Handling Over-Smoothing and Over-Squashing in Graph Convolution With Maximization Operation

Dazhong Shen, *Member, IEEE*, Chuan Qin, *Member, IEEE*, Qi Zhang, *Member, IEEE*, Hengshu Zhu, *Senior Member, IEEE*, and Hui Xiong, *Fellow, IEEE*

***Abstract*— Recent years have witnessed the great success of the applications of graph convolutional networks (GCNs) in various scenarios. However, due to the challenging *over-smoothing* and *over-squashing* problems, the ability of GCNs to model information from long-distance nodes has been largely limited. One solution is to aggregate features from different hops of neighborhoods with a linear combination of them followed by a shallow feature transformation. However, we demonstrate that those methods can only achieve a tradeoff between tackling those two problems. To this end, in this article, we design a simple yet effective graph convolution (GC), named maximization-based GC (MGC). Instead of using the linear combination, MGC applies an elementwise maximizing operation for exploiting all possible powers of the normalized adjacent matrix to construct a GC operation. As evidenced by theoretical and empirical analysis, MGC can effectively handle the above two problems. Besides, an efficient approximated model with a linear complexity is developed to extend MGC for large-scale graph learning. To demonstrate the effectiveness, scalability, and efficiency of our models, extensive experiments have been conducted on various benchmark datasets. In particular, our models achieve competitive performance with lower complexity, even on large graphs with more than 100M nodes. Our code is available at https://github.com/SmilesDZgk/MGC.**

***Index Terms*— Graph convolutional network (GCN), node classification, over-smoothing, over-squashing.**

I. Introduction

As AN advanced neural approach for modeling complex graph-structured data, graph convolutional network (GCN) [29], [66] and its variants have obtained great success in various application scenarios, including social media [2], [21], traffic prediction [5], [33], biology [32], [70], recommender systems [16], [47], [48], [65], language modeling [12], [54], [79], computer vision [51], [63], knowledge discovery and diagnosis [6], [45], [71], [72], and even talent management [26], [46], [56], [76], [77], [78]. To learn the graph representations, GCNs apply the fixed linear feature propagation between each node and its neighbors with graph convolution (GC) operation (i.e., the normalized adjacent matrix) followed by a trainable nonlinear feature transformation.

While considerable efforts have been made to improve the performance of GCNs, there are still some long-standing challenges that cannot be neglected. The most well-known problem is named *over-smoothing* [36], which means the node representations will become indistinguishable and even converge to the same value as the number of layers increases. As a result, most of the recent GCN models tend to use fairly shallow settings (e.g., vanilla GCN [29] achieves their best performance with two layers), which limits their ability to reach the high-order neighbors. One solution for that is to widen the receptive field of feature propagation while limiting the depth of transformation networks to reduce the negative effect caused by deep neural networks [83]. Along this line, a series of works focus on the design of the GC operation and specify it as the linear combination of powers of the normalized adjacent matrix with different weighting coefficients, such as SGC [64], S^2GC [83], and approximate PPNP (APPNP) [30]. Furthermore, graph diffusion convolution (GDC) [31] provides a unified framework to design those GC operations by generalizing the graph diffusion process. All of them can deepen the layers of GCNs without performance degradation to some extent.

However, we will demonstrate that all GDC-based models have to abandon the long-distance information to some extent for relieving over-smoothing. In other words, these models can only achieve a tradeoff between tackling over-smoothing and another critical problem in GCNs, i.e., *over-squashing* [1]. It suggests that GCNs fail to handle the long-distance dependencies between distant nodes even though they are included in the receptive field. The rationale behind is that GCNs absorb incoming edges equally in each layer [1], where the contribution of each distant neighbor would be mostly submerged among the exponentially growing receptive field as the number of layers increases. Most of the previous works to address over-squashing are based on the adjustment of graph structure by adding extra nodes or virtual edges [1],

Manuscript received 1 October 2022; revised 22 January 2024 and 23 June 2024; accepted 2 August 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFF0725001, in part by the National Natural Science Foundation of China under Grant 92370204, in part by Guangzhou-HKUST(GZ) Joint Funding Program under Grant 2023A03J0008, in part by the China Postdoctoral Science Foundation under Grant 2024M750154 and Grant 2023M741849, and in part by the Education Bureau of Guangzhou Municipality. *(Corresponding authors: Hengshu Zhu; Hui Xiong.)*

Dazhong Shen and Qi Zhang are with the Shanghai Artificial Intelligence Laboratory, Shanghai 201210, China (e-mail: dazh.shen@gmail.com; zhangqi.fqz@gmail.com).

Chuan Qin is with the PBC School of Finance, Tsinghua University, Beijing 100083, China (e-mail: chuanqin0426@gmail.com).

Hengshu Zhu is with the Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China, and also with the Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: zhuhengshu@gmail.com).

Hui Xiong is with the Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: xionghui@ust.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2024.3442270, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2024.3442270

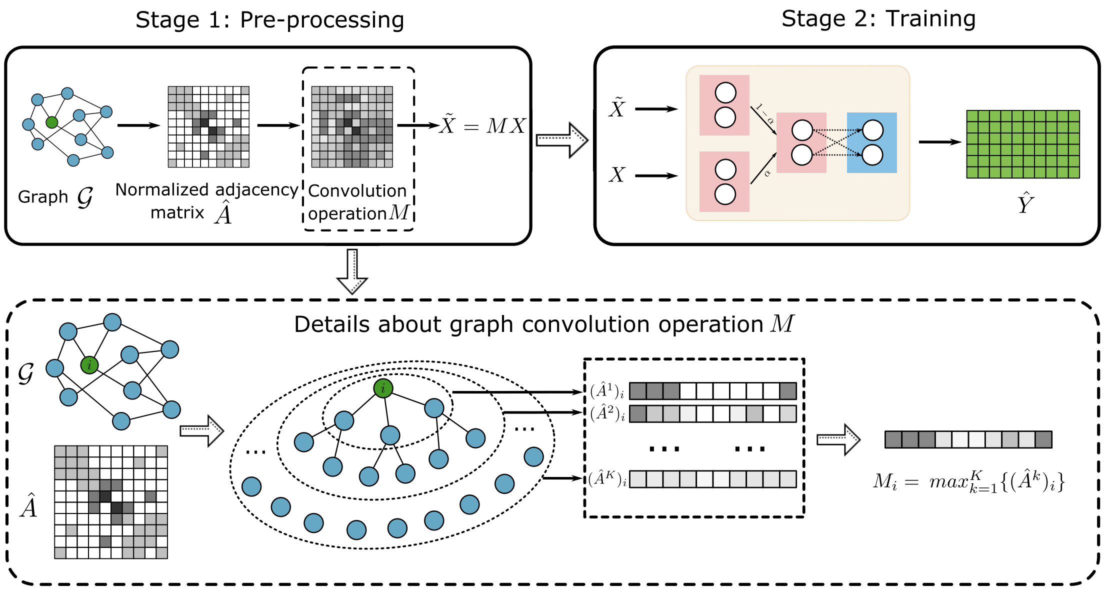


Fig. 1. Overview of the proposed MGC pipeline, where the feature propagation and transformation are decoupled into the preprocessing stage and training stage, respectively.

[14], [52], [58]. However, the correctness and rationality of all involved new nodes or edges cannot be guaranteed, which may include unnecessary computation and noise information.

To this end, in this article, instead of using the linear combination like that in GDC-based models, we propose a simple yet effective GC, i.e., maximization-based GC (MGC). Specifically, we design the whole pipeline as in Fig. 1. Instead of conducting feature propagation and transformation alternately, we propose to decouple them into two separate parts. Our MGC operation is involved in the preprocessing stage to propagate and fuse features among different nodes. Subsequently, during the training stage, a simple multilayer perceptron (MLP) layer is trained using a combination of both the original and processed node features as input, with the node label as the supervision. In particular, as shown in the below box, MGC applies an elementwise maximizing operation for exploiting all possible powers of the normalized adjacent matrix to construct the GC operation. We demonstrate that our MGC operation can alleviate over-squashing better than any GDC operation theoretically and can also avoid over-smoothing with empirical evidence. In addition, to reduce the high complexity for large scalable graph learning, we develop the approximated approach, i.e., approximated MGC (AMGC), with a complexity that is linear in the number of edges. To demonstrate the effectiveness, scalability, and efficiency, extensive empirical analysis has been conducted on 14 node classification benchmark datasets, including large-scale datasets, such as Ogbn-Papers100M and MAG240M [20]. The results demonstrate that our models achieve competitive performance in capturing both the neighbor and long-distance dependency with lower complexity.

The major contributions of this article are listed as follows.

1) We demonstrate that all GDC-based models, which aggregate features from different hops of neighborhoods with a linear combination of them, can only achieve a tradeoff between tackling over-smoothing and over-squashing problems.
2) We propose a simple yet effective GC based on maximization operation, i.e., MGC, to handle both over-smoothing and over-squashing problems. Also, an efficient approximated approach to MGC with a linear complexity has been designed.
3) Extensive experiments on 14 node classification benchmark datasets have been conducted to demonstrate the effectiveness, scalability, and efficiency of our models.

The remainder of this article is structured as follows. In Section II, we briefly introduce the necessary background knowledge about the various GCNs. In Section III, we will point out the tradeoff of all GDC-based models on tackling over-smoothing and over-squashing problems. Technical details of our proposed model, i.e., MGC, will be specified in Section IV with effectiveness and efficiency analysis. In particular, an approximated approach, i.e., AMGC, will also be designed. Then, we evaluate the effectiveness, scalability, and efficiency of our models on 13 public benchmark datasets in Section V, with some further discussions on experimental results. In Section VI, we conclude this article.

II. Preliminaries

Here, we introduce the necessary notations and review the previous related GCN variants.

