Learning High-Order Graph Convolutional Networks via Adaptive Layerwise Aggregation Combination

Article *in* IEEE Transactions on Neural Networks and Learning Systems · October 2021

DOI: 10.1109/TNNLS.2021.3119958

3 authors:

Tianqi Zhang
Shanghai Jiao Tong University
4 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Qitian Wu
Shanghai Jiao Tong University
30 PUBLICATIONS **210** CITATIONS
SEE PROFILE

Junchi Yan
Shanghai Jiao Tong University
270 PUBLICATIONS **5,961** CITATIONS
SEE PROFILE

Learning High-Order Graph Convolutional Networks via Adaptive Layerwise Aggregation Combination

Tianqi Zhang, Qitian Wu, and Junchi Yan, *Senior Member, IEEE*

***Abstract*— Graph convolutional networks have attracted wide attention for their expressiveness and empirical success on graph-structured data. However, deeper graph convolutional networks with access to more information can often perform worse because their low-order Chebyshev polynomial approximation cannot learn adaptive and structure-aware representations. To solve this problem, many high-order graph convolution schemes have been proposed. In this article, we study the reason why high-order schemes have the ability to learn structure-aware representations. We first prove that these high-order schemes are generalized Weisfeiler–Lehman (WL) algorithm and conduct spectral analysis on these schemes to show that they correspond to polynomial filters in the graph spectral domain. Based on our analysis, we point out twofold limitations of existing high-order models: 1) lack mechanisms to generate individual feature combinations for each node and 2) fail to properly model the relationship between information from different distances. To enable a node-specific combination scheme and capture this interdistance relationship for each node efficiently, we propose a new adaptive feature combination method inspired by the squeeze-and-excitation module that can recalibrate features from different distances by explicitly modeling interdependencies between them. Theoretical analysis shows that models with our new approach can effectively learn structure-aware representations, and extensive experimental results show that our new approach can achieve significant performance gain compared with other high-order schemes.**

***Index Terms*— Adaptive combination methods, graph neural networks (GNNs), graph representation learning, high-order graph convolutional networks.**

I. Introduction

REAL-WORLD data in many applications can be represented by graphs, which contain entities with different types (denoted by nodes) and define relations between them (denoted by edges), such as citation networks, social networks, molecules, and transportation systems. To leverage and exploit rich information from graph-structured data, graph representation learning plays a key role before downstream tasks and aims at mapping a graph into a set of low-dimensional node embeddings [1]–[7]. Such node embeddings that capture both semantic and topological features of graphs can provide sufficient representation power, easy accessibility, and highly efficient computation for various downstream tasks over graphs.

Recently, graph neural networks (GNNs) [3], [8] have come into the spotlight due to their superior power for graph representation learning, and many works can be further improved via exploiting the power of GNNs [9]–[13]. The most well-known GNN model is graph convolution networks (GCNs) [5]–[7] whose basic idea is to adapt convolution operations over grid data in the Euclidean space to graph data and iteratively aggregate neighbored nodes' features to update the node representations. There exist two schools of thinking toward understandings the rationale behind GCN. From the message passing perspective, the aggregation propagates information between adjacent nodes and enables each node to incorporate node features in a certain receptive field. From the signal processing perspective, the convolution filters play as a transformation of graph signals and help in extracting informative features.

Despite its theoretical soundness, GCN encounters a serious issue when applied in practice: as the layer number increases, the model performance would significantly degrade [7]. For example, Kipf and Welling [7] observe that GCN often achieves state-of-the-art results on citation networks when using only two layers. This is counterintuitive since deeper models should have access to more information from distant nodes (which we call high-order neighbors in this article). There is plenty of evidence that such a problem is due to oversmoothing, i.e., the node embeddings become indistinguishable after deep GCN layers [14]. Recent works [14] point out that the GCN layers essentially serve as Laplacian smoothing over graph data, which leads to similar node representations and the oversmoothing issue.

