Applied Intelligence (2024) 54:8092–8107
https://doi.org/10.1007/s10489-024-05601-z

Multi-head multi-order graph attention networks

Jie Ben[1] · Qiguo Sun[1] · Keyu Liu[2] · Xibei Yang[1] · Fengjun Zhang[1]

Accepted: 8 June 2024 / Published online: 20 June 2024

Abstract
The Graph Attention Network (GAT) is a type of graph neural network (GNN) that uses attention mechanisms to weigh the importance of nodes' neighbors, demonstrating flexibility and power in representation learning. However, GAT and its variants still face common challenges in GNNs, such as over-smoothing and over-squashing. To address this, we propose Multi-Head Multi-Order Graph Attention Networks (MHMOGAT) as an enhanced GAT layer. MHMOGAT is built based on multi-head attention and adjacency matrices of different orders, aiming to expand the receptive field of GAT to effectively capture long-distance dependencies. Moreover, Bayesian optimization is employed to determine optimal hyperparameter combinations for different datasets. Experimental results on six prevailing datasets demonstrate that MHMOGAT improves GAT accuracy by approximately 2-5% across various datasets with different label rates, indicating its effectiveness. Additionally, MHMOGAT exhibits potential in handling large and complex graphs with low label rates.

Keywords Semi-supervised learning · Graph convolutional network · Attention mechanism · Multi-order information

1 Introduction

Graph data [1] is a natural representation of many complex systems in the real world, such as social networks [2], recommendation systems, biological networks, transportation networks, and more [3–6]. However, in many applications of graph data, label information about nodes is often scarce [7, 8], which limits the applicability and effectiveness of traditional supervised learning approaches. As such, semi-supervised learning methods [9–11] have emerged as powerful tools for these scenarios by leveraging information from unlabeled nodes to enhance model performance. Semi-supervised learning is a machine learning paradigm [12, 13] and there are various methods, including the use of generative models [14, 15], semi-supervised support vector machines [16, 17], graph models [18, 19], and so on. Through these methods, semi-supervised learning can better cope with the incomplete data in the real world and improve the generalization ability of the model.

On the other hand, graph convolutional networks (GCNs) [18] have achieved significant advancements in semi-supervised learning. These networks facilitate information transmission based on the topology between nodes, effectively capturing relational data within graph structures. GCNs demonstrated their potency across various downstream tasks [20–22]. The message passing mechanism in GCNs, while useful for utilizing graph structure information by adding more layers to have an extensive receptive field, can lead to the over-smoothing problem when combined with increased depth [23, 24].

Although GCNs have significant potential for modeling graph-structured data, it has been revealed that they encounter the issues of over-smoothing and over-squashing [25, 26]. over-smoothing occurs as node features become increasingly similar with the increase in convolutional layers, while over-squashing arises from significant information

✉ Qiguo Sun
sunqiguo1992@gmail.com

Jie Ben
benjjie@foxmail.com

Keyu Liu
keyu_liu@my.swjtu.edu.cn

Xibei Yang
jsjxy_yxb@just.edu.cn

Fengjun Zhang
jszhfj@foxmail.com

[1] School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212100, Jiangsu, China

[2] School of Computing and Artificial Intelligence, Southwest Jiaotong University Chengdu, Chengdu 611756, Sichuan, China

compression via bottleneck edges. Various methods have been proposed to tackle these challenges [25–29]. Specifically, Graph Attention Networks [27] utilize attention mechanisms to learn discriminative features, thus reducing the over-smoothing problem. Building upon this, [28] further developed a high-order graph attention network that effectively integrates multi-hop neighbor information for node representation. Moreover, [29] developed DropEdge, which uniformly drops out a certain number of edges during model training to relieve the over-smoothing problem. However, these methods cannot effectively address the over-squashing problem, especially for graphs with large diameters and long-range dependencies between nodes. [30] pointed out that GNNs struggle to transmit messages effectively from distant nodes and demonstrate degraded performance when tasked with predictions relying on interactions spanning long distances. They formally introduced the over-squashing phenomenon of GNNs but did not present a new architecture to solve the problem. Later, [31] employed the Stochastic Jost and Liu Curvature Rewiring algorithm to perform edge addition and removal during GNN training, achieving a suitable compromise between over-smoothing and over-squashing.

To mitigate the over-smoothing and over-squashing problems and increase GCNs' expressiveness, this paper proposes a method called multi-head multi-order graph attention networks(MHMOGAT). We integrated multi-head attention with varying neighborhood orders, resulting in a model that incorporates both multi-order and cross-order multi-head attentions. In the multi-head attention model, each head focus on learning information from different neighborhood orders. This allows for learning at an expanded perception field and capturing relationships within the graph more accurately. In our cross-order setup, we use multiple channels with various neighborhood orders to achieve a cross-order effects. This approach enhances generalization ability of the model by capturing multiple hop's information. Allowing the model to solve different challenging classification tasks. Our model defines a novel self-attention mechanism that evaluates the influence of both immediate neighbors and long-distance nodes, while assigning heterogeneous multi-head attention to enhance feature extraction across different layers. In addition, Bayesian optimization is integrated into the model for hyperparameter optimization.

The main contributions of this paper can be summarized as follows:

- Develop new multi-head multi-order GCN layers which improve GAT performance significantly on various semi-supervised classification datasets.
- Tackle the limitation of GAT in capturing multi-hop information and expand its perception field.
- Show potential in dealing with large and complex graphs with very low label rates.

The remaining structure of this article is as follows: Section 2 covers the related work in this field. In Section 3, we provide an overview of the network architectures of GAT and MOGCN, along with highlighting some of the challenges they face. Our proposed model MHMOGAT is introduced, and its two implementation approaches are described in Section 4. In Section 5, our model will be compared with some other state-of-the-art models to demonstrate its effectiveness. Finally, Section 6 presents the conclusions of this study and discusses future work.

2 Related work

2.1 High-Order Graph Convolutional Networks

Recent works have focused on encoding high-order neighborhood information from target nodes to improve performance. MOGCN [32] constructs multiple simple GCN learners with multi-order adjacency matrices to capture high-order connectivity among the nodes. MixHop [33] mixes powers of the adjacency matrix, which can mix feature representations of neighbors at various distances to learn neighborhood mixing relationships. HCNP [34] can simultaneously aggregate information from various neighborhoods by constructing high-order convolutions. Additionally, similar to our model, both HGRN [28] and MulStepNET [35] combine the attention mechanism with high-order neighborhoods to improve the model's learning ability. However, HGRN and MulStepNET simply combine the attention mechanism with higher-order information without considering the effective combination of multi-head and multi-order information to make the model more flexible and efficient.

2.2 Over-smoothing and over-squashing

In the realm of GNNs, over-smoothing and over-squashing stand out as significant limitations. To address these challenges, numerous strategies have been devised. For instance, Huang et al. [29] proposed a methodology centered on the removal of graph edges during GNN training, while Zhao et al. [36] and Zhou et al. [37] introduced novel normalization techniques tailored to directly counteract over-smoothing effects. Additionally, Chien et al. [38] developed a novel graph convolutional filter aimed at circumventing over-smoothing phenomena. In tackling the over-squashing issue, Topping et al. [31] employed the Stochastic Jost and Liu Curvature Rewiring algorithm to facilitate edge addition and removal during GNN training. Moreover, recent advancements have seen the utilization of state-of-the-art large language models in addressing graph-related tasks, exemplified by the Graphllm [39] and Graphgpt [40] models,

offering an alternative approach to mitigating both over-smoothing and over-squashing concerns.

2.3 Attention mechanism

Graph Attention Networks (GAT) [27] utilize self-attention mechanisms to compute the contribution weights of each neighboring node to update the central node's features, followed by a weighted summation to obtain new node features. GAT also uses multi-head attention, allowing the model to focus on the features of different neighbor nodes in parallel to better capture relationships and features in the graph data. HAN [41] introduces multiple layers of attention mechanisms, enabling the model to assign weights differently based on node types and relationships, facilitating better capturing of associations between nodes in heterogeneous graphs. In addition, other models such as GGAN-DGC [42], GA-GNN [43], SGATs [44] also incorporate attention mechanisms into their architectures, enhancing the model's capabilities. HGRN [28] incorporates multi-hop neighbor information to enhance the node representation ability of the original GAT, effectively addressing the over-smoothing problem. However, none of the previous research combined multi-head and multi-order information. Additionally, few studies focused on evaluating model performance at scarce label rates for semi-classification tasks.

Overall, the original GAT model may encounter the over-squashing issue in certain datasets. HGRN addresses this problem by incorporating high-order augmentation graphs. However, this approach might lead to an over-smoothing issue by introducing a large number of new edges. This concern forms the primary motivation for our work: we aim to combine multi-order and multi-head approaches to add heterogeneous paths for different-order augmented graphs, thereby solving both over-smoothing and over-squashing issues. In addition, considering the increase of the dimension of hyperparameter space, a more advanced hyperparameter optimization method should be investigated.