Formally, given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ($\mathcal{G}$ is assumed to be connected for convenience), where $\mathcal{V}$ represents the vertex set consisting of n nodes, i.e., $\{v_1, \ldots, v_n\}$, and $\mathcal{E}$ represents the edge set with size of m. We denote the $A \in \mathbb{R}^{n\times n}$ as the adjacent matrix and $D = \mathrm{diag}(d_1, \ldots, d_n)$ as the diagonal degree matrix, respectively. Here, d_i is equal to the row sum of the adjacency matrix, i.e., $d_i = \sum_j A_{ij}$. Then, the adjacent matrix with self-loops can be denoted by $\tilde{A} = A + I_n$ with the corresponding diagonal degree matrix $\tilde{D} = D + I_n$, where I_n is an identity matrix. $\hat{A}$ is the normalized adjacency matrix with added self-loops, i.e., $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$. Meanwhile, each node is represented by a b-dimensional vector $x_i \in \mathbb{R}^b$, while the integrated feature matrix is denoted by $X = [x_1, \ldots, x_n]^{\mathrm{T}} \in \mathbb{R}^{n\times b}$. Besides, each node is assigned by one of c classes. The label vector y_i of each node is one-hot vectors, i.e., $y_i \in \{0, 1\}^c$. The node classification task aims to predict the unlabeled node set based on the labeled node set.

A. Vanila GCN [29]

Different from MLPs, which only model the feature transformation independently on each instance, a K-layer GCN also propagates the representation of each node among its neighbors at each layer. This procedure can be defined as follows:

$$H^{(k+1)} = \sigma\left(\hat{A}H^{(k)}W^{(k)}\right) \tag{1}$$

where $H^{(k+1)}$ and $H^{(k)}$ are the smoothed feature matrix at layer $k+1$ and k, respectively. $H^{(0)}$ is set as the input feature matrix, X. $W^{(k)}$ is a layer-specific trainable weight matrix, and $\sigma(\cdot)$ is a nonlinear activation function, such as ReLU.

B. SGC [64]

A GCN layer averages the hidden representations among one-hop neighbors, which implies that K-layer GCN can capture feature information from all nodes in the K-hop neighbors. By hypothesizing that the majority of the benefit of GCN models arises from the local averaging at each GCN layer not the nonlinearity between layers, SGC simplifies the K-layer GCN by applying the Kth power of the normalization adjacent matrix $\hat{A}$ to model the feature propagation and feeds output feature to a single linear model before the prediction function

$$\hat{Y} = \mathrm{Softmax}\left(\hat{A}^K XW\right). \tag{2}$$

Following the main idea to simplifying GCNs into linear models, **S^{2}GC** [83] further replaces $\hat{A}^k$ with $1/K\sum_{k=1}^{K}\hat{A}^k$ to alleviate the over-smoothing problem while maintaining the high scalability and efficiency.

C. APPNP [30]

Inspired by the personalized PageRank [43] algorithm, Klicpera et al. [30] proposed to derive a PPR-based convolution, PPNP, by decoupling the feature transformation and propagation

$$H = \alpha\left(I_n - (1-\alpha)\hat{A}\right)^{-1}H^{(0)} \tag{3}$$

where $\alpha \in (0, 1)$ represents the teleport probability in PPR to balance the needs of preserving locality and leveraging the information from a distant neighbor and $H^{(0)} = f(X)$ is the output of a two-layer fully connected neural network on the feature matrix X. To avoid the complex calculation for the inverse of matrix $\hat{A}$, APPNP is proposed with a truncated power iteration on the K-hop neighbors

$$H^{(k+1)} = (1-\alpha)\hat{A}H^{(k)} + \alpha H^{(0)}. \tag{4}$$

D. Graph Diffusion Convolution [31]

A generalized graph diffusion is a linear combination of all possible powers of normalized adjacent matrix $\{\hat{A}^k\}_{k=1}^{\infty}$, called GDC operation

$$S = \sum_{k=1}^{\infty}\theta_k\hat{A}^k \tag{5}$$

where the weighting coefficients θ_k are constrained by that $\sum_{k=1}^{\infty}\theta_k = 1$ to ensure the above equation converges. The S itself or its normalization $\hat{S} = D_s^{-1/2}SD_s^{-1/2}$ with the diagonal degree matrix D_s is used as the convolution operation to model the feature propagation before the feature transformation, i.e., $\hat{S}X$. Note that θ_k can be a trainable parameter or fixed with prior knowledge.

Indeed, by assuming the nonlinearity in GCNs is not critical [64], many works can be regarded as the expansion of GDC with different choices of θ_k, where we call them as GDC-based models.

1) Vanila GCN is with $\theta_k = 1$ if $k = 1$, else 0.
2) SGC is with $\theta_k = 1$ if $k = K$, else 0.
3) S^{2}GC is with $\theta_k = 1/K$ if $0 < k \leq K$, else 0.
4) APPNP is with $\theta_k = \alpha(1-\alpha)^k$, if $0 \leq k \leq K$, else 0.

III. Tradeoff Between Tackling Over-Smoothing and Over-Squashing

Although GCNs have been applied successively to deal with broad classes of systems of relations and interactions, there are still some long-standing challenges that cannot be neglected [1], [22], [67]. Here, we first discuss the most popular two problems, i.e., *over-smoothing* and *over-squashing*, which essentially limit GCNs for modeling information from long-distance nodes. Then, we point out that the GDC-based models can only achieve a tradeoff between tackling those two problems.

A. Over-Smoothing

Over-smoothing is the most well-known problem in GCNs, which suggests that the node representations are inclined to be indistinguishable as the number of feature propagation steps increases. Li et al. [36] explain the hidden rationale is that most eigenvalues of the normalized adjacent matrix $\hat{A}^k$ will converge to 0 as k increases to infinity. Taking SGC as an example, the output representation $X^{(\infty)}$ after an infinite number of feature propagation will become

$$X^{(\infty)} = \hat{A}^{\infty}X = U\hat{\Lambda}^{\infty}U^{\mathrm{T}}X \tag{6}$$

where $\hat{\Lambda} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_n\}$ is a diagonal matrix of the eigenvalues of $\hat{A}$ and $U \in R^{n\times n}$ is a unitary matrix that consists of the eigenvectors. Noting that all eigenvalues $\lambda_1, \ldots, \lambda_n$ fall to $(-1, 1]$ and only the max one $\lambda_1 = 1$. Therefore, we have $\hat{\Lambda}^{\infty} = \mathrm{diag}\{1, \ldots, 0\}$ and $\hat{A}^{\infty} = U_1^T U_1$ with $U_{1,j} = ((d_j+1)^{1/2}/(2m+n)^{1/2})$. In other words, each row vector of $\hat{A}^{\infty}$ has the same direction, and each element $\hat{A}_{ij}^{\infty}$ only relates to the node degrees of target nodes and source nodes without the full graph structure information, which results indistinguishable feature vectors in $X^{(\infty)}$.

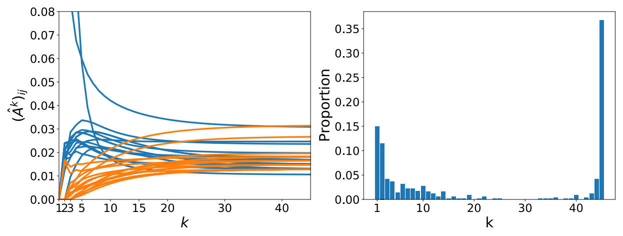


Fig. 2. Left part shows the illustrative curve of $\hat{A}^k_{ij}$ between two nodes v_i and v_j as k increases in the Zachary Karate dataset. In particular, the source node v_i is randomly selected and fixed, while the target node v_j varies across all nodes. The lines that $\hat{A}^k_{ij}$ has not achieved the maximum before $k = 45$ are highlighted by orange color. The right part shows the proportion of elements $\hat{A}^k_{ij}$ in the matrix $\hat{A}^k$ that reach the maximum on different k values, where $1 \leq k < 45$ and zero elements have been omitted.

B. *Over-Squashing*

Over-squashing is another critical problem in GCNs and attracting more attention recently [1], [58], which suggests that the node representations fail to handle long-distance dependency between distant nodes. GCNs need to have at least K layers to receive information from the nodes beyond a distance of K. However, as the number of layers increases, GCNs must compress information from the exponentially growing receptive field into fixed-length node vectors [1]. Because GCNs absorb incoming edges equally in each layer, the contribution of each distant node would be mostly ignored compared with that of neighbors. Most of the previous works to tackle over-squashing are based on the adjustment of graph structure by adding "supersource" nodes [52] or virtual edges [14], [58] to connect nodes with distant distance and even connect all nodes to capture global information [1]. Here, we aim to avoid over-squashing without modifying the graph structure, to decrease unnecessary computation and noise information.

C. *Tradeoff Dilemma*

Following [58] and [68], we assess the over-squashing effect between node v_i and its distant neighbor v_j in GDC-based models by measuring how much a change in the input feature of v_j affects the representation of v_i with the Jacobian, i.e.,

$$\frac{\partial (SX)_i}{\partial X_j} = S_{ij} = \sum_{k=1}^{\infty} \theta_k \left(\hat{A}^k\right)_{ij}. \tag{7}$$

Intuitively, to strengthen the connection between nodes v_i and v_j, we need to assign more weight to $(\hat{A}^k)_{ij}$ with larger value. However, if the distance (shortest path distance) between them is d, we have $(\hat{A}^k)_{ij} = 0\ \forall k < d$. Therefore, we must enlarge several values of θ_k, where $k \geq d$ at least. In addition, the strongest connection between two distant nodes with the largest $(\hat{A}^k)_{ij}$ rarely occurs around $k = d$. Take the small graph, Zachary Karate dataset [73] with 34 nodes, as an example. We show the curve of $\hat{A}^k_{ij}$ between one node v_i and each of other nodes $j \in \{1, 2, \ldots, 34\}$ as k increases in Fig. 2(a), where each curve corresponds to one node pair (v_i, v_j). We can find that the value $(\hat{A}^k)_{ij}$ of several node pairs is larger with larger k (colored by orange) and even monotonically increases as k increases, especially for long-distance node pairs (v_i, v_j) with $(\hat{A}^k)_{ij} = 0$ for $k = 1,\ 2$, or 3. In addition, we further show the proportion of elements in $\hat{A}$ that reach the largest on different k values, where $1 \leq k < 45$. We find that a considerable proportion of $\hat{A}^k_{ij}$ reaches the largest on the last power. As a result, to alleviate the over-squashing and enhance the relation between long-distance nodes, we must assign more weight θ_k with large k.