To address the oversmoothing, many works study properties and limitations of the neighborhood aggregation scheme used in GCN and propose high-order structure-aware graph convolution schemes [15]–[19]. They find that GCN's neighborhood aggregation scheme, which is a first-order Chebyshev polynomial approximation, will give the global graph representation

Manuscript received April 16, 2021; revised September 22, 2021; accepted October 4, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0107600, in part by the Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102, and in part by NSFC under Grant 61972250 and Grant 72061127003. *(Corresponding author: Junchi Yan.)*

The authors are with the Department of Computer Science and Engineering, and the MoE Key Laboratory of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: lygztq@sjtu.edu.cn; echo740@sjtu.edu.cn; yanjunchi@sjtu.edu.cn).

Data is available on-line at https://github.com/lygztq/SEAggregation.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3119958.

Digital Object Identifier 10.1109/TNNLS.2021.3119958

and preserve limited information about individual nodes as the network goes deeper. Instead of that, these high-order schemes, as illustrated in Fig. 1, attempt to use stacked aggregations to collect the representations of neighbors at various distances and consider a weighted combination method mixing these representations to obtain final representations. However, these approaches either highly rely on handcrafted methods for the combination of representations from different aggregations or lack proper modeling of the coupling effects between representations from different aggregations. Since graphs are non-Euclidean data and there exists heterogeneity among different nodes, the fixed combination method may constrain the model's flexibility. Moreover, improper modeling of the relationship between representations from different aggregations can bring noise and damage the performance of networks. Models such as DAGNN [20] do not even consider this relationship.

To solve the dilemma of these high-order schemes, in this article, we first investigate the rationale behind these high-order graph convolution schemes from two perspectives: the equivalence to the Weisfeiler–Lehman (WL) algorithm and the analysis on the spectral domain. Both results show that high-order graph convolutional networks have the ability to learn structural-aware features. Firstly, we prove that the high-order methods are equivalent to a generalized version of the WL algorithm [21], which is widely used to distinguish isomorphic graphs. Our analysis bridges the gap between high-order graph convolutions and the WL algorithm, which indicates that high-order graph convolution can learn structure-aware representations for graphs.

As further enlightenment, we also conduct spectral analysis on these high-order schemes and show that they essentially play as polynomial filters with arbitrary coefficients in the graph spectral domain. The polynomial filters can treat graph signals from various distances in different manners, which helps to further explain the structure-awareness trait of high-order methods from another perspective. However, existing high-order methods with fixed combination methods (such as weighted sum, concatenation, or max-pooling) induce a fixed and static filter, which would limit the capacity of the GNN model to learn node-adaptive importance on information from different orders. Also, improper specification of the relationship between different filters in previous models would further constrain the power of high-order methods. For example, combination methods using LSTM [22], [23] to model features from a different distance as sequences cannot make proper use of all information from the aggregation results obtained by deep networks due to the poor support of recurrent networks for long sequences, while the method in DAGNN may generate noise on large regions on graphs since it does not consider the interdependency between different features.

Based on these theoretical insights, we propose a new adaptive combination method, coined squeeze-and-excitation aggregation (SE-aggregation) method, for high-order graph convolution schemes to model the relationship between features from different distances properly. Specifically, our new method can learn to generate a unique combination scheme for each node in graphs and adaptively gather representations from different distances. We borrow the squeeze-and-excitation (SE) idea from SENet [24] and generate linear combination weights of feature representations from different distances for each node in graphs via an SE module. This gives the graph convolution operator the ability to provide a unique polynomial filter for each node and model the relationship between features from different orders, which further enlarges the model capability. We compare our new combination method with other common methods used in high-order graph convolutional networks theoretically in Section III-F and Table I. To elaborate on the contribution of SE-aggregation, we also perform an ablation study and compare the SE-aggregation method with other combination methods used in high-order graph convolution. Moreover, to show that the SE module can make better use of information from large regions on graphs with many aggregation results, we conduct experiments on deep models with depth from 32 to 256. To further explore this kind of method that we proposed, we also try a self-attention [25]-based combination method, which is a more complex version of our SE-aggregation method. In addition, experiments on node classification tasks demonstrate the soundness of our analysis and the superiority of our proposed scheme compared with other high-order schemes. The contributions of the article are given as follows.