3 Preliminary

3.1 Graph attention networks(GAT)

A Graph Attention Network (GAT) is a type of graph neural network that uses an attention mechanism to capture the most important relationships between nodes in the graphs to enhance prediction or classification performances. The weights of edges can be calculated by the following formula:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}\vec{h}_i||\mathbf{W}\vec{h}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}\vec{h}_i||\mathbf{W}\vec{h}_k\right]\right)\right)}, \quad (1)$$

where α_{ij} is the weight between the i-th and j-th nodes, $\mathbf{W}$ is a trainable weight matrix for node feature transformation, $\vec{h}_i$ and $\vec{h}_j$ are the feature vectors of the i-th node and the j-th node, $\vec{\mathbf{a}}$ is a vector of weights, ^{T} represents transposition, $\mathcal{N}_i$ is the immediate neighborhood of the i-th node in the graph. The feature information is processed by combining with the weighted neighbor information from the attention adjacency matrix and then transformed through a nonlinear function, which is described by

$$\vec{h'}_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right), \quad (2)$$

where K is the number of multi-heads. σ denotes the final nonlinear function (usually a softmax). Finally, a prediction is to be made using the output of the final layer.

3.2 Mixed-order graph convolutional networks(MOGCN)

Mixed-order graph convolutional networks (MOGCN) [32] is a novel end-to-end ensemble framework. MOGCN combines the results of multiple GCN learners that are trained on adjacency matrices of various orders to boost the performance of semi-supervised node classification tasks. MOGCN can directly capture the dependencies between nodes at different distances, which enhances its information acquisition capability. Firstly, it constructs several similar GCN learners and assigns them adjacency matrices of different order in order to learn the information relationship between neighbors of different orders. Let's denote a finite set of multi-order adjacency matrices as $\left\{\mathbf{A}^1, \cdots, \mathbf{A}^k\right\}$. Then, we can formally define the GCN learners with respect to this set as follows,

$$\begin{aligned} f_1 &= \text{softmax}\left(\tilde{\mathbf{A}}_s^1\text{ReLU}\left(\tilde{\mathbf{A}}_s^1\mathbf{X}\mathbf{W}_0^1\right)\mathbf{W}_1^1\right), \\ f_2 &= \text{softmax}\left(\tilde{\mathbf{A}}_s^2\text{ReLU}\left(\tilde{\mathbf{A}}_s^2\mathbf{X}\mathbf{W}_0^2\right)\mathbf{W}_1^2\right), \\ &\vdots \\ f_k &= \text{softmax}\left(\tilde{\mathbf{A}}_s^k\text{ReLU}\left(\tilde{\mathbf{A}}_s^k\mathbf{X}\mathbf{W}_0^k\right)\mathbf{W}_1^k\right), \end{aligned} \quad (3)$$

where $\tilde{\mathbf{A}}_s^k = \tilde{\mathbf{D}}^{k-\frac{1}{2}}\tilde{\mathbf{A}}^k\tilde{\mathbf{D}}^{k-\frac{1}{2}}$, and $\tilde{\mathbf{A}}^k = \mathbf{A}^k + \mathbf{I}$, and the associated degree matrix is $\tilde{\mathbf{D}}^{(k)} = \mathbf{D}^{(k)} + \mathbf{I}$, $\mathbf{I}$ is a diagonal matrix, $\mathbf{X}$ is the node feature matrix, $\mathbf{W}_0^k$ and $\mathbf{W}_1^k$ are the trainable weight matrices in the k-th learner f_k.

For the obtained k learners $\{f_1, \cdots, f_k\}$ finally combined through an ensemble module [45]:

$$f_{ens}(\mathbf{X}) = \frac{1}{m}\sum_{k=1}^{m} f_k(\mathbf{X}) \quad (4)$$

In Eq. (4), the outputs of f_{ens} are the predicting label matrix based on the k-th learner.

Despite both GAT and MOGCN achieve state-of-the-art performance on various datasets, there still exist limitations. GAT can only transfer and aggregate information in the first-order neighborhood of the target node, and it cannot take into account nodes that are far away, thus ignoring the important information implied in them. While MOGCN can consider nodes in multi-order neighborhoods, it is unable to adaptively assign weights to individual neighbors, treating all nodes as having the same importance. Based on these problems, we propose a new model that can combine multi-order neighborhood information and adaptively assign weights to neighbor nodes.

4 The proposed method

In this section, we detail our method in two parts: the individual MHMOGAT layer and two distinct architectures. Our approach leverages a mix of multi-head attention and adjacency matrices of different orders, building upon the foundation established by HGRN [28]. HGRN incorporates multi-hop neighbor information to enhance node representation, addressing the over-smoothing problem. Extending upon this framework, our proposed model integrates multi-head and multi-order information, enhancing adaptability and efficiency across various tasks. Additionally, we employ Bayesian optimization to determine optimal hyperparameter combinations for different datasets. We propose two specific implementation methods through a reasonable combination of multi-head attention and adjacency matrices of different orders.

4.1 Attention layer

We start with the individual graph attention layer of MHMOGAT. The input to the attention layer of each graph is a set of node features, which we represent as a set of vectors:

$$\mathbf{h} = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_Z\right\}, \tag{5}$$

where $\vec{h}_i \in \mathbb{R}^F$, F is the number of features for each node, and Z is the number of nodes. The layer produces a new set of node features (of potentially different cardinality F'), $\mathbf{h}' = \left\{\vec{h'}_1, \vec{h'}_2, \ldots, \vec{h'}_Z\right\}$, $\vec{h'}_i \in \mathbb{R}^{F'}$, as its output. Let's denote a finite set of multi-order adjacency matrices as $\left\{\mathbf{A}^1, \cdots, \mathbf{A}^n\right\}$, where

$$\mathbf{A}^n = \mathbf{A}^{n-1}\mathbf{A}^1, \tag{6}$$

In order to convert the primary features into a more advanced feature representation and improve the capabilities of the model, we perform the following processing on the data:

$$e_{ij} = a\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right), \tag{7}$$

where $\mathbf{W}$ is a trainable weight matrix and a is a shared attention mechanism, e_{ij} represents the importance of the j-th node to the i-th node.

In the initial processing step, a linear transformation is applied on the nodes like original GAT model to learn the structures and patterns of data. Subsequently, the proposed model defines a novel self-attention mechanism that evaluates the influence of both immediate neighbors and long-distance nodes, while assigning heterogeneous multi-head attention to enhance feature extraction across different layers. This multi-level feature extraction facilitates the model's ability to handle complex features and manage long-distance dependencies effectively.

The modified attention coefficient is defined as follows:

$$\alpha_{ij}^k = \text{softmax}_j\left(e_{ij}^k\right) = \frac{\exp\left(e_{ij}^k\right)}{\sum_{m\in\mathcal{N}_i}\exp\left(e_{im}^k\right)}, \tag{8}$$

$$\alpha_{ij}^k = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}_k\vec{h}_i||\mathbf{W}_k\vec{h}_j^n\right]\right)\right)}{\sum_{m\in\mathcal{N}_i^n}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}_k\vec{h}_i||\mathbf{W}_k\vec{h}_m^n\right]\right)\right)}, \tag{9}$$

where α_{ij}^k represents the weight between i-th and j-th nodes in the k-th attention head, $\mathbf{W}_k$ is a trainable weight matrix, $\vec{h}_i$ and $\vec{h}_j^n$ represent the features of the i-th node and n-th order neighbor node j of i-th node, $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$ is a vector of weights, $\mathcal{N}_i$ is the n-th order neighborhood of the i-th node in the graph. After calculating the attention coefficient according to Eq.(9), it is used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node.

$$\vec{h'}_i = \sigma\left(\sum_{j\in\mathcal{N}_i^n}\alpha_{ij}^k\mathbf{W}_k\vec{h}_j^n\right), \tag{10}$$

where α_{ij}^k is the attention coefficient calculated by the k-th attention head, $\vec{h'}_i$ is the new node representation aggregated by multiple different order attention heads.

4.2 General framework

In this subsection, we describe the implementation details of the two distinct architectures of MHMOGAT: the multi-head

different orders attention architecture and the cross-order multi-head attention architecture.

4.2.1 MHMOGAT with multi-head different orders attention

In this architecture, in order to improve the performance capability of the model and to enable the model to learn the weights in neighbor relationships at different scales, we use adjacency matrices of different orders in each attention head for information aggregation.