Following [80], we assess the degree of over-smoothing by measuring the distance between GDC operation S in (5) and over-smoothing stationarity $\hat{A}^{\infty}$ with Frobenius norm. Then, we can derive

$$\begin{aligned}
\left\| S - \hat{A}^{\infty} \right\|_F^2 &= \left\| U\left(\sum_{k=0}^{\infty} \theta_k \hat{\Lambda}^k - \hat{\Lambda}^{\infty}\right) U^{\mathrm{T}} \right\|_F^2 \\
&\leq \|U\|_F^2 \left\| \sum_{k=0}^{\infty} \left(\theta_k \hat{\Lambda}^k - \hat{\Lambda}^{\infty}\right) \right\|_F^2 \|U^{\mathrm{T}}\|_F^2 \\
&= n^2 \sum_{i=2}^{n} \left(\sum_{k=0}^{\infty} \theta_k \lambda_i^k \right)^2 \leq n^3 \sum_{i=2}^{n} \sum_{k=0}^{\infty} \theta_k^2 \lambda_i^{2k}
\end{aligned} \tag{8}$$

where the first line is based on the decomposition $\hat{A} = U \hat{\Lambda} U^{\mathrm{T}}$ and the second line is based on the property of the Frobenius norm $||\cdot||_F$. Noting that $|\lambda_i| < 1, \forall i > 1$, and λ_i^k approximates exponentially to 0, we need to assign more weight θ_k with smaller k to avoid the left term to be too small, which causes the over-smoothing problem.

In summary, based on the above analysis, we observe the opposite requirements to assign the weighting coefficients θ_k on handling the over-squashing and over-smoothing problems in GDC-based models. Therefore, we argue that any GDC-based models with the GC operation defined in (5) can only achieve a tradeoff of tackling with over-smoothing and over-squashing problems at most.

IV. Maximization-Based GC

Here, we first introduce the proposed GC operation, i.e., MGC, with a detailed discussion about its effect on handling over-squashing and over-smoothing. Then, the design of the whole MGC pipeline is described. Finally, the efficiency analysis is conducted, where an efficient approximated approach is developed to reduce complexity.

A. *MGC Operation*

Similar to GDC operations, our goal is to find a suitable feature propagation matrix $M \in R^{n \times n}$ as the GC operation with consideration on powers of the normalized adjacent matrix $\hat{A}$. Inspired by that in (7), we assume the Jacobian $\partial (MX)_i / \partial X_j = M_{ij}$ as the measure of the influence on the output feature of node v_i from the input feature of node v_j. We have an intuitive idea that any other nodes, including both the short-distance and long-distance neighbors, should all affect the nodes v_i at most. Therefore, instead of the linear combination in the GDC-based models, we propose to use the maximization operation among the set $\{\hat{A}^k\}_{k=1}^{K}$ to construct the GC operation M, called MGC operation, that is,

$$M = \max\nolimits_{k=1}^{K} \left\{ \hat{A}^k \right\} \tag{9}$$

where $\max\{\cdot\}$ is an elementwise maximization operation on the matrix set. We can consider all possible feature propagation by increasing K to ∞ to capture comprehensive graph information. Actually, MGC operation can be regarded as one special case of the generalized GDC operation, i.e.,

$$M_{ij} = \sum_{k=1}^{K} \theta_{k,ij} \left(\hat{A}^k\right)_{ij}$$
$$\theta_{k,ij} = \begin{cases} 1, & k = \operatorname{argmax}_{s=1}^{K}\left\{\left(\hat{A}^s\right)_{ij}\right\} \\ 0, & \text{others.} \end{cases} \tag{10}$$

Note that we have also constrained the weighting coefficients by $\theta_{k,ij}$ that $\sum_{k=0}^{K}\theta_{k,ij} = 1, \forall i, j$ to ensure convergence and control the scale.

B. Effectiveness Analysis

Here, we turn to analyze the effectiveness of MGC operation on alleviating over-squashing and over-smoothing with theoretical or empirical evidence.

1) Alleviating Over-Squashing: By comparing GDC operation in (5) and MGC operation in (9), we can derive the following corollary.

Corollary 1: Given any GDC operation S defined in (5) with consideration of K'-hop neighbors (i.e., $\theta_k = 0$, $\forall k > K'$), we can specify an MGC operation with $\alpha = \theta_0$ and any $K \geq K'$, in which

$$\frac{\partial (SX)_i}{\partial X_j} = S_{ij} \leq M_{ij} = \frac{\partial (MX)_i}{\partial X_j} \quad \forall i, j. \tag{11}$$

This corollary suggests that the effect of the interaction between any pair nodes, including the distant nodes, in the MGC operation can be larger than that in any GDC operations, which indicates that MGC operation prefers to alleviate the over-squashing problem better.

2) Avoiding Over-Smoothing: As for measuring the degree of over-smoothing problem in MGC operation, it may be not suitable to use the distance between M and $\hat{A}^\infty$, such as (8), because the maximization operation has changed both the eigenvectors and eigenvalues of $\hat{A}$, which means there is no direct correspondence between the eigenvalues of M and $\hat{A}^k$. Actually, the rank of the GC operation is highly related to the degree of the over-smoothing problem. In extreme cases, when the rank of the GC operation is equal to 1, the over-smoothing will occur where the output representations of all nodes are linearly correlated with each other. In the opposite sense, a larger rank number indicates a lower degree of over-smoothing, which the number of nondegenerated eigenvalues can measure. Therefore, we turn to depict the spectrum (the distribution of eigenvalues) and count the number of nondegenerated eigenvalues (without vanishing to 0) of the MGC operation to analyze the ability to avoid over-smoothing.

To be specific, taking Cora [53] and Cornell [44] dataset as examples, Fig. 3(a) and (b) shows the spectrum of MGC operation M and powers of normalized adjacent matrix $\hat{A}^K$ with different K values. We can observe that the eigenvalues degeneration would not occur in MGC operation even with quite large $K > 1000$. More interestingly, we note that MGC operations with different K values all have a very similar spectrum with $\hat{A}$ in the medium region. It indicates that MGC operation and $\hat{A}$ preserve the similar structure information. In contrast, $\hat{A}^k$ will degenerate to the over-smoothing stationarity rapidly as K increases. In other words, the GDC-based models with the linear combination of $\{\hat{A}^k\}_{k=0}^{\infty}$ as GC operation can only benefit from one choice, alleviating over-smoothing or capture long-distance information, not both, i.e., the tradeoff dilemma discussed in Section III (more illustrations on other datasets can be found in Supplementary material, where similar phenomena have been witnessed).

Here, we also provide a synthetic experiments. Specifically, we first generate 300 synthetic random graphs with different sizes and sparsity by stochastic block model [19], which is a popular generative model for random graphs. Then, as for M and $\hat{A}^K$, we count the number of nondegenerate eigenvalues λ with $\lambda > \epsilon$. Intuitively, the ratio of $R(S) = C(S)/C(\hat{A})$ is regarded as an index of the ability to avoid over-smoothing in GC operation S. Based on the numerical results in Fig. 3(c), we can find that MGC operation can always keep large $R(M)$ (slightly less than 1.0) even with large K, while $R(\hat{A}^K)$ vanishes to 0 rapidly. It leads the consistent conclusion with the two examples in Fig. 3(a) and (b).

C. MGC Pipeline

Following SGC, S^{2}GC, and APPNP, which all decouple the feature propagation and transformation, we propose a simple pipeline as follows:

$$\hat{Y} = F_3((1-\alpha)\sigma(F_1(MX)) + \alpha\sigma(F_2(X))) \tag{12}$$

where $F_*(\cdot)$ is a trainable linear layer, the activation function $\sigma(\cdot)$ is set as ReLU, and the row $\hat{Y}_i$ in $\hat{Y}$ is the predicted label distributions (before the softmax function) for the node v_i. Besides, following [83], the hyperparameter $\alpha \in [0, 1)$ aims to balance the self-information of node versus consecutive neighbors. Actually, as Fig. 1 shows, our MGC pipeline can be split into two main parts: 1) the fixed feature propagation component $\tilde{X} = MX$, which requires no trainable weight and can be done in the feature preprocessing stage and 2) the feature transformation component as the only part in training stage, which aims to train a multiclass classifier on the preprocessed features $\tilde{X}$ and original features X.

