with one component are much lower than those of full DRC. Therefore, both two components play important roles in the performance improvement. Besides, it can be found that the residual connection plays more important role than the output weight in shallow GNNs (4 layers), while the output weight are more critical than residual connection in deep GNNs (16 layers). Thus, this verifies that these two components are adaptively complementary to performance improvements.

5 CONCLUSIONS

To alleviate the issue of GNNs on over-smoothing and ineffectiveness on network with heterophily, a novel and well-performed Difference Residual Connection (DRC) is proposed by simply feeding the difference of the output and input of previous layer as the input of the next layer. Theoretically, DRC essentially inserts layers with the opposite effect (e.g., sharpening) into the network to prevent the excessive effect (e.g., over-smoothing issue) induced by too many layers with the similar role (e.g., smoothing) in GNNs. From the perspective of optimization, DRC minimizes an objective function with both smoothing and sharpening terms, and the analytic solution is determined by both graph topology and node attributes, which theoretically proves that DRC can prevent over-smoothing issue. Extensive experiments demonstrate the superiority of DRC on real networks with heterophily and preventing over-smoothing issue and the universality to enhance the existing GNNs.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1406704, the National Natural Science Foundation of China under Grant 61972442, Grant 62102413, Grant U2001202, Grant 62025604, Grant 61876128, in part by the Key Research and Development Project of Hebei Province of China under Grant 20350802D and 20310802D; in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Natural Science Foundation of Tianjin of China under Grant 20JCYBJC00650, and in part by the China Postdoctoral Science Foundation under Grant 2021M703472.

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] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. 2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*.

[3] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. (2021), 3950–3957.

[4] Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Dual Quaternion Knowledge Graph Embeddings. In *AAAI*. 6894–6902.

[5] Jingjing Chen, Liangming Pan, Zhipeng Wei, Xiang Wang, Chong-Wah Ngo, and Tat-Seng Chua. 2020. Zero-Shot Ingredient Recognition by Multi-Relational Graph Convolutional Network. In *AAAI*. 10542–10550.

[6] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*. 1725–1735.

[7] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

[8] Wanqiu Cui, Junping Du, Dawei Wang, Feifei Kou, and Zhe Xue. 2021. MVGAN: Multi-View Graph Attention Network for Social Event Detection. *ACM Trans. Intell. Syst. Technol.* 12, 3 (2021), 27:1–27:24.

[9] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *CVPR*. 2261–2269.

[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] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. In *NeurIPS*. 13333–13345.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*. 1106–1114.

[17] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *ICCV*. 9266–9275.

[18] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. 3538–3545.

[19] Zhaopeng Li, Qianqian Xu, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang. 2020. Quaternion-Based Knowledge Graph Network for Recommendation. In *MM*. 880–888.

[20] Changshu Liu, Liangjian Wen, Zhao Kang, Guangchun Luo, and Ling Tian. 2021. Self-supervised Consensus Representation Learning for Attributed Graph. In *MM*. 2654–2662.

[21] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards Deeper Graph Neural Networks. In *SIGKDD*. 338–348.

[22] Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. 2019. Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks. In *NeurIPS*. 10943–10953.

[23] 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:cs.LG/2010.01777

[24] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*.

[25] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning. In *ICCV*. IEEE, 6518–6527.

[26] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[27] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

[28] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE Signal Process. Mag.* 30, 3 (2013), 83–98.

[29] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*. 1–9.

[31] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks. In *SIGKDD*. ACM, 807–816.

[32] Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *SIGGRAPH*. 351–358.

[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[34] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. 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. 2021. A Comprehensive Survey on Graph Neural Networks. *TNNLS* 32, 1 (2021), 4–24.

[36] 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.

[37] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2021. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. arXiv:cs.LG/2102.06462

[38] Liang Yang, Chuan Wang, Junhua Gu, Xiaochun Cao, and Bingxin Niu. 2021. Why Do Attributes Propagate in Graph Convolutional Neural Networks?. In *AAAI*. 4590–4598.

[39] Kun Zhan and Chaoxi Niu. 2021. Mutual teaching for graph convolutional networks. *Future Gener. Comput. Syst.* 115 (2021), 837–843.

[40] Lingxiao Zhao and Leman Akoglu. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *ICLR*.

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

[42] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*.

[43] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *WWW*. 1215–1226.