Firstly, we assign different order adjacency matrices to each attention head based on Eq.(7)-(9) to obtain the k-th attention head as follows:(Figs. 1 and 2)

$$\alpha_{ij}^{1} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{T}\left[\mathbf{W}_1\vec{h}_i||\mathbf{W}_1\vec{h}_j^{1}\right]\right)\right)}{\sum_{m\in\mathcal{N}_i^{1}}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{T}\left[\mathbf{W}_1\vec{h}_i||\mathbf{W}_1\vec{h}_m^{1}\right]\right)\right)},$$
$$\alpha_{ij}^{2} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{T}\left[\mathbf{W}_2\vec{h}_i||\mathbf{W}_2\vec{h}_j^{2}\right]\right)\right)}{\sum_{m\in\mathcal{N}_i^{2}}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{T}\left[\mathbf{W}_2\vec{h}_i||\mathbf{W}_2\vec{h}_m^{2}\right]\right)\right)}, \tag{11}$$

$$\vdots$$

$$\alpha_{ij}^{k} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{T}\left[\mathbf{W}_k\vec{h}_i||\mathbf{W}_k\vec{h}_j^{k}\right]\right)\right)}{\sum_{m\in\mathcal{N}_i^{k}}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{T}\left[\mathbf{W}_k\vec{h}_i||\mathbf{W}_k\vec{h}_m^{k}\right]\right)\right)},$$

After obtaining the attention coefficients of different orders of each head, we design an information aggregation method as follows. The K attention heads execute the transformation of Eq.(10), and then their features are concatenated, resulting in the following output feature representation.

$$\vec{h'}_i = \Big\Vert_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i^{k}}\alpha_{ij}^{k}\mathbf{W}_k\vec{h}_j^{k}\right), \tag{12}$$

where $||$ represents concatenation, K is the number of attention heads, α_{ij}^{k} is the attention coefficient calculated by the k-th attention head, $\mathbf{W}_k$ is a trainable weight matrix, and $\vec{h'}_i$ is a new node representation aggregated by multiple different order attention heads.

Specifically, if we perform multi-head attention on the final layer of the network, for efficient implementation of multi-head attention, we replace concatenation with averaging and apply a nonlinear activation function to generate the

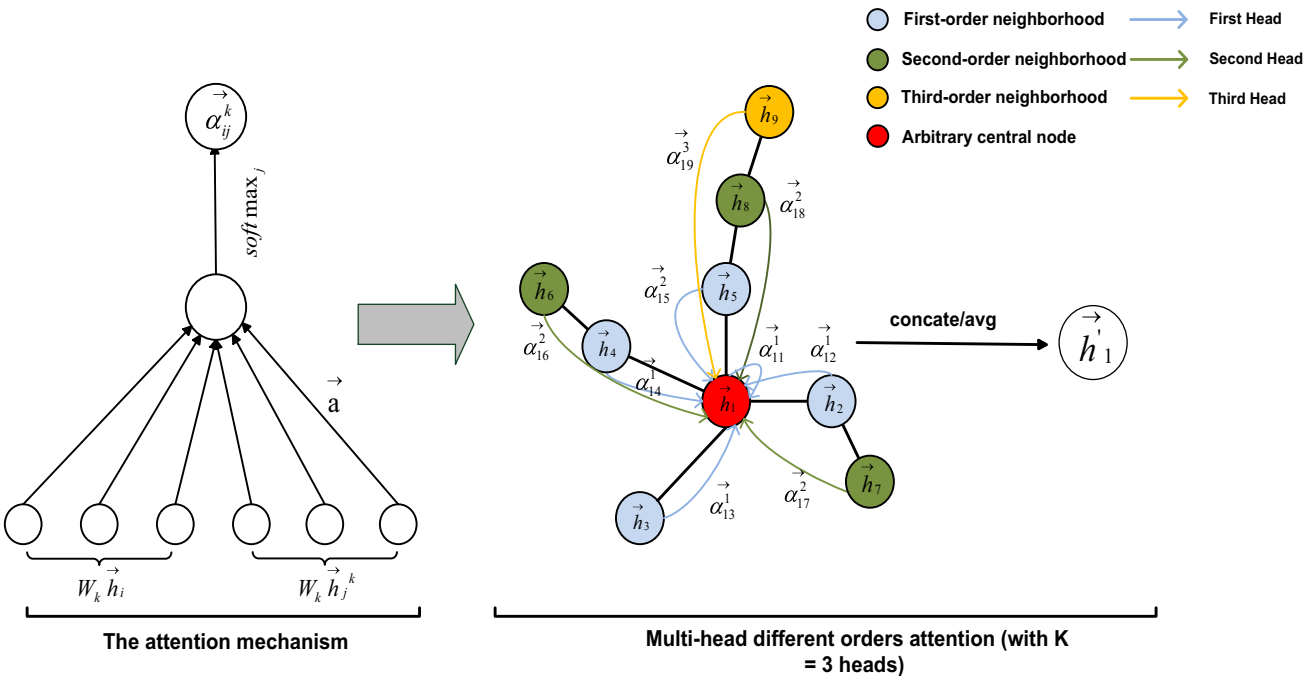


Fig. 1 The message aggregation in the Multi-Head Multi-Order Graph Attention Network (MHMOGAT) framework is illustrated. The red node represents an arbitrary central node, while the other nodes represent the neighbors of the central node at different orders. Initially, the attention coefficients are computed based on current node representations. This is followed by executing a multi-head different orders attention graph convolution on multi-hop nodes (where blue nodes represent 1-hop neighbors, green for 2-hop neighbors and yellow for 3-hop neighbors). The three differently colored arrows signify various levels of attention heads. Ultimately, to obtain $\vec{h'}_1$, the embedding from these heads are averaged

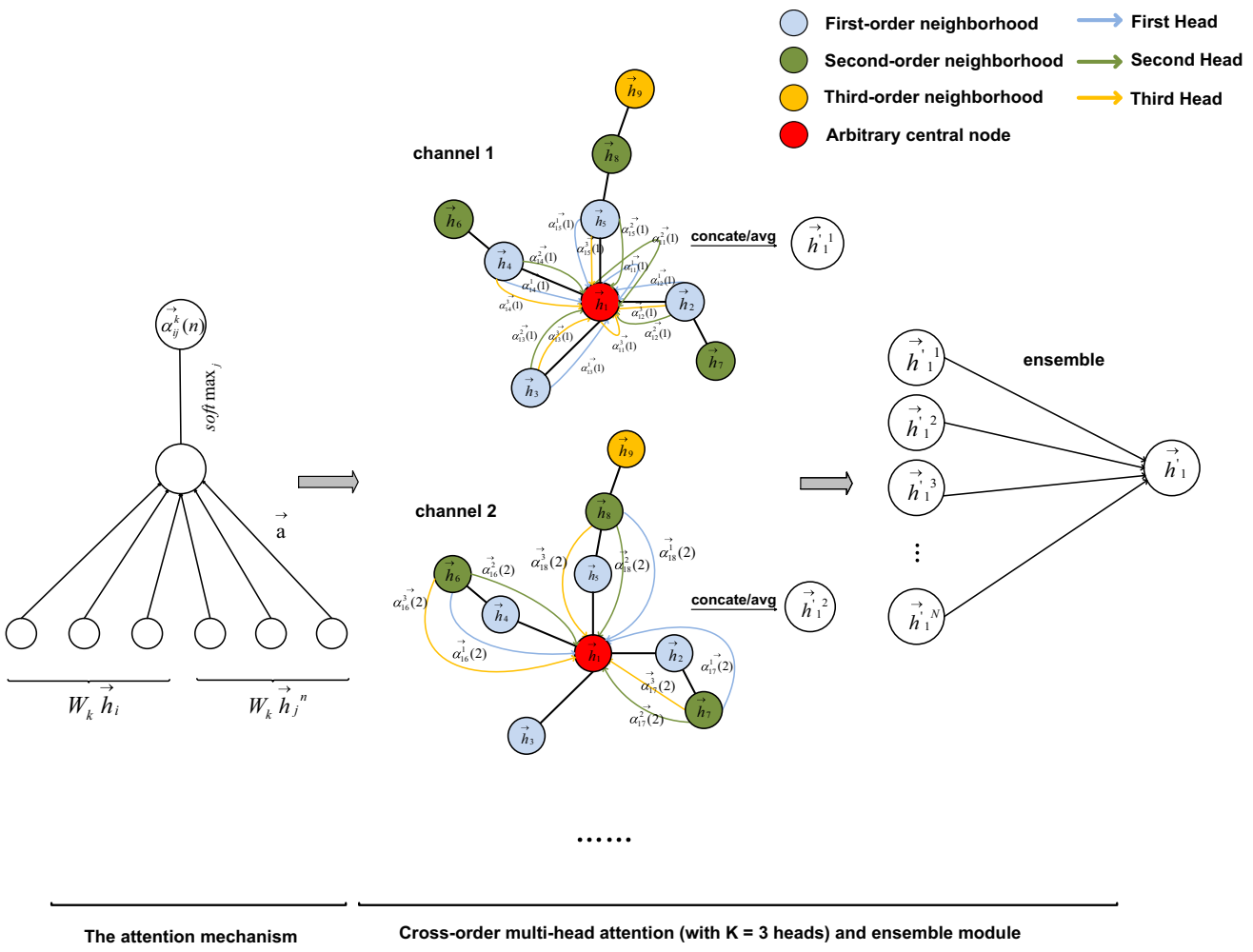


Fig. 2 The message aggregation in the Multi-Head Multi-Order Graph Attention Network (MHMOGAT) framework is illustrated. The red node represents an arbitrary central node, while the other nodes represent the neighbors of the central node at different orders. Initially, the attention coefficients are computed based on current node representations. This is followed by executing across-order multi-head attention graph convolution on multi-hop nodes. In channel 1, the current node executes across-order multi-head attention graph convolution on 1-hop neighbors; in channel 2, the node executes across-order multi-head attention graph convolution on 2-hop neighbors. The process continues until it reaches the predefined channel number N. Ultimately, the $\vec{h'}_i^1,...,\vec{h'}_i^n$ are averaged to generate the final node embedding

high-level node embedding.

$$\vec{h'}_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i^k}\alpha_{ij}^k\mathbf{W}_k\vec{h}_j^k\right). \tag{13}$$

Overall, the implementation of the MHMOGAT with multi-head different orders attention is described in Algorithm 1.