D. Efficiency Analysis

Table I compares the complexity of MGC with various GCNs. We find that MGC is equipped with high efficiency in training and inference stages, where both the computation and storage complexity are similar to that of SGC and S^{2}GC without storing the adjacent matrix and propagating features. However, in the preprocessing stage, directly calculating MGC operation M and conducting the feature propagation MX are computationally inefficient and result in a dense $n \times n$ size matrix M. Here, inspired by mini-batch-based GCNs [10], [18], we propose to compute feature propagation MX with b size of mini-batches to reduce the original storage cost $O(nf + n^2)$ into $O(nb + nf)$, as shown in Algorithm 1. Nevertheless, the computation cost $O(Kmn + n^2 f)$ is also expensive, especially for large graphs. To solve this problem,

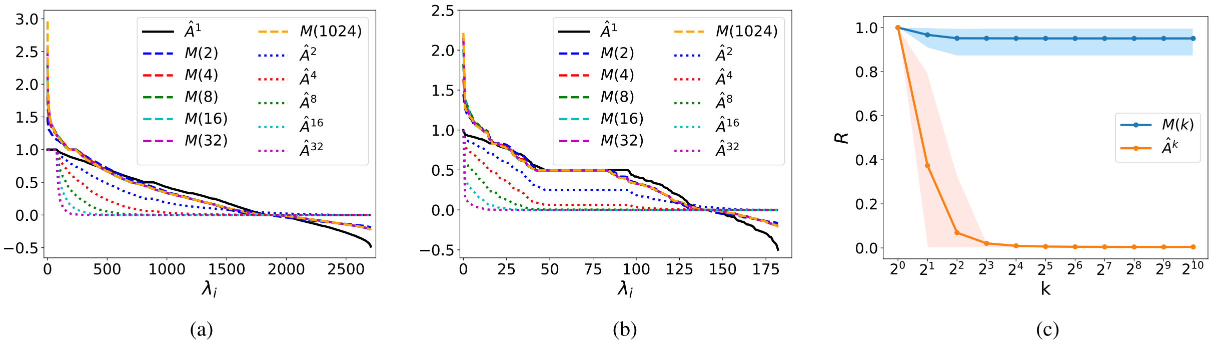


Fig. 3. Empirical analysis of the over-smoothing problem in the MGC operation. (a) and (b) Spectrum of MGC operation $M(K)$ and powers of normalized adjacent matrix $\hat{A}^K$ with different K values on Cora and Cornell datasets, where the eigenvalues λ_i are ranked in decreasing order. (c) Change of the ratio $R(S)$ with $\epsilon = 0.01$ of $M(K)$ and $\hat{A}^K$ with varying K on 300 synthetic random graphs, where the solid line corresponds to the average value and the transparent region is bounded by the 0.1 quantiles and 0.9 quantiles.

TABLE I
COMPUTATIONAL AND STORAGE COMPLEXITIES $O(\cdot)$

		Pre-processing		Training /Inference	
Type	Method	Computation	Storage	Computation	Storage
Coupled	GCN	-	-	$Knf^2 + Kmf$	$m + nf + Kf^2$
	GAT	-	-	$Knf^2 + Kmf$	$m + nf + Kf^2$
	GCNII	-	-	$Knf^2 + K(m+n)f$	$m + nf + Kf^2$
Sampling	GraphSAGE	-	-	$k^K nf^2$	$bk^K f + Kf^2$
	FastGCN	-	-	$kKnf^2$	$bkKf + Kf^2$
	Cluster-GCN	m	m	$Kmf + Knf^2$	$bKf + Kf^2$
GDC-based	SGC	Kmf	$nf + m$	$K'nf^2$	$bf + K'f^2$
	S^{2}GC	$K(m+n)f$	$nf + m$	$K'nf^2$	$bf + K'f^2$
	APPNP	-	-	$K'nf^2 + K(m+n)f$	$m + nf + K'f^2$
Our	MGC	$Kmn + n^2 f$	$nf + nb$	$K'nf^2$	$bf + K'f^2$
	AMGC	$K(m+n)f$	$nf + m$	$K'nf^2$	$bf + K'f^2$

[1] n, m, f, and k are the number of nodes, edges, feature dimensions (or hidden size), and sampled nodes respectively.
[2] K and K' denote the number of feature propagation and transformation steps, where $K = K'$ in coupled and sampling GCNs, $K' = 1$ in SGC and S^{2}GC, and $K' = 2$ in MGC.
[3] b is the batch size.

Algorithm 1 Feature Propagation MX With Mini-Batches

```
Input: Adjacent matrix $A$, feature $X$, $\alpha$, $K$, $b$;
Output: Node representation $\hat{X}$;
Computing $\hat{A} = (D + I_n)^{-1/2}(A + I_n)(D + I_n)^{-1/2}$
for $i = 0$ to $\lceil (N/b) \rceil - 1$ do
    idxs $= b \cdot i : b \cdot (i+1)$
    $P = A[\text{idxs}]$;
    $M = 0 \cdot P$;
    for $k = 1$ to $K - 1$ do
        $P = P\hat{A}$
        $M = \max\{M, P\}$
    end for
    $\hat{X}[\text{idxs}] = (1 - \alpha)(MX) + \alpha X$
end for
```

inspired by GDC-based models, where the feature propagation is conducted via iteration without computing the specific GC operation, we propose to approximate MX softly by the following AMGC, which can achieve linear complexity via intermediate multiplications $(\hat{A}(\ldots(\hat{A}X^*)\ldots))$

$$M'X = \max_{k=1}^{K}\{\hat{A}^k X^+\} - \max_{k=1}^{K}\{\hat{A}^k X^-\} \tag{13}$$

where $X^+, X^- \in R^{n \times f}$ are the positive part and negative part of X, respectively, i.e., $X_{ij}^+ = X_{ij}$ if $X_{ij} > 0$, else 0, and $X_{ij}^- = -X_{ij}$ if $X_{ij} < 0$, else 0. Note that AMGC may not be tight approximation to MGC. However, we have

$$MX = \max_{k=1}^{K}\{\hat{A}^k\}X^+ - \max_{k=1}^{K}\{\hat{A}^k\}X^-. \tag{14}$$

Indeed, AMGC approximates into MGC by moving X^+ and X^- from the outside of maximization operation into the inside, where $\hat{A}^k X^*$ can be computed with a linear complexity. Intuitively, we can derive

$$\left(A^k Z\right)_{ij} \le \left(M'Z\right)_{ij} \le (MZ)_{ij} \quad \forall\ Z > 0, i, j, k \tag{15}$$

which indicates that AMGC may achieve better approximation than any GDC-based models, at least when $X > 0$. In particular, if $X > 0$, we even have the following inequation, where $M'X$ achieves a strict tight approximation to MX than any GDC operation, i.e.,

$$\begin{aligned} \|MX - M'X\|_1 &= \sum_{ij}(MX)_{ij} - \left(M'X\right)_{ij} \\ &\le \sum_{ij}(MX)_{ij} - (SX)_{ij} = \|MX - SX\|_1. \end{aligned} \tag{16}$$

TABLE II
STATISTICS OF THE DATASETS FOR THE EVALUATION ON EFFECTIVENESS

Dataset	Class	Node	Edge	Input Dim	Train/Val/Test	$H(G)$	Assortative
Cora	7	2,708	5,429	1,433	140/500/1000	0.83	Yes
Citeseer	6	3,327	4,732	3,703	120/500/1000	0.72	Yes
Pubmed	3	19,717	44,338	500	60/500/1000	0.79	Yes
Chameleon	4	2,277	36,101	2,325	10 folds	0.25	No
Cornell	5	183	295	1,703	10 folds	0.11	No
Texas	5	183	309	1,703	10 folds	0.06	No
Wisconsin	5	251	499	1,703	10 folds	0.16	No
Actor	5	7600	33544	931	10 folds	0.24	No

V. EXPERIMENTS

Here, we evaluate MGC against the state-of-the-art GCN models on a wide variety of benchmark datasets. To be specific, we would first explore the **effectiveness** of MGC from the abilities in three aspects: capturing neighbor information, avoiding the over-smoothing problem, and handling long-distance dependency without the over-squashing problem. Then, we further estimate the **scalability** of our methods for larger graphs in both transductive and inductive settings. Finally, we turn to discuss the **efficiency** of our models.

A. Evaluation on Effectiveness

1) Datasets and Assortativity: We leveraged eight benchmark graph datasets for the node classification task to evaluate the effectiveness of MGC, including three standard citation graph benchmarks (Cora, Citeseer, and Pubmed), four web networks (Chameleon [50], Cornell, Texas, and Wisconsin [44]), and another subgraph of the knowledge graph in the film industry [57] (Actor). We applied the standard training/validation/test split for three citation graphs and conducted tenfold cross validation following [44]. Table II has summarized the statistic of those datasets. The detailed dataset description is the following.

1) Cora, Citeseer, and Pubmed are citation network benchmark datasets [53] for node classification, where nodes and edges denote documents and citations, respectively. Node features correspond to the bag-of-words embedding, and node labels are the academic topics.
2) Chameleon [50], Cornell, Texas, and Wisconsin [44] are web networks, where nodes and edges represent web pages and hyperlinks, respectively. Node features correspond to several informative nouns on the page. Those pages are manually classified by semantic topic or monthly traffic.
3) Actor is the actor-only induced subgraph of the knowledge graph in the film industry [57]. Each node corresponds to an actor, and the edge denotes co-occurrence on the same Wikipedia page. Node features correspond to some keywords with manual labels.

Actually, those datasets can be split into two categories: assortative graphs, including Cora, Citeseer, and Pubmed, with high node homophily index (i.e., neighbor nodes have the same labels and vice versa), and the disassortative graphs, including other five networks, with relative lower homophily index (i.e., nodes of the same class are far apart from each other). In particular, we follow [44] and define the homophily index $\mathcal{H}(\mathcal{G})$ as follows:

$$\mathcal{H}(\mathcal{G}) = \frac{1}{|V|}\sum_{v\in V}\frac{|\{y_{v'} = y_v | v' \in \mathcal{N}_v\}|}{|\mathcal{N}_v|} \qquad (17)$$

where $\mathcal{N}_v$ is the neighbors of the node v and y_v is the label of the node v.

Along this line, we can conduct node classification experiments on assortative graphs and disassortative graphs to evaluate the ability of GCNs to capture information from the neighbor nodes and long-distance nodes, respectively.

2) Baselines: Various GCN models are involved as baselines, which can be grouped into five categories as follows.

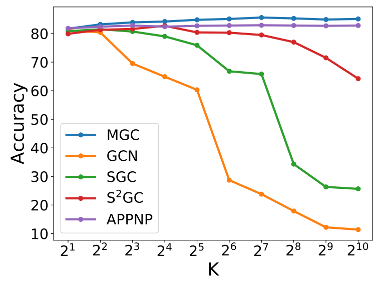
1) Vanila GCN [29] and GAT [59] are the state-of-the-art shallow models.
2) SGC [64], S^{2}GC [83], and APPNP [30] are the GDC-based models.
3) JKNet [68], GCNII [7], and GRAND [4] are GCN variants for tackling over-smoothing.
4) Geom-GCN [44] and GCN + FA [1] are GCN variants for tackling over-squashing problem.
5) DGI [60] and GIN [67] are selected as the representations of other state-of-the-art GCN variants.