1) We perform theoretical analysis on high-order graph convolution [19], and the analysis shows that high-order graph convolution is equivalent to a generalized WL algorithm and a polynomial filter in the spectral domain, which helps to explain why high-order schemes have the ability to learn structure-aware representations.
2) Based on our analysis, we point out the weakness of existing high-order graph convolution methods (lack capability for an adaptive combination of information from different distances with regard to each centered node, and ignore the coupling effects between node features in different layers) and propose a new node and depth adaptive method based on SE mechanism to address this weakness.
3) Ablation studies and experimental results on node classification demonstrate the soundness of our analysis and the superiority of our proposed method against other high-order methods. For further exploration, we also try a self-attention-based combination method as a more complex version of our proposed method. The source code has been made publicly available.

II. Related Works

In this section, we introduce some fields, problems, and works involved in our research. We first give a brief history of the graph convolutional networks. Then, we introduce the oversmoothing problem and its existing solutions. Finally, we introduce high-order graph convolution methods.

A. Graph Convolutional Networks

In the seminal work, Bruna *et al.* [5] show how to generalize convolutional neural networks (CNNs) to graph-structured data

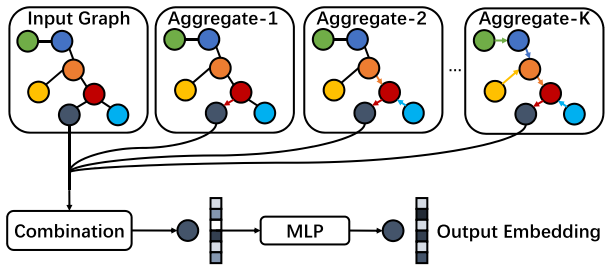


Fig. 1. Illustration of higher-order graph convolution schemes proposed in this article. These schemes use stacked aggregations to collect the representations of neighbors at various distances, and a weighted combination method is developed for mixing these representations to obtain final representations.

from a spectral perspective. To make the spectral filters spatially localized and more efficient, Defferrard *et al.* [6] propose ChebNet based on the K-order Chebyshev polynomials to define graph convolution without expensive Laplacian eigendecomposition. GCN [7] is a special case of ChebNet that uses the first-order Chebyshev polynomials and a redefined adjacency matrix to perform graph convolution. Recently, many variants of GCN have been proposed. GAT [26] introduces an attention mechanism that estimates the importance of different neighbors on the central node and uses the attention scores as weights for feature aggregation. SGC [27] separates the aggregation and update operations of one graph convolutional layer and shrink aggregation operations into one module at the beginning of the model as a preprocess. It also removes all the nonlinearities to combine update operations into a single linear layer. GIN [28] also derives its update scheme from the WL algorithm [21]. However, the GIN is based on the original graph and uses only first-order neighbors from aggregation, while we derive the update scheme from the WL algorithm on a new graph with high-order edges. Due to the limitations of memory and device, traditional GNNs are difficult to train on large graphs. To solve this problem, GraphSage [29] uses a neighborhood sampling mechanism to perform a minibatch training. Apart from the sampling method, Cluster-GCN [30] divides the input graph into several subgraphs and performs training on these subgraphs. These scalable GNN models are orthogonal to our work, and our proposed model can also combine with them to achieve hardware-efficient computation.