4.2.2 MHMOGAT with cross-order multi-head attention

In this section, we introduce another specific implementation method with cross-order multi-head attention.

Different from the first way, the MHMOGAT with cross-order multi-head attention introduces multiple attention heads across different orders. Therefore each attention head can aggregate information from different orders of adjacency matrix.

Firstly, several channels are established, each containing K attention heads. For the n-th channel, the n-th order adjacency matrix is applied to K attention heads following Eq. (7)-(9):

$$\alpha_{ij}^1(n) = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}_1\vec{h}_i||\mathbf{W}_1\vec{h}_j^n\right]\right)\right)}{\sum_{m\in\mathcal{N}_i^n}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}_1\vec{h}_i||\mathbf{W}_1\vec{h}_m^n\right]\right)\right)}, \tag{14}$$

Algorithm 1 : MHMOGAT with multi-head different orders attention.

Input: Node features $\mathbf{h}$, 1st-order adjacency matrix $\mathbf{A}^{(1)}$, number of attention heads K, and maximum number of iterations max_iter.
Output: Final prediction result of the unlabeled nodes with Eq.(13).
Construct higher-order adjacency matrices sets $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(k)}\}$ via Eq.(6) and initialize the weight vector $\vec{\mathbf{a}}$;
for $iter = 1 \rightarrow max\_iter$ **do**
  for $k = 1 \rightarrow K$ **do**
    Initialize the weight matrix $\mathbf{W}_k$;
    Calculate the graph attention layer of the k-th attention head α_{ij}^k via Eq.(11);
  end for
  if perform multi-head attention on the final (prediction) layer **then**
    Average the K results via Eq.(13);
  else
    Concatenate K results via Eq.(12);
  end if
  Calculate the cross-entropy loss function L;
  Train model with cross-entropy loss function L;
end for

$$\alpha_{ij}^k(n) = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}_k\vec{h}_i||\mathbf{W}_k\vec{h}_j^n\right]\right)\right)}{\sum_{m\in\mathcal{N}_i^n}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T\left[\mathbf{W}_k\vec{h}_i||\mathbf{W}_k\vec{h}_m^n\right]\right)\right)}. \tag{14}$$

where $\alpha_{ij}^k(n)$ is the attention coefficient calculated by the k-th attention head in the n-th channel. Eq.(14) allows for the calculation of the attention coefficient, as determined by K attention heads for each channel.

Subsequently, these K attention coefficients per channel are aggregated as follows:

$$\vec{h'}_i^n = \mathop{\Big\|}_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i^n}\alpha_{ij}^k(n)\mathbf{W}_k\vec{h}_j^n\right), \tag{15}$$

Specially, when multi-head attention is implemented on the final layer of the network, the tensor has been aggregated along the n-th channel, which can be expressed as,

$$\vec{h'}_i^n = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i^n}\alpha_{ij}^k(n)\mathbf{W}_k\vec{h}_j^n\right), \tag{16}$$

where $\vec{h'}_i^n$ is the new node representation formed by cross-order aggregation of K attention heads in the n-th channel.

Finally, we integrate the results obtained by N channels through an integration module, which is defined as follows:

$$\vec{h'}_i = \frac{1}{N}\sum_{n=1}^{N}\vec{h'}_i^n. \tag{17}$$

Algorithm 2 : MHMOGAT with cross-order multi-head attention.

Input: Node features $\mathbf{h}$, 1st-order adjacency matrix $\mathbf{A}^{(1)}$, number of orders and channels N, number of attention heads K, and maximum number of iterations max_iter.
Output: Final prediction result of the unlabeled nodes with Eq.(17).
Construct higher-order adjacency matrices sets $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(n)}\}$ via Eq.(6), and initialize weight vector $\vec{\mathbf{a}}$;
for $k = 1 \rightarrow K$ **do**
  Initialize the weight matrix $\mathbf{W}_k$;
end for
for $iter = 1 \rightarrow max\_iter$ **do**
  for $n = 1 \rightarrow N$ **do**
    for $k = 1 \rightarrow K$ **do**
      Calculate the graph attention layer of the k-th attention head α_{ij}^k via Eq.(14);
    end for
    if perform multi-head attention on the final (prediction) layer **then**
      Average the K results via Eq.(16);
    else
      Concatenate K results via Eq.(15);
    end if
  end for
  Integrate the results obtained by N channels via Eq.(17);
  Calculate the cross-entropy loss function L;
  Train model with cross-entropy loss function L.
end for

Overall, this approach incorporates multiple attention heads across various orders, enabling each head to learn and balance information propagation at different scales: from first-order neighbors to higher-order ones. Consequently, each attention head can identify useful features within different orders of information and subsequently integrate these features. The implementation details of this method are shown in Algorithm 2.

5 Experiments

In this section, the proposed model is compared with several prevailing GCNs across six general semi-supervised datasets.

5.1 Datasets

Table 1 presents the datasets used in the experiment, namely Cora, Citeseer, Pubmed, ACM, Chameleon and Actor. Cora, Citeseer and Pubmed are three citation networks, the nodes represent publications, the edges represent citation links, the feature of each node is the bag-of-word representation of the corresponding publication, and the class is the number of clusters. ACM is a paper network, and the nodes represent papers. If two papers are written by the same author, then there is an edge between the two papers. The feature is the bag-of-words representation of keywords. Papers are divided

Table 1 The statistics of the datasets

Datasets	Nodes	Edges	Features	Classes	labeled per class	Label Rate
Cora	2708	5429	1433	7	2/4/8/12/16/20	0.5%/1%/2%/3%/4%/5%
Citeseer	3327	4732	3703	6	3/6/12/18/24/30	0.5%/1%/2%/3%/4%/5%
Pubmed	19717	44338	500	3	2/3/7	0.03%/0.05%/0.1%
ACM	3025	13128	1870	3	1/2/3/4/5	0.1%/0.2%/0.3%/0.4%/0.5%
Chameleon	2277	36101	2325	5	2/23/46/69	0.5%/5%/10%/15%
Actor	7600	33544	931	5	8/15/30/45	0.5%/1%/2%/3%

into three categories according to research fields. Chameleon is a web page dataset where nodes are web pages on a specific topic and edges are hyperlinks between them. Actor is a dataset where nodes are actors and edges mean two actors co-occurring on the same Wikipedia page.

5.2 Comparison with state-of-the-art methods

5.2.1 Baseline methods

To evaluate the performance of the proposed MHMOGAT, we compare it with the following methods:

1. GCN [18]: GCN is a graph convolutional network used to learn the representation of nodes in graph data. It updates the representation of a node by aggregating information from each node's neighbor nodes.

2. GAT [27]: GAT is an improved graph convolutional network that introduces an attention mechanism to dynamically adjust the weights of neighbor nodes. GAT allows nodes to assign different weights to their neighbor nodes, which makes the model more flexible to learn the relationship between nodes.

3. MOGCN [32]: a graph convolutional neural network improved on the basis of GCN. It is finally integrated by applying a hybrid high-order adjacency matrix to multiple GCN modules.

4. GCNII [46]: a simple and deep graph convolutional network model, which effectively solves the problem of over-smoothing through initial residuals and unit mappings.

5. KGCN [47]: KGCN is a graph convolutional network specifically designed for knowledge graphs. KGCN is used to learn embeddings between entities and relationships to support various reasoning and prediction tasks on knowledge graphs.

6. SGC [48]: SGC is a simplified graph convolutional network that reduces the complexity of the model by subjecting the features of all neighbor nodes to the same linear transformation.

7. GFNN [49]: a simple and deep graph convolutional network model, which effectively solves the problem of over-smoothing through initial residuals and unit mappings.

8. SJLR [31]: SJLR introduces the Stochastic Jost and Liu Curvature Rewiring (SJLR) algorithm, which adjusts edges during training to effectively solve over-smoothing and over-squeezing issues while maintaining efficiency and reliability.

9. DropEdge (DE) [29]: DE is an operation-based approach that developed a method for dropping edges to overcome the over-smoothing issue in deep GCNs.

10. GRAND [50]: GRAND approaches deep learning on graphs as a continuous diffusion process and treats Graph Neural Networks (GNNs) as discretizations of an underlying PDE.