3) Settings: As shown in (12), the network structure of MGC in the training stage is similar to a two-layer MLP. Here, we also use the dropout strategy in the input MX and X with dropout rate η_1 and the hidden layer with dropout rate η_2 to improve performance. Actually, the best hyperparameters are selected using grid search, where each hyperparameter is iteratively tuned within a predefined range. Specifically, parameter α is varied from 0.0 to 1.0 with a step size of 0.05. Dropout rates η_1 and η_2 are selected from [0, 0.05, 0.1, ..., 0.75]. The hidden dimension of MLP layers is selected from a range of powers of 2, ranging from 2 to 2^{11}. Similarly, the number of GCN layers K is chosen from a range of powers of 2, ranging from 2 to 2^{10}. We use Adam [28] as the optimizer. The learning rate is selected from the range [1e−5, 1.0], with each interval [1e−i, 1e−(i − 1)] divided into smaller segments using a step size of 0.5e−i, for i ranging from 5 to 1. Weight decay is determined from the range [1e−6, 1.0] ∪ {0}, with a similar division into smaller segments. We summarize the important hyperparameters of MGC for different datasets in Table III. (See Supplementary material for more parameter analysis.) In particular, all our results are based on ten random runs.

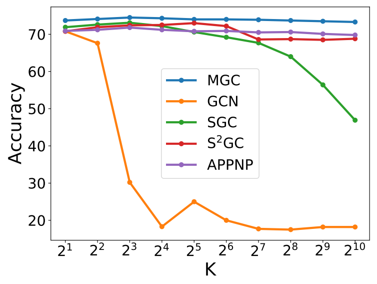
4) Effect of Capturing Neighbor Information: We followed the previous works [7], [29] and conducted experiments on the

TABLE III
HYPERPARAMETERS FOR MGC IN DIFFERENT DATASETS FOR THE EVALUATION ON EFFECTIVENESS

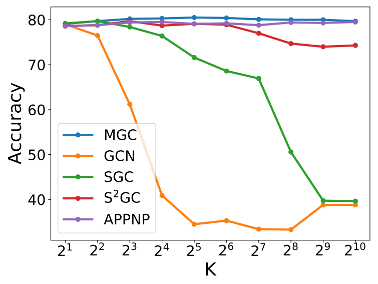
Dataset	K	α	Learning Rate	Weight Decay	Hidden Size	Dopout η_1	Dopout η_2
Cora	128	0.05	0.1	0.0003	32	0.4	0.4
Citeseer	8	0.2	0.05	0.004	128	0	0.55
Pubmed	32	0.05	0.02	0.0015	32	0.45	0.25
Chameleon	100	0	0.1	0.000001	64	0.5	0.2
Cornell	100	0.2	0.1	0.00002	256	0.3	0.4
Texas	100	0.2	0.1	0.000055	256	0.3	0.2
Wisconsin	100	0.7	0.1	0.00001	64	0.5	0.35
Actor	100	0.05	0.1	0.00001	64	0.2	0.2



(a)



(b)



(c)

Fig. 4. Prediction accuracy on (a) Cora, (b) Citeseer, and (c) Pubmed datasets with different K values.

TABLE IV
PREDICTION ACCURACY OF DIFFERENT MODELS ON ASSORTATIVE GRAPHS

Method	Cora	Citeseer	Pubmed
GCN	81.8±0.5	70.8±0.5	79.3±0.7
GAT	83.3±0.7	72.5±0.7	79.0±0.3
SGC	81.0±0.03	71.9±0.11	78.9±0.01
S^{2}GC	83.5±0.02	73.6±0.09	80.2±0.02
APPNP	83.3±0.5	71.7±0.6	80.1±0.2
JK-Net	81.8±0.5	70.7±0.7	78.8±0.7
GCNII	85.5±0.5	73.4±0.6	80.2±0.4
GCNII*	85.3±0.2	73.2±0.8	80.3±0.4
GRAND	84.7±0.6	73.6±0.3	**81.0±0.4**
DGI	82.5±0.7	71.6±0.7	78.4±0.7
GIN	77.6±1.1	66.1±0.9	77.0±1.2
MGC	**85.6±0.3**	**74.5±0.2**	80.5±0.3

assortative graphs, where the neighbor information has a large important effect on node labels. The average classification accuracy of all models with the standard deviation on the test nodes over ten random runs is summarized in Table IV. In particular, we reused the reported results of SGC, S^{2}GC, APPNP, DGI, and DIN from the paper of S^{2}GC [83], and the reported results of GCNII, GCNII*, and GRAND from their original papers [4], [7]. As for GCN and GAT, we reproduced the performance based on those code links.[1,2] As for JKNet, we implemented it by following the original paper.

The results demonstrate that our MGC model achieves better accuracy than other state-of-the-art baselines on Cora and Citeseer datasets. As for Pubmed, MGC also achieves competitive performance, especially noting that GRAND reaches the reported results for all three datasets with two variants, i.e., GRAND-l for Cora, GRAND-nl-rw for Citeseer, and Pubmed. In particular, on Citeseer, MGC is about 1.3% better than the best baseline, i.e., GRAND. It is also worth noting that MGC achieves this result with $K = 128$. It indicates that the MGC model can hold the neighbor information effectively, which is important for prediction in assortative graphs, even including long-distance neighbors.

5) Effect of Avoiding Over-Smoothing: Following [7], [22], and [83], we explored how the performance of MGC varies as the hyperparameter K increases on those three assortative graphs, i.e., Cora, Citeseer, and Pubmed, where the over-smoothing problems are often reported. To be specific, we stacked K GCN layers with both K feature propagation steps and K feature transformation steps. Also, for MGC, SGC, S^{2}GC, and APPNP, we only varied the number of feature propagation steps as different K values and fixed the number of feature transformation steps. In particular, different from the previous works [22], [83], where the number of layers or the corresponding parameter K varies in a relatively small range from 2 to 2^6, we enlarged the upper bound to 2^{10} to provide more sufficient analysis.

According to the results in Fig. 4, we find that MGC not only achieves the best performance on each dataset with almost all settings, but also avoids the significant decrease in performance as K increases, even beyond 1000. In other words, MGC can avoid over-smoothing problems with high performance, which limits the ability of other baselines. Indeed, APPNP can also avoid over-smoothing theoretically [30], but it fails to achieve the best performance. The hidden reason may lie in the tradeoff dilemma we point out in the GDC-based models, where the long-distance information is discarded. In contrast, based on Corollary 1, MGC can exploit both the short-distance and long-distance information better than any GDC-based models.

[1]https://github.com/dmlc/dgl/tree/master/examples/pytorch/gcn
[2]https://github.com/dmlc/dgl/tree/master/examples/pytorch/gat

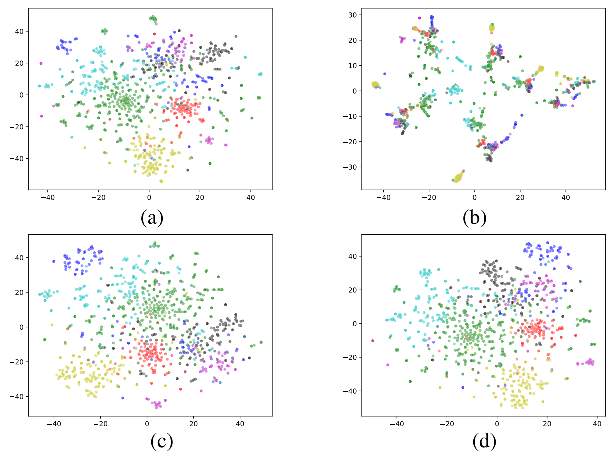


Fig. 5. t-SNE visualization of the node embeddings learned by SGC and MGC operations on Cora with the number of layers $K = 4$ or 128. (a) SGC ($K = 4$). (b) SGC ($K = 128$). (c) MGC ($K = 4$). (d) MGC ($K = 128$).

TABLE V
PREDICTION ACCURACY OF DIFFERENT MODELS ON DISASSORTATIVE GRAPHS

Method	Cham.	Corn.	Texas	Wisc.	Actor
GCN	28.18	52.70	52.16	45.88	26.86
GAT	42.93	54.32	58.38	49.41	28.45
SGC	65.09	58.11	58.38	57.65	30.98
S^2GC	64.36	59.46	59.73	60.98	32.61
APPNP	54.30	73.51	65.41	69.02	35.59
JKNet	60.07	57.30	56.49	48.82	29.81
Geom-GCN-I	60.31	56.76	57.58	58.24	29.09
Geom-GCN-P	60.90	60.81	67.57	64.12	31.63
Geom-GCN-S	59.96	55.68	59.73	56.67	30.30
GCNII*	62.48	76.49	77.84	81.57	34.50
GCN+FA	65.24	56.76	58.92	55.29	29.55
MGC	**67.21**	**85.41**	**86.22**	**86.67**	**36.71**

In addition, in order to provide an intuitive perspective about the effect of MGC on alleviating over-smoothing problems, we also show the t-SNE visualization of the learned node embeddings of the MGC operation and SGC operation on the Cora dataset with 4 and 128 layers in Fig. 5. We can find that SGC and MGC can both distinguish nodes with different labels to some extent with the shallow layers. However, SGC tends to stack nodes into multiple groups with $K = 128$, where different categories cannot be separated in each group. This phenomenon indicates the occurrence of over-smoothing. In contrast, MGC can also achieve great performance in distinguishing nodes with a large number of layers.