B. Oversmoothing and Deep Networks

Because high-order graph convolution methods are proposed to solve the oversmoothing problem, it is necessary to introduce the oversmoothing problem and related works here. There are plenty of empirical observations that node representations would become indistinguishable when GNN models go deep, which is referred to as an oversmoothing problem, a pain point for GNN models [7]. It has been proven in [14] that the aggregation operation in the GCN model is a special case of Laplacian smoothing on graphs. Excessive smoothing makes the features indistinguishable and affects the model performance for downstream tasks. This makes it impossible for GCN to judge the importance of node information at different distances, which limits the use of information from further nodes. To address the oversmoothing issue, many works propose different solutions. Kipf and Welling [7] use residual connections borrowed from CNNs to alleviate oversmoothing, while Chiang *et al.* [30] propose a renormalization trick to further improve deep model training. The PPNP/APPNP [31] utilizes a propagation scheme derived from personalized PageRank [32], which permits the use of infinitely many propagation steps. GCNII [33] also adopts initial residual connection and identity mapping to stack nonlinear transforms used to combine graph signals filtered with different orders of graph Laplacian.

C. High-Order Graph Convolution

Apart from these variants of GCN that directly performs aggregation on the first-order neighbors, there also exist methods that exploit information from high-order neighbors. N-GCN [15] trains multiple instances of GCNs that use different order adjacency matrices as the propagation matrix and concatenates results from different GCN networks at the end. Similarly, IncepGCN [16], [17] uses kernels with different sizes in one convolution layer and combines convolution results at the end of each layer. Based on the random-walk theory, JK-Net [18] uses stacked aggregation layers as the propagation scheme and combines all aggregation results at the end of their model. MixHop [19] repeatedly mixes feature representations of neighbors at various distances. However, the mechanisms used to combine representations from different distances in these high-order models are fixed and not learnable. Different from these methods, a recent work DAGNN [20] devises a trainable projection vector to compute attention scores for representations generating from various distances. Nevertheless, DAGNN ignores the coupling effects among node features in different orders and would suffer undesirable capacity for learning on graphs with various properties and large node regions. To show this difference, we compare our new combination method with other common methods used in high-order graph convolutional networks in Section III-F.

III. Model Analysis and Methodology

In this section, we first generalize the 1-dim WL (WL-1) algorithm [21] to a differentiable and parameterized form on a new graph where two nodes have an edge, provided that the two nodes are reachable to each other in at most K hops in the original graph. The edge weights in this new graph imply how we handle information at different distances. For example, if two nodes are far apart on the original graph, we can assign a small weight to their edges on the new graph. Next, we show that high-order graph convolution schemes are equivalent to the generalized WL-1 algorithm. After that, we prove that high-order graph convolution schemes are polynomial filters on the spectral domain. These two results both show that high-order graph convolutional networks have the ability to learn structural-aware features. Based on our analysis, we propose a new SE-aggregation method to enhance high-order graph convolutions.

Algorithm 1 WL-1 Algorithm [21]

Input: Initial node label $(h_1^{(0)}, h_2^{(0)}, \ldots, h_n^{(0)})$
Output: Final node label $(h_1^{(T)}, h_2^{(T)}, \ldots, h_n^{(T)})$

```
t ← 0;
repeat
    for v ∈ V do
        h_v^(t+1) ← hash(Σ_{u∈N(v)} h_u^(t));
    end
    t ← t + 1;
until stable node label is reached;
```

We define notations in this article here. A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set with $|\mathcal{V}| = n$ and $\mathcal{E}$ is the edge set. Denote $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ the node features, where $\mathbf{x}_v \in \mathbb{R}^m$ is the m-dimensional feature vector for node v. Given an L-layer network, vector $\mathbf{h}_v^{(l)}$ is the state of node v at the lth layer where $0 \le l \le L$ and $\mathbf{h}_v^{(0)} = \mathbf{x}_v$.

A. Preliminaries of WL-1 Algorithm

The 1-dim WL (WL-1) algorithm (see Algorithm 1) is the 1-D form of WL test [21] for graph isomorphism: two graphs are isomorphic if they are topologically identical. Graph isomorphism is challenging since no algorithm running in polynomial time is known yet [28]. However, the WL test can distinguish broad classes of graphs [34] both effectively and efficiently if we do not consider some corner cases [35], which makes it powerful for graph isomorphism. The WL test iteratively aggregates the labels of nodes and their neighbors and feeds the aggregation result into a hash function to obtain the unique new labels until the labels converge. The process can be written as

$$h_v^{(t+1)} \leftarrow \text{hash}\left(\sum_{u \in \mathcal{N}(v)} h_u^{(t)}\right) \tag{1}$$

where $\mathcal{N}(v)$ is the set of neighbors of node v and $h_v^{(t)}$ is the label of node v at time step t. The WL test decides two graphs are isomorphic if their stable labels are the same.