5.2.2 Parameter settings

All experiments are conducted in a semi-supervised setting. The maximum iteration number (max_iter) is set to 500, and the method utilizes a hidden layer with 8 neurons. We train our model by using the full batch in each training epoch and implement our algorithm in Pytorch [51], and we optimize it with the Adam [52] algorithm. Additionally, learning rate and dropout rate are set at 0.005 and 0.6 respectively. Following [53, 54], we conduct experiments on the following label rates: 0.5%, 1%, 2%, 3%, 4% and 5% on Citeseer and Cora datasets, and 0.03%, 0.05%, and 0.1% on the Pubmed dataset, and 0.5%, 1%, 2% and 3% on Actor dataset. Furthermore, we set the label rates to 0.1%, 0.2%, 0.3%, 0.4% and 0.5% on the ACM dataset and 0.5%, 5%, 10% and 15% on the Chameleon dataset and then conduct experiments. The mean classification accuracy (ACC) on test nodes is reported after running our method over specified data splits twenty times. The number of attention heads is set to 8. In MHMOGAT based on different orders of multi-head attention, the number of different order n is selected among $\{1, 2, 3\}$. In MHMOGAT based on cross-order multi-head attention, the number of different order n is selected among $\{1, 2, ..., 6\}$.

5.2.3 Results

The semi-supervised node classification results of the baseline methods and two variations of MHMOGAT under optimal hyperparameters n are reported in Tables 2, 3, 4 and 5.

Table 2 Comparison results of the proposed MHMOGAT-1 and MHMOGAT-2 algorithms with several state-of-the-art methods(%)

Dataset	Metrics	Label Rate	GAT	MOGCN	GCN	DE	GCNII	KGCN	SGC	GFNN	SJLR	GRAND	MHMO GAT-1	MHMO GAT-2
Cora	ACC	0.5%	57.8	54.0	48.2	51.2	50.6	54.1	52.0	37.3	56.8	56.6	**61.4**	**60.0**
		1%	65.9	63.0	59.4	60.5	56.7	64.5	61.2	44.3	64.8	66.1	**68.5**	**68.1**
		2%	73.1	71.9	70.5	71.1	68.4	73.0	69.4	52.1	72.7	**74.3**	73.6	**74.2**
		3%	76.9	74.9	74.4	74.7	73.1	77.3	72.6	56.0	76.4	76.8	**78.0**	**78.0**
		4%	78.3	77.1	76.5	76.9	72.3	78.0	73.9	57.7	78.4	78.7	**78.8**	**79.0**
		5%	79.9	79.5	78.4	78.9	74.3	79.9	75.3	59.0	**80.8**	**81.1**	80.4	80.6
	F1	0.5%	55.8	52.6	47.8	49.8	47.1	50.9	49.3	35.2	54.3	51.9	**60.1**	**58.4**
		1%	65.4	62.5	59.2	60.3	56.2	63.6	59.9	43.4	64.1	65.0	**67.0**	**67.6**
		2%	72.1	71.0	69.6	70.1	67.6	71.7	68.1	51.0	72.0	**73.0**	72.6	**73.1**
		3%	75.6	73.9	73.2	73.7	71.6	75.9	71.2	54.8	75.7	76.0	**76.8**	**76.4**
		4%	77.2	75.9	75.4	75.5	71.1	76.6	72.7	56.5	**77.5**	77.3	**77.5**	**77.9**
		5%	78.5	78.1	77.0	77.5	72.4	78.7	74.1	57.8	**79.3**	**79.9**	78.9	79.2
Citeseer	ACC	0.5%	49.5	45.5	41.4	45.7	38.2	45.1	44.2	32.9	45.3	**54.4**	50.6	**51.3**
		1%	58,8	57.9	55.4	58.1	53.8	56.9	57.0	41.2	58.3	**61.3**	58.1	**60.0**
		2%	64.8	64.2	63.1	64.3	61.0	64.0	63.8	46.8	64.3	**69.7**	**65.8**	**65.8**
		3%	66.9	66.3	65.6	66.5	62.7	66.2	66.6	48.2	66.4	**70.8**	67.2	**67.5**
		4%	68.7	68.3	67.1	68.4	64.7	68.7	68.1	52.0	68.2	**72.5**	**69.3**	69.0
		5%	69.1	68.8	68.2	69.0	66.3	69.1	68.8	51.7	68.8	**73.0**	**69.6**	69.3
	F1	0.5%	43.3	38.4	35.1	38.7	31.3	37.9	38.0	25.5	37.5	**46.9**	43.3	**44.8**
		1%	54.0	53.1	51.2	52.1	48.9	52.3	52.8	37.1	51.6	**55.1**	53.0	**55.2**
		2%	61.1	60.8	59.9	61.0	57.1	60.5	60.0	44.0	60.3	**64.1**	61.8	**62.0**
		3%	63.3	62.7	62.5	63.0	59.0	63.0	62.5	45.8	62.4	**65.8**	63.4	**63.8**
		4%	65.2	64.5	63.9	64.9	60.8	65.3	64.9	50.0	64.6	**67.0**	**65.6**	65.4
		5%	66.5	66.2	65.6	65.8	66.3	65.7	65.6	50.1	65.3	**67.2**	**67.2**	66.3
Pubmed	ACC	0.03%	61.6	60.9	58.4	59.4	54.3	54.4	56.2	46.5	61.2	54.3	**65.3**	**64.3**
		0.05%	66.4	65.2	61.5	63.6	62.6	62.1	63.8	57.3	66.8	59.8	**68.7**	**69.1**
		0.1%	71.0	70.4	63.6	65.7	66.0	65.8	67.2	62.1	**71.4**	66.0	**72.8**	71.2
	F1	0.03%	58.6	57.5	54.9	56.1	51.2	50.7	53.1	42.5	57.8	50.9	**62.7**	**61.3**
		0.05%	65.2	64.4	60.3	62.4	61.8	61.1	62.1	56.2	65.3	57.7	**67.5**	**68.2**
		0.1%	70.4	69.5	62.9	65.1	65.2	64.9	66.7	61.7	70.2	64.9	**71.1**	**70.5**

Table 2 continued

Dataset	Metrics	Label Rate	GAT	MOGCN	GCN	DE	GCNII	KGCN	SGC	GFNN	SJLR	GRAND	MHMO GAT-1	MHMO GAT-2
ACM	ACC	0.1%	50.8	55.2	55.1	53.1	46.6	54.6	52.7	52.9	52.4	–	**61.1**	**59.8**
		0.2%	68.5	65.8	64.0	65.0	50.9	65.0	60.6	60.1	64.5	–	**70.5**	**72.5**
		0.3%	74.4	72.4	71.9	72.6	54.7	71.0	67.4	62.8	73.3	–	**77.3**	**76.4**
		0.4%	**79.8**	78.1	77.2	78.7	57.3	77.9	74.4	69.9	78.8	–	79.7	**83.9**
		0.5%	**82.7**	80.6	79.4	80.6	60.7	79.7	78.8	73.3	81.0	–	81.6	**84.1**
	F1	0.1%	42.6	48.3	48.4	46.1	37.6	47.7	44.7	49.5	45.2	–	**55.4**	**54.3**
		0.2%	65.1	61.8	60.2	61.0	43.4	62.0	55.9	57.2	60.8	–	**67.9**	**70.7**
		0.3%	73.7	70.9	69.8	70.1	48.7	69.6	65.5	61.4	72.5	–	**76.1**	**74.9**
		0.4%	79.0	77.5	76.8	77.3	51.9	77.6	73.5	69.1	78.1	–	**79.3**	**83.8**
		0.5%	**82.4**	80.3	79.2	80.4	56.1	79.3	78.3	73.0	80.7	–	81.0	**83.9**
Chameleon	ACC	0.5%	**33.3**	32.5	33.0	30.4	26.9	32.4	26.9	31.4	30.6	27.5	31.3	**33.6**
		5%	49.4	47.1	**50.5**	41.8	27.1	**50.7**	32.9	45.0	41.8	38.3	45.7	49.1
		10%	**54.6**	51.0	54.2	46.7	29.2	53.9	34.8	48.1	46.4	39.5	49.6	**54.9**
		15%	**55.3**	52.7	55.2	48.4	32.6	54.9	35.5	49.6	48.0	41.5	51.9	**55.9**
	F1	0.5%	**31.6**	30.9	31.2	28.2	20.5	30.8	24.5	29.7	28.1	21.0	29.6	**31.7**
		5%	49.0	46.3	**50.1**	41.3	20.4	**50.6**	32.0	44.8	41.0	36.8	44.1	48.2
		10%	**54.1**	50.3	53.5	46.7	23.6	53.7	34.2	47.8	46.1	38.0	48.0	**54.3**
		15%	**55.1**	52.2	**55.1**	48.1	27.4	54.2	34.7	49.4	47.6	40.3	50.7	**55.4**
Actor	ACC	0.5%	19.8	19.4	19,8	19.9	15.6	19.4	19.9	19.2	20.1	**21.0**	**21.1**	20.5
		1%	19.7	19.9	20.3	20.1	15.7	19.8	20.5	20.4	20.6	**22.2**	**21.6**	20.9
		2%	20.0	20.6	21.9	21.1	20.5	20.6	21.2	20.5	21.0	**24.0**	**22.0**	21.4
		3%	20.4	20.9	22.3	22.0	21.6	20.9	21.3	21.4	21.8	**25.2**	**22.5**	21.4
	F1	0.5%	18.6	18.0	19.8	18.7	15.0	18.7	19.6	18.7	19.4	**20.5**	**20.7**	19.7
		1%	19.0	18.8	19.8	19.5	15.2	19.4	20.2	19.7	19.9	**21.4**	**21.2**	20.0
		2%	19.5	20.5	21.2	20.4	20.0	20.0	20.7	20.0	20.6	**23.5**	**21.6**	20.5
		3%	20.3	20.7	**22.1**	21.6	21.2	20.7	21.2	21.2	20.8	**24.5**	**22.1**	20.6