6) Effect of Handling Long-Distance Dependency: Here, we estimate MGC on the disassortative graphs, where the long-distance dependency between nodes is more important for classifying nodes [58]. In particular, for capturing the long-distance information, we set $K = 100$ to absorb information in the 100-hop neighborhood. The results are summarized in Table V, where some baselines have been removed due to the lack of competitiveness. In particular, we reused the reported results of Geom-GCN-I, Geom-GCN-P, and Geom-GCN-S from the original paper [44], and the reported results of GCN, GAT, APPNP, JKNet, and GCNII* on Chameleon, Corn, Texas, and Wisconsin datasets from the GCNII paper [7]. Furthermore, we implemented GCN, GAT, APPNP, and GCNII* on the Actor dataset and reproduced SGC, S^2GC on all five datasets with the code links.[3],[4] We reproduced JKNet on Actor and GCN + FA on all five datasets by ourselves based on the description in the original papers.

Based on the results, we find that MGC can outperform other baselines by a significant margin on all five datasets, which demonstrates the powerful ability of MGC to handle long-distance dependency. In other words, MGC can relieve the over-squashing problem effectively. Meanwhile, we note that the most competitive baseline, GCNII*, is equipped with a large number of layers for both feature propagation and feature transformation [7], which cause the high training and test time cost and even optimization problem. In contrast, benefitting from the design of separating the feature propagation into the preprocessing stage, MGC only needs to conduct the feature transformation based on a simple two-layer MLP model with low time complexity in the training and test stages. In addition, GCN + FA achieves a significant improvement compared with its base model, i.e., GCN, by connecting all nodes to link the long-distance nodes. However, it is not competitive on most graph datasets. It may be because connecting all nodes will also include unnecessary noise information.

B. Evaluation on Scalability

1) Datasets: Here, we turn to estimate the performance of our models on large-scale graph learning with five large graphs, including three large-scale OGB benchmark datasets for transductive learning (Arxiv, Products, and Papers100M) and two graphs for inductive learning (Flickr and Reddit). Standard splits are applied in all five datasets. Table VI summarized the statistic of those datasets. The detailed descriptions can be found as follows.

1) Ogbn-Arxiv and Ogbn-Papers100M are two citation networks between all computer science (CS) arXiv papers indexed by MAG [62] in different scales. Each node is a paper, and each edge indicates the citation relation between two papers. Node features are based on the word2vec embedding of the title and abstract. The subject area of each paper is set as the label. Ogbn-Papers100M is much larger than most existing public node classification datasets with 111M nodes.
2) MAG240M is another huge heterogeneous academic graph [20] extracted from MAG [62], consisting of 121M paper nodes and 122M authors nodes. Here, only the paper nodes and the corresponding citation links are used in our experiments. Node features are based on the RoBERTa sentence encoder [39], [49]. The subject area of each paper is set as the label.
3) Ogbn-Products is an Amazon product copurchasing network [10]. Nodes represent products, and edges between two products indicate that the products are purchased together. Node features are based on the bag-of-words features from the product descriptions. The top-level categories of products are set as labels.

[3]https://github.com/Tiiiger/SGC
[4]https://github.com/allenhaozhu/SSGC

TABLE VI
STATISTICS OF THE DATASETS FOR THE EVALUATION ON SCALABILITY

Dataset	Class	Node	Edge	Input Dim	Train/Val/Test	Setting
Ogbn-Arxiv	40	169,343	1,166,243	128	9K/30K/49K	Transductive
Obgn-Products	47	2,449,029	61,859,140	100	20K/39K/221K	Transductive
Obgn-Papers100M	172	111,059,956	1,615,685,872	128	1,207K/125K/214K	Transductive
MAG240M	153	121,751,666	1,297,748,926	768	1,112K/138K/88K	Transductive
Flickr	7	89,250	899,756	500	44K/22K/22K	Inductive
Reddit	41	232,965	11,606,919	602	155K/23K/22K	Inductive

TABLE VII
HYPERPARAMETERS FOR MGC AND AMGC IN DIFFERENT DATASETS FOR THE EVALUATION ON SCALABILITY

Method	Dataset	K	α	Learning Rate	Weight decay	Hidden Size	Dopout η_1	Dopout η_2
MGC	Ogbn-Arxiv	16	0.05	0.00015	0.000015	1024	0	0.4
	Flickr	8	0	0.0001	0.00001	512	0	0.2
	Reddit	8	0.2	0.0002	0.00001	1024	0.2	0.4
AMGC	Ogbn-Arxiv	16	0.05	0.00015	0.000015	1024	0	0.4
	Ogbn-Products	8	0.05	0.001	0	1024	0.3	0.2
	Ogbn-Papers100M	8	0.05	0.001	0	2048	0	0
	MAG240M	8	0.05	0.00003	0.0001	2048	0.05	0.05
	Flickr	8	0.05	0.0001	0.00001	512	0	0.2
	Reddit	8	0.2	0.0002	0.00001	1024	0.2	0.4

TABLE VIII
PREDICTION ACCURACY OF DIFFERENT MODELS WITH THE TRANSDUCTIVE SETTING

Method	Arxiv	Products	Papers100M	MAG240M
MLP	55.50±0.23	61.06±0.08	47.24±0.31	52.67
GCN	71.74±0.29	75.64±0.21	OOM	OOM
GraphSAGE	71.49±0.27	78.29±0.16	**67.06±0.17**	**66.79**
ClusterGCN	70.20±0.13	78.97±0.33	-	-
SGC	68.78±0.02	68.87±0.01	63.29±0.19	65.82
S^{2}GC	70.15±0.13	70.22±0.01	64.70±0.30	66.08±0.06
S^{2}GC+MLP	72.01±0.25	76.84±0.20	65.73±0.15	66.46±0.07
MGC	**72.83±0.10**	OOM	OOM	OOM
AMGC	72.22± 0.13	76.78±0.08	65.42±0.17	66.18±0.08
AMGC+MLP	72.38±0.12	**79.68±0.15**	65.87±0.15	66.69±0.05

4) Flickr is graphs with images as nodes [75]. The edge between two nodes denotes that those two images share some common properties (e.g., the same geographic location). Node features are the bag-of-word representation of the image descriptions. Each image is classified into one of the seven classes manually.
5) Reddit is a well-known inductive training dataset [18], which is derived from the community structure of numerous Reddit posts. It is a post-to-post graph, connecting posts if the same user comments on both. Node features are based on the word2vec vectors of the post. The community or "subreddit" that a post belongs to is set as the node label.

2) Baselines: Various models for large-scale graph learning are set as baselines, including the following.

1) MLP is a two-layer neural network with the node features as the input.
2) Vanila GCN [29] has also been involved for common comparisons.
3) GraphSAGE [18] and ClusterGCN [10] are both sampling-based GCNs and reduce the storage complexity via sampling subgraphs from the large-scale input graphs in each mini-batch.
4) SGC [64] and S^{2}GC [83] are GDC-based models. Note that APPNP is omitted, because it is not suitable for scalable graph learning where the whole graph structure must be involved in both forward and backward passes during training.

In addition, following S^{2}GC, where the authors claim that MLP plays a more important role in OGB datasets, we achieve a variant of AMGC, i.e., AMGC + MLP, where the liner layers in (12), i.e., F_1, F_2, and F_3, are replaced with a two-layer MLP, respectively.

3) Settings: Here, we chose the best hyperparameters using grid search with the same predefined ranges as those in Section V-A3. As a result, we list the important hyperparameters of our models for different datasets in Table VII. In particular, as for AMGC + MLP, we use the same hyperparameter as that of AMGC for a fair comparison.

4) Results: The results are summarized in Tables VIII and IX. Specifically, in Table VIII, we reused the reported results of MLP, GCN, and GraphSAGE on the OGB leaderboards,[5] except the result of GraphSAGE on Ogbn-Products, which is from S^{2}GC paper [83]. The result of ClusterGCN

[5]https://ogb.stanford.edu/docs/leader_nodeprop/

TABLE IX
PREDICTION ACCURACY OF DIFFERENT MODELS WITH THE INDUCTIVE SETTING

Method	Flickr	Reddit
MLP	45.6± 0.1	75.6±0.1
GCN	49.2±0.3	93.3±0.0
GraphSAGE	50.1± 1.3	95.4± 0.0
ClusterGCN	48.1±0.5	96.6±0.0
SGC	50.2±0.1	94.9±0.0
S^{2}GC	50.8±0.1	95.3±0.0
MGC	**52.1±0.1**	**96.7±0.0**
AMGC	52.0±0.1	96.3±0.0

TABLE X
TIME COST IN SECONDS FOR DIFFERENT MODELS ON FLICKR AND REDDIT

Method	Flickr			Reddit		
	Pre.	Train	Acc.	Pre.	Train	Acc.
SGC	0.8	6.3	50.2	4.4	12.1	94.9
SSGC	0.8	6.8	50.8	4.6	12.3	95.3
GraphSAGE	-	387.2	50.1	-	661.2	95.4
ClusterGCN	0.7	85.7	48.1	117.8	426.9	96.6
MGC	25.3	13.2	52.1	2444.1	30.0	96.7
AMGC	1.0	13.1	52.0	9.1	29.6	96.3

on Ogbn-Products is also from the OGB leaderboards. The results of SGC, S^{2}GC, and S^{2}GC + MLP on Ogbn-Arxiv and Ogbn-Products are copied from S^{2}GC paper [83]. The results of SGC on Ogbn-Papers100M and MAG240M are also from the OGB leaderboards. In particular, we show the results of MAG240M on its validated set due to the inaccessibility of the labels of the test dataset. Other results in this table are based on our implementation with the guidance of their original papers. Besides, we have not found the result of ClusterGCN on Ogbn-Papers100M and MAG240M in previous works. Meanwhile, We cannot conduct this experiment due to the large memory requirements, although it can be achieved theoretically. In Table IX, we reused the reported results of GraphSAGE, ClusterGCN, SGC, and S^{2}GC on Reddit from its original paper [10], [18], [64], [83]. We reproduced other results of baselines with the guidance of their original papers. In particular, MLP is implemented as two linear layers with ReLU activation and dropout in the hidden layer. The size of the hidden layer is the same as that of our models.