Note: The best and second best performing methods for each dataset at different label rates are shown in **red** and **blue**, respectively

Table 3 Experimental results of the order-number

Method	Cora						Citeseer						Pubmed		
	0.5%	1%	2%	3%	4%	5%	0.5%	1%	2%	3%	4%	5%	0.03%	0.05%	0.1%
2nd-order	58.7	67.3	73.1	77.6	78.8	80.1	49.6	57.7	65.0	66.8	68.7	69.4	62.9	67.2	70.7
3rd-order	61.1	67.5	72.8	77.3	78.4	79.5	50.3	57.8	65.2	67.0	68.2	69.0	63.1	67.5	70.1
4th-order	60.2	67.2	72.2	77.1	78.0	79.4	50.1	57.5	65.0	66.7	68.0	68.8	–	–	–
5th-order	59.8	66.8	71.5	76.8	77.5	79.1	49.7	57.1	64.8	66.3	67.8	68.6	–	–	–
6th-order	59.4	66.4	70.1	76.2	77.1	78.9	49.3	56.8	64.3	65.9	67.4	68.1	–	–	–

Method	ACM					Chameleon				Actor			
	0.1%	0.2%	0.3%	0.4%	0.5%	0.5%	5%	10%	15%	0.5%	1%	2%	3%
2nd-order	55.5	65.5	74.3	76.2	78.8	28.9	43.8	48.0	50.3	19.2	19.1	19.2	19.5
3rd-order	50.0	65.0	75.7	77.7	80.3	28.3	39.8	45.2	46.2	20.5	21.1	21.3	21.1
4th-order	50.0	63.2	71.0	75.3	76.7	28.2	38.8	43.5	44.4	21.8	21.0	22.9	21.9
5th-order	46.8	63.9	66.5	65.8	76.6	28.1	38.5	40.6	44.0	20.4	20.2	22.4	21.3
6th-order	49.6	59.7	63.4	68.8	75.1	27.6	38.2	40.7	42.6	19.7	20.2	21.7	21.9

Note: In some cases it cannot be run on some datasets due to memory limitations, as indicated by "-"

Specifically, MHMOGAT-1 indicates MHMOGAT with cross-order multi-head attention and MHMOGAT-2 indicates MHMOGAT with multi-head different orders attention. We obtain the following observations:

In comparison to all the baseline methods, MHMOGAT-1 and MHMOGAT-2 achieve superior performance across various label rates on the six datasets. The GRAND model outperforms our model at most label rates on Citeseer and Actor datasets, but underperforms on Cora, Pubmed and Chameleon datasets. Furthermore, our model's performance is almost in the top two at each label rate across all datasets. The GCN model shows low accuracy because it only employs a uniform neighbor aggregation strategy without an attention mechanism. Although GAT incorporates an attention mechanism, its performance is limited as it only captures information from first-order neighbors and is thus potentially missing important information from distant nodes. In contrast, through unique combinations of multi-head attention and varying order adjacency matrices, our models exhibit excellent node classification capabilities.

Table 4 Adjacency matrix combination results

Order	adj	adj2	adj3
Combination 1	4	4	0
Combination 2	4	0	4
Combination 3	0	4	4
Combination 4	3	3	2
Combination 5	3	2	3
Combination 6	2	3	3

The proposed method shows excellent performance even at low label rates. For example, at a label rate of 0.5%, the classification accuracy of GCN and GCNII on the Cora dataset is only about 48% and 50%, but the accuracy of MHMOGAT-1 reaches 61%. The accuracy of MHMOGAT-2 is approximately 60%, exceeding the performance of other methods. Despite the extremely low label rate of 0.03% on the Pubmed dataset, both MHMOGAT-1 and MHMOGAT-2 still achieve classification accuracy of 65% and 64%, respectively, surpassing other methods.

Figure 3 illustrates the impact of high-order GATs and MHMO on a graph with bottleneck issues. Initially, there were only two paths between the left and right clusters, leading to an over-squashing problem. High-order GAT [28] introduces high-order information, which increases paths between clusters homogeneously and effectively curbs the over-squashing issue. In contrast, the proposed MHMO framework adds heterogeneous paths (2 attention heads for 1^{st} order and 1 attention head for 2^{nd} order) and designs a novel attention mechanism to assign weights to each path, thereby enhancing the model's expressiveness and reducing potential over-smoothing issues.

5.3 Ablation study

To understand the impacts of the components of MHMOGAT on its performance, we conduct an ablation study in this section. Here, we consider two variants of our model:

- **(1) MHMOGAT-1-SO**: MHMOGAT-1 uses a single order adjacency matrix ({1, 2, ..., 6}) for training, instead

Table 5 Experimental results of the combinations

Method	Cora						Citeseer						Pubmed		
	0.5%	1%	2%	3%	4%	5%	0.5%	1%	2%	3%	4%	5%	0.03%	0.05%	0.1%
Combination 1	58.3	67.5	73.8	77.6	78.7	80.2	50.7	59.7	65.1	67.5	69.0	69.3	62.7	67.0	71.5
Combination 2	59.4	67.7	73.5	77.3	78.3	79.6	51.3	60.0	65.8	67.0	68.6	69.0	63.1	66.8	71.8
Combination 3	59.0	65.1	69.9	73.1	73.7	78.3	50.5	58.8	64.3	66.3	67.8	68.4	62.5	65.7	70.2
Combination 4	59.1	66.3	73.1	76.2	77.0	78.9	50.0	58.8	65.3	67.0	68.7	69.0	61.9	65.6	71.2
Combination 5	58.7	65.2	70.1	72.7	73.6	78.0	50.6	58.1	64.4	66.2	67.9	68.6	61.3	65.3	70.8
Combination 6	58.8	65.1	69.7	73.0	73.8	78.6	50.6	58.2	64.4	66.2	67.9	68.8	61.6	65.5	70.6

Method	ACM					Chameleon				Actor			
	0.1%	0.2%	0.3%	0.4%	0.5%	0.5%	5%	10%	15%	0.5%	1%	2%	3%
Combination 1	59.2	71.1	76.4	83.9	84.0	33.6	49.1	54.9	55.2	20.5	20.1	20.6	21.2
Combination 2	54.4	72.5	75.4	81.4	84.1	31.8	46.7	51.6	55.9	20.2	20.6	20.1	21.1
Combination 3	48.4	55.9	61.0	62.4	73.3	23.8	28.6	31.4	32.9	19.4	20.2	20.9	21.3
Combination 4	51.4	60.8	71.3	70.4	81.3	23.0	32.9	37.6	37.8	19.3	19.0	19.8	21.1
Combination 5	49.4	54.6	68.0	64.1	77.8	24.5	30.0	33.4	35.0	19.8	20.9	21.4	21.4
Combination 6	47.7	59.4	64.1	63.1	79.6	22.4	31.7	35.7	36.5	19.6	20.4	20.7	21.1

of different order matrices. The best result among the single order setups is reported.

- **(2) MHMOGAT-2-SO**: MHMOGAT-2 uses a single order adjacency matrix ({1, 2, ..., 6}) for training, instead of different order matrices. The best result among the single order setups is reported.

Figure 4 illustrates the comparison results of MHMOGAT and its variants. MHMOGAT-1-SO and MHMOGAT-2-SO all perform worse than the proposed MHMOGAT-1 and MHMOGAT-2 models on the six datasets. This is because the two single-order variants cannot capture the rich relation information in the graph, while MHMOGAT can capture multi-order neighbor information using multi-order adjacency matrices and thus achieve significant performance. In addition, by utilizing multi-order adjacency matrices, hierarchical feature representation of nodes is achieved, which adapts to more complex graph structures and integrates information at multiple levels, thereby enhancing the model's performance.

5.4 Parameter sensitivity

In MHMOGAT, the hyperparameter n, which determines the number of adjacency matrices of different orders, can influence the model performance significantly. Therefore, we conducted an analysis to examine its sensitivity.

The 2nd to the 6th high-order matrices are considered in MHMOGAT-1 and the results are presented in Table 3. We found that MHMOGAT-1 with 2nd and 3rd-order matrices provides comparably satisfactory performance on Cora, Citeseer, Pubmed, ACM and Chameleon. This means that it is suitable to incorporate low-order relations to enhance the

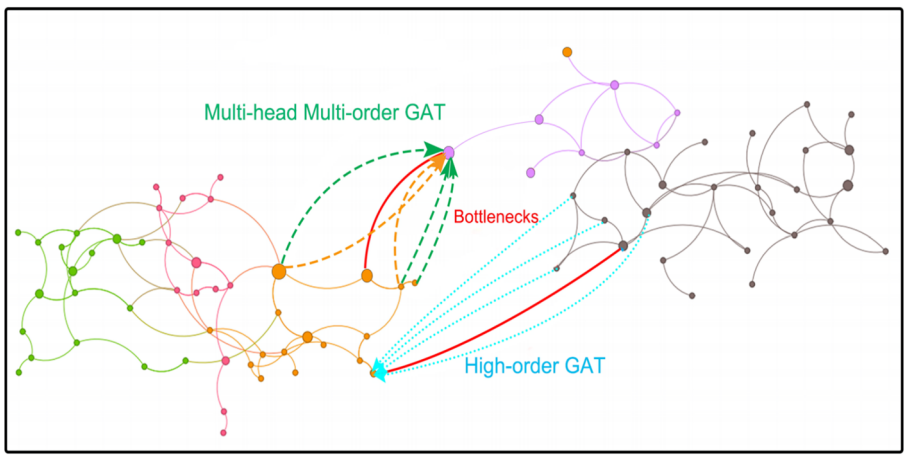
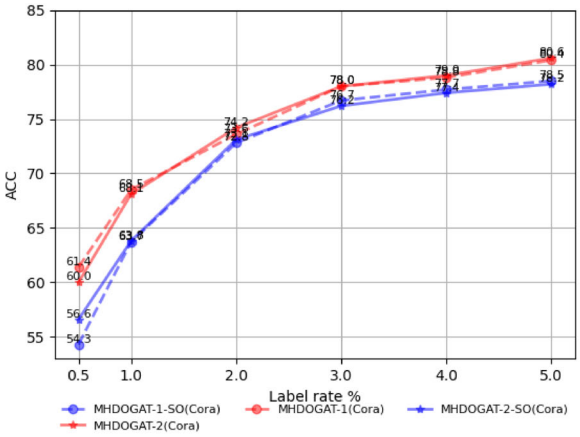
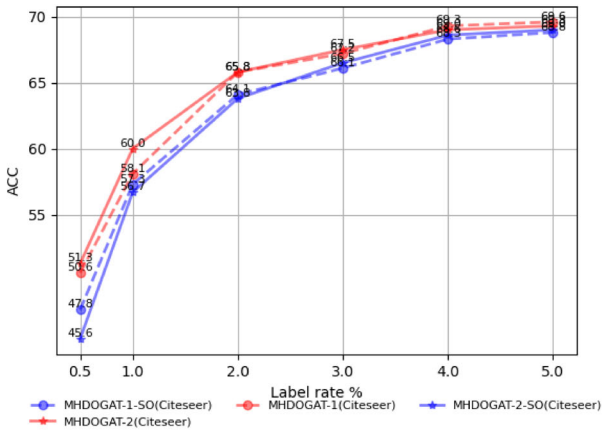


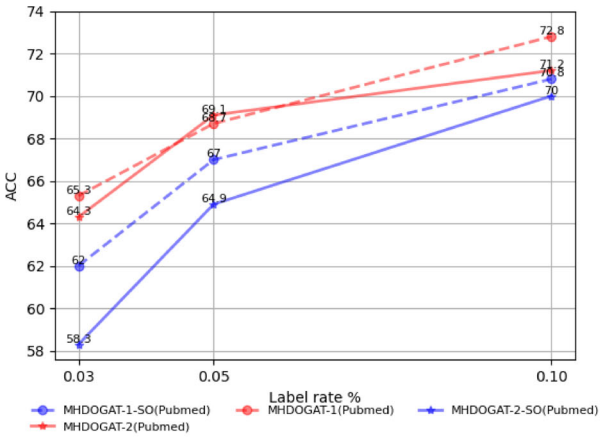
Fig. 3 Integrating Multi-head, Multi-order relations to overcome over-smoothing and over-squashing Issues. The two red lines represent the original paths connecting the two clusters



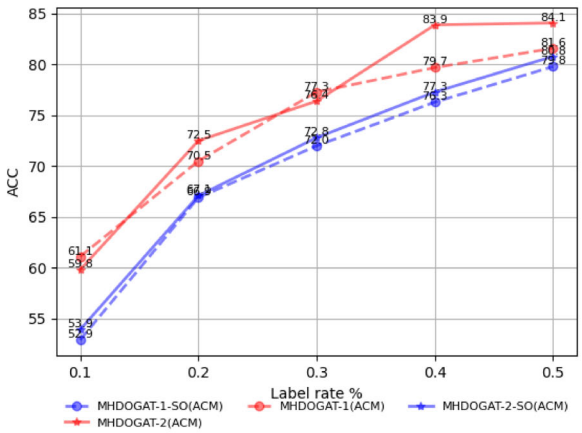
(a) Cora dateset



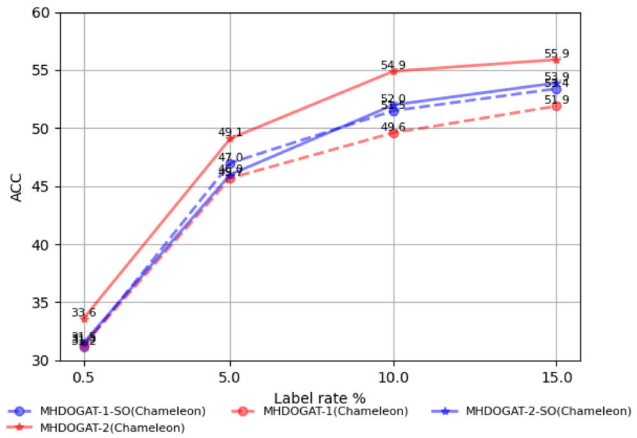
(b) Citeseer dateset



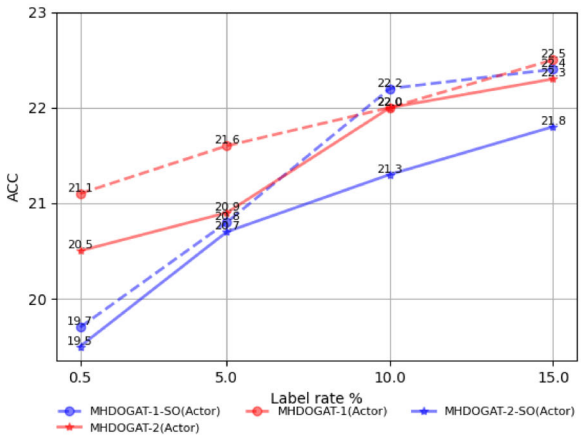
(c) Pubmed dateset



(d) ACM dateset



(e) Chameleon dateset



(f) Actor dateset

Fig. 4 Illustration of the performances of variants of MHMOGAT at various label rates

effectiveness of GAT on these datasets. In contrast, on the Actors dataset, MHMOGAT-1 with 3rd-order and 4th-order performs better. This indicates that high-order information is crucial on the Actors dataset. However, it is found that further increasing the order of matrices results in the decline of classification performance.

In MHMOGAT-2, the number of different orders, denoted as n, is selected from the set $\{1, 2, 3\}$. We combine these three adjacency matrices and then place the resulting eight combinations into the eight attention heads sequentially. The combined results are shown in Table 4.

Specifically, 'adj', 'adj2', and 'adj3' indicate the first-order, second-order, and third-order adjacency matrices, respectively. For example, combination 1 consists of 4 first-order adjacency matrices and 4 second-order adjacency matrices. Then, we conduct experiments on six datasets, and the results are as shown in Table 5.

The performance of MHMOGAT-2 is more complex, which is related to the combination of two hyperparameters as shown in Table 4. It is found that MHMOGAT-2 with hyperparameters of combination 1 and combination 2 performs well on the Cora, Citeseer, Pubmed, ACM, and Chameleon datasets. This is because under these settings, the model can effectively learn at various scales and capture relationships within the graph more accurately. On the Actors dataset, using hyperparameters of combination 5 performs better because the nodes in the Actors dataset are more sensitive to high-order information.

6 Conclusions and future work

In the realm of Graph Neural Networks (GNNs), significant limitations such as over-smoothing and over-squashing have been observed. To address these deficiencies, this work introduces the Multi-Head Multi-Order Graph Attention Network (MHMOGAT) architecture. By leveraging multi-head attention mechanisms, over-smoothing is mitigated, while the incorporation of multi-order adjacency matrices tackles over-squashing arising from bottlenecks in the graph data. Furthermore, Bayesian optimization is employed to identify optimal hyperparameter combinations tailored to different datasets. The proposed MHMOGAT framework offers two distinct implementations, each combining multi-head attention with adjacency matrices of varying orders. Through extensive experimentation across six datasets, our approach demonstrates state-of-the-art performance and shows effectiveness in handling large and complex graphs with low label rates.