Based on the results, we first find that MGC cannot be conducted on Products, Papers100M, and MAG240M due to the high complexity in the preprocessing stage. In contrast, AMGC can be applied to all five large datasets. We also find that our models have achieved competitive performance in both transductive and inductive settings. Specifically, our models outperform GDC-based models consistently on all five datasets. In particular, with MLP in OGB datasets, both S^{2}GC + MLP and AMGC + MLP can achieve a significant improvement, but AMGC + MLP performs better. Compared with sampling-based GCNs, our models achieve the best performance on four datasets. As for Papers100M and MAG120M datasets, our models cannot outperform GraphSAGE with deliberated network structures. However, as shown in Table I, GraphSAGE is computation cost during training or inference with exponential complexity in the number of feature propagation steps. In addition, as an approximated variant of MGC, AMGC has achieved close performance to MGC, although with a linear complexity. In other words, AMGC is an efficient approximation for large-scale graphs.

C. Evaluation on Efficiency

1) Baselines and Settings: Here, we analyze the efficiency of our models by measuring the time cost on Flickr and Reddit datasets. To be specific, we use GraphSAGE, ClusterGCN, SGC, and S^{2}GC as baselines. In GraphSAGE and ClusterGCN, we used the same hyperparameter settings as their original papers on Reddit. As for Flickr, we set batch size as 512 and hidden size as 128 in GraphSAGE and set cluster number as 500 and batch size as 10 in ClusterGCN, where the best performance was achieved. As for SGC, S^{2}GC, and our models, i.e., MGC and AMGC, we used the mini-batch training strategy in the training stage with a batch size of 2048. All models are trained with 100 epochs. In particular, the batch size of MGC in the preprocessing stage is set as 256 here.

2) Results: Table X summarizes the time cost of different models in preprocessing and training stages. First, we can find that our models are efficient in training strategy with high accuracy. Second, MGC is time-costing in the preprocessing stage due to the high-computational complexity shown in Table I. However, the approximated approach, i.e., AMGC, enjoys an efficient preprocessing stage with competitive performance. In particular, the large difference between the time cost of MGC in the preprocessing stage in two datasets is caused by the different scales of two graphs (89K nodes and 900K edges versus 232K nodes and 12M edges).

VI. RELATED WORKS

In this article, we have focused on GDCs and discussed their tradeoff in addressing the over-smoothing and over-squashing issues. Here, we provide additional analysis of other related literature for GCNs [66].

To mitigate the over-smoothing problem, two popular directions have emerged, namely, modifications to the network architecture and the graph structure. First, inspired by convolutional neural networks (CNNs), dilated convolutions and residual connections have been incorporated into GCNs to create deeper architectures [34]. Besides directly adding residual connections to initial node features, their transformed features by a trainable projection are also proven to be effective [9] with the help of graph-regularization optimization. In addition, normalization techniques, such as batch normalization, have been employed to prevent nodes from becoming indistinguishable [35], [37], [69], [81], [82], even normalizing the learnable weights of GCNs [42]. More recently, nonlinear aggregators were also used in the message propagation mechanism in GCNs to enhance the network's capacity and robustness [61]. Second, better graph structure can be inferred from node embedding to alleviate over-smoothing, which, in turn, results in more expressive node embeddings [8], [23]. Localized subgraphs for each node can also be utilized to restrict the scope of GCNs and enable the network to reach arbitrary depths [25], [74]. However, these methods solely focus on the over-smoothing problem without over-squashing.

Regarding the over-squashing problem, most solutions concentrate on graph rewiring techniques, such as adding fully connected layers [1] or modifying the graph structure by adding or removing edges based on the analysis of graph curvature [3], [11], [13], [58] or graph spectrum [24], [55]. However, recent studies have reported that those methods can also result in a tradeoff between the over-smoothing and over-squashing problems, supported by empirical evidence [27], [38] or theoretical analysis [15], [41]. In addition, different from our methods, graph rewiring methods often require high complexity and are not easily extendable to large graphs. Recently, some other works also attempted to enhance the long-range interactions by defining multihop message passing layer directly [17] or skipping the convolution operation for some nodes randomly [40]. However, they usually introduce expensive training costs or several unnecessary noises compared with our methods.

VII. Conclusion

In this article, we find that GDC-based models can only achieve a tradeoff between tackling over-smoothing and over-squashing problems. To this end, instead of using the linear combination, such as that in GDC-based models, we proposed a simple yet effective GC, i.e., MGC, by applying an elementwise maximizing operation for exploiting all possible powers of the normalized adjacent matrix. MGC can alleviate over-squashing better than any GDC-based models theoretically and avoid over-smoothing with several empirical evidences. Furthermore, an efficient approximated approach with a linear complexity is developed for large-scale graph learning. Finally, extensive experiments on various benchmark datasets have been conducted to compare the effectiveness, scalability, and efficiency of different models. In particular, our models achieve competitive performance with lower complexity, even on large graphs with more than 100M nodes.

References

[1] U. Alon and E. Yahav, “On the bottleneck of graph neural networks and its practical implications,” in *Proc. 9th Int. Conf. Learn. Represent.*, 2021.

[2] T. Bian et al., “Rumor detection on social media with bi-directional graph convolutional networks,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 549–556.

[3] M. Black, Z. Wan, A. Nayyeri, and Y. Wang, “Understanding over-squashing in GNNs through the lens of effective resistance,” in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 2528–2547.

[4] B. P. Chamberlain, J. Rowbottom, M. Goronova, S. Webb, E. Rossi, and M. M. Bronstein, “GRAND: Graph neural diffusion,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1407–1418.

[5] J. Chen, L. Yang, C. Qin, Y. Yang, L. Peng, and X. Ge, “Heterogeneous graph traffic prediction considering spatial information around roads,” *Int. J. Appl. Earth Observ. Geoinf.*, vol. 128, Apr. 2024, Art. no. 103709.

[6] L. Chen et al., “Collaboration-aware hybrid learning for knowledge development prediction,” in *Proc. ACM Web Conf.*, vol. 38, May 2024, pp. 3976–3985.

[7] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1725–1735.

[8] Y. Chen, L. Wu, and M. Zaki, “Iterative deep graph learning for graph neural networks: Better and robust node embeddings,” in *Proc. NIPS*, vol. 33, 2020, pp. 19314–19326.

[9] Z. Chen, Z. Wu, Z. Lin, S. Wang, C. Plant, and W. Guo, “AGNN: Alternating graph-regularized neural networks to alleviate over-smoothing,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, May 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10138925

[10] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, “Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 257–266.

[11] F. Di Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lio, and M. M. Bronstein, “On over-squashing in message passing neural networks: The impact of width, depth, and topology,” in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 7865–7885.

[12] C. Fang et al., “RecruitPro: A pretrained language model with skill-aware prompt learning for intelligent recruitment,” in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2023, pp. 3991–4002.

[13] L. Fesser and M. Weber, “Mitigating over-smoothing and over-squashing using augmentations of forman-RICCI curvature,” in *Proc. Learn. Graphs Conf.*, 2024, pp. 1–19.

[14] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proc. 34th Int. Conf. Mach. Learning*, vol. 70, 2017, pp. 1263–1272.

[15] J. H. Giraldo, K. Skianis, T. Bouwmans, and F. D. Malliaros, “On the trade-off between over-smoothing and over-squashing in deep graph neural networks,” in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2023, pp. 566–576.

[16] Q. Guo et al., “A survey on knowledge graph-based recommender systems,” *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3549–3568, Aug. 2022.

[17] B. Gutteridge, X. Dong, M. M. Bronstein, and F. Di Giovanni, “DRew: Dynamically rewired message passing with delay,” in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 12252–12267.

[18] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[19] P. W. Holland, K. B. Laskey, and S. Leinhardt, “Stochastic blockmodels: First steps,” *Social Netw.*, vol. 5, no. 2, pp. 109–137, Jun. 1983.

[20] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, “OGB-LSC: A large-scale challenge for machine learning on graphs,” 2021, *arXiv:2103.09430*.

[21] H. Huang, H. Tian, X. Zheng, X. Zhang, D. D. Zeng, and F.-Y. Wang, “CGNN: A compatibility-aware graph neural network for social media bot detection,” *IEEE Trans. Computat. Social Syst.*, pp. 1–16, Jun. 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10549990

[22] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, “Tackling over-smoothing for general graph convolutional networks,” 2020, *arXiv:2008.09864*.

[23] C. Huo, D. Jin, Y. Li, D. He, Y. B. Yang, and L. Wu, “T2-GNN: Graph neural networks for graphs with incomplete features and structure via teacher–student distillation,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 4339–4346.

[24] A. Jamadandi, C. Rubio-Madrigal, and R. Burkholz, “Spectral graph pruning against over-squashing and over-smoothing,” 2024, *arXiv:2404.04612*.

[25] B. Jiang, Y. Chen, B. Wang, H. Xu, and B. Luo, “DropAGG: Robust graph neural networks via drop aggregation,” *Neural Netw.*, vol. 163, pp. 65–74, Jun. 2023.

[26] F. Jiang et al., “Towards efficient resume understanding: A multi-granularity multi-modal pre-training approach,” 2024, *arXiv:2404.13067*.

[27] K. Karhadkar, P. K. Banerjee, and G. Montúfar, “FoSR: First-order spectral rewiring for addressing over-squashing in GNNs,” in *Proc. 11th Int. Conf. Learn. Represent.*, 2023.

[28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, *arXiv:1412.6980*.

[29] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. 5th Int. Conf. Learn. Represent.*, Toulon, France, Apr. 2017.

[30] J. Klicpera, A. Bojchevski, and S. Günnemann, “Predict then propagate: Combining neural networks with personalized pagerank for classification on graphs,” in *Proc. 6th Int. Conf. Learn. Represent.*, 2018.

[31] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 13354–13366.

[32] B. Li and S. Nabavi, "A multimodal graph neural network framework for cancer molecular subtype classification," *BMC Bioinf.*, vol. 25, no. 1, p. 27, Jan. 2024.