In future studies we plan to improve this work in three ways: 1) validate our model using more diverse datasets; 2) compare it against state-of-the-art deep GCN models; and finally 3) test the effectiveness of our model when applied to regression problems.

Acknowledgements This work is supported by National Natural Science Foundation of China (Nos. 62076111, 62006099).

Author Contributions Jie Ben: Investigation,Resources,DataCuration, Writing - Review& Editing. Qiguo Sun: Conceptualization, Methodology, Formal analysis. Keyu Liu: Investigation,Writing - Original Draft,Writing - Review& Editing. Xibei Yang: Investigation, Project administration, Resources. Fengjun Zhang: Data Curation.

Funding This work is supported by National Natural Science Foundation of China (Nos. 62076111, 62006099).

Data Availability The authors do not have permission to share data.

Declarations

Conflicts of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical and informed consent The data used in this study were sourced from public databases, which are openly accessible and do not contain any personally identifiable information. Therefore, the issue of informed consent does not arise. All data handling procedures adhered to ethical guidelines for research.

References

1. Kang Z, Peng C, Cheng Q, Liu X, Peng X, Xu Z, Tian L (2021) Structured graph learning for clustering and semi-supervised classification. Pattern Recognit 110:107627
2. Maltseva D, Batagelj V (2021) Journals publishing social network analysis. Scientometrics 126(4):3593–3620
3. Sun Q, Wei X, Yang X (2024) Graphsage with deep reinforcement learning for financial portfolio optimization. Expert Syst Appl 238:122027122027
4. Wang X, Yang X, Wang P, Yu H, Xu T (2023) Ssgcn: a sampling sequential guided graph convolutional network. Int J Mach Learn Cybern 1–16
5. Guo Q, Yang X, Zhang F, Xu T (2024) Perturbation-augmented graph convolutional networks: A graph contrastive learning architecture for effective node classification tasks. Eng Appl Artif Intell 129:107616
6. Sun Q, Wei X, Yang X (2024) Graphsage with deep reinforcement learning for financial portfolio optimization. Expert Syst Appl 238:122027
7. Zhou Z-H, Zhan D-C, Yang Q (2007) Semi-supervised learning with very few labeled training examples. In: AAAI, vol 7, pp 675–680
8. Li Y, Yin J, Chen L (2023) Informative pseudo-labeling for graph neural networks with few labels. Data Min Knowl Discov 37(1):228–254
9. Van Engelen JE, Hoos HH (2020) A survey on semi-supervised learning. Mach Learn 109(2):373–440

10. Gao C, Zhou J, Miao D, Wen J, Yue X (2021) Three-way decision with co-training for partially labeled data. Inf Sci 544:500–518
11. Kim D, Seo D, Cho S, Kang P (2019) Multi-co-training for document classification using various document representations: Tf-idf, lda, and doc2vec. Inf Sci 477:15–29
12. Jordan MI, Mitchell TM (2015) Machine learning: Trends, perspectives, and prospects. Science 349(6245):255–260
13. Cozman FG, Cohen I, Cirelo M (2002) Unlabeled data can degrade classification performance of generative classifiers. In: Flairs conference, pp 327–331
14. Kingma DP, Mohamed S, Jimenez Rezende D, Welling M (2014) Semi-supervised learning with deep generative models. Adv Neural Inf Process Syst 27
15. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. Commun ACM 63(11):139–144
16. Dong H, Yang L, Wang X (2021) Robust semi-supervised support vector machines with laplace kernel-induced correntropy loss functions. Appl Intell 51:819–833
17. Calma A, Reitmaier T, Sick B (2018) Semi-supervised active learning for support vector machines: A novel approach that exploits structure information in data. Inf Sci 456:13–33
18. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
19. Saluja A, Awadalla HH, Toutanova K, Quirk C (2014) Graph-based semi-supervised learning of translation models from monolingual data. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp 676–686
20. Wang J, Chen Q, Gong H (2020) Stmag: A spatial-temporal mixed attention graph-based convolution model for multi-data flow safety prediction. Inf Sci 525:16–36
21. Wang B, Sun Y, Chu Y, Min C, Yang Z, Lin H (2023) Local discriminative graph convolutional networks for text classification. Multimed Syst 1–11
22. Huang H, Song Y, Wu Y, Shi J, Xie X, Jin H (2020) Multitask representation learning with multiview graph convolutional networks. IEEE Trans Neural Netw Learn Syst 33(3):983–995
23. Dai M, Guo W, Feng X (2020) Over-smoothing algorithm and its application to gcn semi-supervised classification. In: Data science: 6th international conference of pioneering computer scientists, engineers and educators, ICPCSEE 2020, Taiyuan, China, September 18-21, 2020, Proceedings, Part II 6, pp 197–215. Springer
24. Yang R, Dai W, Li C, Zou J, Xiong H (2023) Tackling over-smoothing in graph convolutional networks with em-based joint topology optimization and node classification. IEEE Trans Signal Inf Process Netw 9:123–139
25. Oono K, Suzuki T (2020) Graph neural networks exponentially lose expressive power for node classification. In: International conference on learning representations
26. Topping J, Di Giovanni F, Chamberlain BP, Dong X, Bronstein MM (2022) Understanding over-squashing and bottlenecks on graphs via curvature. In: International conference on learning representations
27. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y et al (2017) Graph attention networks. Stat 1050(20):10–48550
28. He L, Bai L, Yang X, Du H, Liang J (2023) High-order graph attention network. Inf Sci 630:222–234
29. Rong Y, Huang W, Xu T, Huang J (2020) Dropedge: Towards deep graph convolutional networks on node classification. In: International conference on learning representations
30. Alon U, Yahav E (2021) On the bottleneck of graph neural networks and its practical implications. In: International conference on learning representations
31. Giraldo JH, Skianis K, Bouwmans T, Malliaros FD (2023) On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In: Proceedings of the 32nd ACM international conference on information and knowledge management, pp 566–576
32. Wang J, Liang J, Cui J, Liang J (2021) Semi-supervised learning with mixed-order graph convolutional networks. Inf Sci 573:171–181
33. Abu-El-Haija S, Perozzi B, Kapoor A, Alipourfard N, Lerman K, Harutyunyan H, Ver Steeg G, Galstyan A (2019) Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning, pp 21–29. PMLR
34. Liu X, Xia G, Lei F, Zhang Y, Chang S (2021) Higher-order graph convolutional networks with multi-scale neighborhood pooling for semi-supervised node classification. IEEE Access 9:31268–31275
35. Liu X, Lei F, Xia G (2023) Mulstepnet: stronger multi-step graph convolutional networks via multi-power adjacency matrix combination. J Ambient Intell Humaniz Comput 14(2):1017–1026
36. Zhao L, Akoglu L (2020) Pairnorm: Tackling oversmoothing in gnns. In: International conference on learning representations
37. Zhou K, Huang X, Li Y, Zha D, Chen R, Hu X (2020) Towards deeper graph neural networks with differentiable group normalization. Adv Neural Inf Process Syst 33:4917–4928
38. Chien E, Peng J, Li P, Milenkovic O (2021) Adaptive universal generalized pagerank graph neural network. In: International conference on learning representations
39. Chai Z, Zhang T, Wu L, Han K, Hu X, Huang X, Yang Y (2023) Graphllm: Boosting graph reasoning ability of large language model. arXiv:2310.05845
40. Tang J, Yang Y, Wei W, Shi L, Su L, Cheng S, Yin D, Huang C (2023) Graphgpt: Graph instruction tuning for large language models. arXiv:2310.13023
41. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, Yu PS (2019) Heterogeneous graph attention network. In: The world wide web conference, pp 2022–2032
42. Yu R, Wang L, Xin Y, Qian J, Dong, Y (2023) A gated graph attention network based on dual graph convolution for node embedding. Appl Intell, 1–14
43. Chen J, Fang C, Zhang X (2023) Global attention-based graph neural networks for node classification. Neural Process Lett 55(4):4127–4150
44. Ye Y, Ji S (2021) Sparse graph attention networks. IEEE Trans Knowl Data Eng 35(1):905–916
45. Krogh A, Vedelsby J (1994) Neural network ensembles, cross validation, and active learning. Adv Neural Inf Process Syst 7
46. Chen M, Wei Z, Huang Z, Ding B, Li Y (2020) Simple and deep graph convolutional networks. In: International conference on machine learning, pp 1725–1735. PMLR
47. Wang H, Zhao M, Xie X, Li W, Guo M (2019) Knowledge graph convolutional networks for recommender systems. In: The world wide web conference, pp 3307–3313
48. Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: International conference on machine learning, pp 6861–6871. PMLR
49. Nt H, Maehara T (2019) Revisiting graph neural networks: All we have is low-pass filters. arXiv:1905.09550
50. Chamberlain B, Rowbottom J, Gorinova MI, Bronstein M, Webb S, Rossi E (2021) Grand: Graph neural diffusion. In: International conference on machine learning, pp 1407–1418. PMLR
51. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32

52. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980
53. Li Q, Han Z, Wu X-M (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
54. Sun K, Lin Z, Zhu Z (2020) Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 5892–5899