[33] F. Li et al., "Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution," *ACM Trans. Knowl. Discovery Data*, vol. 17, no. 1, pp. 1–21, Feb. 2023.

[34] G. Li et al., "DeepGCNs: Making GCNs go as deep as CNNs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 6923–6939, Jun. 2023.

[35] J. Li, Q. Zhang, W. Liu, A. B. Chan, and Y.-G. Fu, "Another perspective of over-smoothing: Alleviating semantic over-smoothing in deep GNNs," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, May 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10540641

[36] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.

[37] L. Liang, Z. Xu, Z. Song, I. King, Y. Qi, and J. Ye, "Tackling long-tailed distribution issue in graph neural networks via normalization," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 5, pp. 2213–2223, May 2024.

[38] Y. Liu et al., "CurvDrop: A Ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 221–230.

[39] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[40] W. Lu et al., "SkipNode: On alleviating performance degradation for deep graph convolutional networks," *IEEE Trans. Knowl. Data Eng.*, pp. 1–14, Apr. 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10497904

[41] K. Nguyen, N. M. Hieu, V. D. Nguyen, N. Ho, S. Osher, and T. M. Nguyen, "Revisiting over-smoothing and over-squashing using ollivier-ricci curvature," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 25956–25979.

[42] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019.

[43] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep. SIDL-WP-1999-0120, Nov. 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[44] H. Pei, B. Wei, K. Chen-Chuan Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," 2020, *arXiv:2002.05287*.

[45] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 13071–13102, Nov. 2023.

[46] C. Qin et al., "A comprehensive survey of artificial intelligence techniques for talent analytics," 2023, *arXiv:2307.03195*.

[47] C. Qin et al., "Automatic skill-oriented question generation and recommendation for intelligent job interviews," *ACM Trans. Inf. Syst.*, vol. 42, no. 1, pp. 1–32, Jan. 2024.

[48] Y. Qin, W. Ju, H. Wu, X. Luo, and M. Zhang, "Learning graph ODE for continuous-time sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3224–3236, Jul. 2024.

[49] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," 2019, *arXiv:1908.10084*.

[50] B. Rozemberczki, C. Allen, R. Sarkar, and x. T. Gross, "Multi-scale attributed node embedding," *J. Complex Netw.*, vol. 9, no. 1, pp. 1–22, Apr. 2021.

[51] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018.

[52] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.

[53] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–106, Sep. 2008.

[54] D. Shen, C. Qin, C. Wang, Z. Dong, H. Zhu, and H. Xiong, "Topic modeling revisited: A document graph-based neural network perspective," in *Proc. Adv. neural Inf. Process. Syst.*, vol. 34, 2021, pp. 14681–14693.

[55] X. Shen, P. Lio, L. Yang, R. Yuan, Y. Zhang, and C. Peng, "Graph rewiring and preprocessing for graph neural networks based on effective resistance," *IEEE Trans. Knowl. Data Eng.*, pp. 1–14, Feb. 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10521752

[56] Y. Sun, F. Zhuang, H. Zhu, Q. Zhang, Q. He, and H. Xiong, "Market-oriented job skill valuation with cooperative composition neural network," *Nature Commun.*, vol. 12, no. 1, p. 1992, Mar. 2021.

[57] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun. 2009, pp. 807–816.

[58] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," 2021, *arXiv:2111.14522*.

[59] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018.

[60] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. ICLR*, 2019, vol. 2, no. 3, p. 4.

[61] B. Wang, B. Jiang, J. Tang, and B. Luo, "Generalizing aggregation functions in GNNs: Building high capacity and robust GNNs via nonlinear aggregation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13454–13466, Jun. 2023.

[62] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: When experts are not enough," *Quant. Sci. Stud.*, vol. 1, no. 1, pp. 396–413, Feb. 2020.

[63] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6857–6866.

[64] F. Wu, A. H. Souza, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6861–6871.

[65] W. Wu, C. Wang, D. Shen, C. Qin, L. Chen, and H. Xiong, "AFDGCF: Adaptive feature de-correlation graph collaborative filtering for recommendations," 2024, *arXiv:2403.17416*.

[66] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[67] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. 6th Int. Conf. Learn. Represent.*, 2018.

[68] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5453–5462.

[69] R. Yang, W. Dai, C. Li, J. Zou, and H. Xiong, "Tackling over-smoothing in graph convolutional networks with EM-based joint topology optimization and node classification," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 9, pp. 123–139, Feb. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10379114

[70] K. Yi, B. Zhou, Y. Shen, P. Liò, and Y. Wang, "Graph denoising diffusion for inverse protein folding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 10238–10257.

[71] X. Yu et al., "RDGT: Enhancing group cognitive diagnosis with relation-guided dual-side graph transformer," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3429–3442, Jul. 2024.

[72] X. Yu et al., "RIGL: A unified reciprocal approach for tracing the independent and group learning processes," 2024, *arXiv:2406.12465*.

[73] W. W. Zachary, "An information flow model for conflict and fission in small groups," in *Proc. J. Anthropological Res.*, 1977, vol. 33, no. 4, pp. 452–473.

[74] H. Zeng et al., "Decoupling the depth and scope of graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 19665–19679.

[75] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph sampling based inductive learning method," 2019, *arXiv:1907.04931*.

[76] R. Zha et al., "Career mobility analysis with uncertainty-aware graph autoencoders: A job title transition perspective," *IEEE Trans. Computat. Social Syst.*, vol. 11, no. 1, pp. 1–11, Feb. 2023.

[77] L. Zhang et al., "Large-scale talent flow embedding for company competitive analysis," in *Proc. Web Conf.*, Apr. 2020, pp. 2354–2364.

[78] L. Zhang et al., "Attentive heterogeneous graph embedding for job mobility prediction," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 2192–2201.

[79] M. Zhang et al., "GraphTranslator: Aligning graph model to large language model for open-ended tasks," in *Proc. ACM Web Conf.*, May 2024, pp. 1003–1014.
[80] W. Zhang et al., "Node dependent local smoothing for scalable graph learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 20321–20332.
[81] L. Zhao and L. Akoglu, "PairNorm: Tackling oversmoothing in GNNs," 2019, *arXiv:1909.12223*.
[82] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, "Towards deeper graph neural networks with differentiable group normalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 4917–4928.
[83] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020.

Dazhong Shen (Member, IEEE) received the Ph.D. degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2023.

He is currently a Researcher with Shanghai Artificial Intelligence Laboratory, Shanghai, China. He has authored more than 20 journal and conference papers in the fields of generative model and graph learning, including the *ACM Transactions on Information Systems* (TOIS), *ACM Transactions on Management Information Systems* (TMIS), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, NeurIPS, CVPR, IJCAI, and WWW.

Chuan Qin (Member, IEEE) received the Ph.D. degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2021.

He is currently a Post-Doctoral Researcher with the PBC School of Finance, Tsinghua University, Beijing, China. He has authored more than 50 journals and conference papers in the fields of natural language processing and recommender systems, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, *ACM Transactions on Information Systems*, SIGKDD, SIGIR, WWW, ICDE, NeurIPS, AAAI, IJCAI, and ICDM.

Dr. Qin has been honored with the Excellent Award from the President of Chinese Academy of Sciences in 2021, a nomination for the Baidu Scholarship (top 20 globally) in 2021, and the Best Student Paper Award of SIGKDD-2018.

Qi Zhang (Member, IEEE) received the Ph.D. degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2023.

He is currently a Post-Doctoral Researcher with Shanghai Artificial Intelligence Laboratory, Shanghai, China. He has authored ten journal and conference papers in the fields of data mining and artificial intelligence, including *ACM Transactions on Information Systems*, ACM IMWUT, KDD, SIGIR, *Nature Communications*, and ICDM. His research interests include data mining, artificial intelligence, sequential modeling, embodied AI, and natural language processing.

Hengshu Zhu (Senior Member, IEEE) received the B.E. and Ph.D. degrees in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2009 and 2014, respectively.

He has published prolifically in refereed journals and conference proceedings, such as *Nature Communications*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON MOBILE COMPUTING, *ACM Transactions on Information Systems*, SIGKDD, SIGIR, and NeurIPS. His research interests include data mining and machine learning, with a focus on developing advanced data analysis techniques for innovative business applications.

Dr. Zhu is a Senior Member of ACM, CAAI, and CCF. He served as the Program Co-Chair for KDD CUP-2019 Regular ML Track, the Industry Chair for PRICAI-2022, the Area Chair for KDD, AAAI, and IJCAI, and regularly as the (senior) program committee members in numerous top conferences. He was a recipient of the Distinguished Dissertation Award of CAS in 2016, the Distinguished Dissertation Award of CAAI in 2016, the Special Prize of President Scholarship for Postgraduate Students of CAS in 2014, the Best Student Paper Award of KSEM-2011, WAIM-2013, and CCDM-2014, and the Best Paper Nomination of ICDM-2014 and WSDM-2022.

Hui Xiong (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, USA, in 2005.

He is currently a Chair Professor, an Associate Vice President (Knowledge Transfer), and the Founding Head of the AI Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests include artificial intelligence, data mining, and mobile computing.

Dr. Xiong has served on numerous organization and program committees for conferences, including the Founding Editor-in-Chief for *npj Artificial Intelligence*, the Co-Editor-in-Chief for the *Encyclopedia of GIS* (Springer), the Program Co-Chair for the Industrial and Government Track for the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), the Program Co-Chair for the IEEE 2013 International Conference on Data Mining (ICDM), the General Co-Chair for the 2015 IEEE International Conference on Data Mining (ICDM), and the Program Co-Chair for the Research Track for the 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. He received several awards, such as the 2021 AAAI Best Paper Award and the 2011 IEEE ICDM Best Research Paper Award. For his significant contributions to data mining and mobile computing, he was elected as a fellow of AAAS in 2020.