B. Generalization of the WL-1 Algorithm

Since high-order schemes perform operations on multihop neighborhoods, to derive methods that can exploit both information from high-order and low-order neighbors on graphs, we generalize the WL-1 algorithm on a new graph with both high- and low-order edges. We also generalize the node state from a 1-dim scalar to an m-dim vector.

Let $\mathcal{G}$ be the original graph and $\mathcal{G}'$ be the new graph where two nodes have an edge, provided that these two nodes are reachable to each other in at most K hops in $\mathcal{G}$. For node v, we use $\mathcal{N}^i(v)$ to denote the set of nodes reachable from v in at most i hops and $\mathcal{N}^1(v) = \mathcal{N}(v)$.

The most significant step in the WL-1 algorithm is to gather information from all neighbors of a central node and pass the gathered information to a hash function. On graph $\mathcal{G}'$, this can be written as

$$\mathbf{h}_v^{(t+1)} \leftarrow \text{hash}\left(\sum_{u \in \mathcal{N}^K(v)} \mathbf{h}_u^{(t)}\right). \tag{2}$$

To make it differentiable, we use a multilayer perceptron to approximate the hash function. This is possible because of the universal approximation theorem [28], [36], [37]. We also change the sum $\sum_{u \in \mathcal{N}^K(v)} \mathbf{h}_u^{(t)}$ to a weighted linear combination to introduce edge weights for node state aggregation. Thus, we can rewrite (2) into

$$\mathbf{h}_v^{(t+1)} \leftarrow \text{MLP}\left(\sum_{u \in \mathcal{N}^K(v)} w_{u,v} \mathbf{h}_u^{(t)}\right) \tag{3}$$

where $w_{u,v}$ is the weight of edge (u, v).

C. Model Analysis

To perform analysis on high-order graph convolution schemes, we first give the formal definition. Theoretically, performing aggregation on graphs with high-order edges needs the power of the adjacency matrix of the original graph; this may be problematic in realistic applications. The adjacency matrix is usually stored in the sparse matrix form in implementation, while high-order adjacency matrices are often much denser than the first-order adjacency matrix, which may cause large space cost and low computational speed. A common practice is to use stacked one-hop aggregations to achieve the equivalent result. Given the hidden state $\mathbf{h}_v^{(l)}$ of node v at layer l, in stacked aggregations of the lth layer, we use $\mathbf{h}_{v,i}^{(l)}$ to denote the result of the ith aggregation at node v, and the initial node state is $\mathbf{h}_{v,0}^{(l)} = \mathbf{h}_v^{(l)}$. We can write the result after the $(i+1)$th aggregation as

$$\mathbf{h}_{v,i+1}^{(l)} = \sum_{u \in \mathcal{N}(v)} c_{u,v,i}^{(l)} \mathbf{h}_{u,i}^{(l)} \tag{4}$$

where $0 \le c_{u,v,i}^{(l)} \le 1$ is the aggregation constant between node u and v at the ith aggregation layer and the lth global layer. Common aggregation functions, such as max, mean, and sum, can all be written as this form (e.g. $c_{u,v,i}^{(l)} = 1/|\mathcal{N}(v)|$ for mean aggregation), and the general form of $c_{u,v,i}^{(l)}$ in unweighted graphs is given by

$$c_{u,v,i}^{(l)} = \alpha_i^{(l)}(u, v) e_{u,v} \tag{5}$$

where $\alpha_i^{(l)}(u, v)$ is the aggregator and $e_{u,v} = 1$ if there is an edge between u and v. For example, the aggregator of max aggregation can be written as

$$\alpha_i^{(l)}(u, v) = \mathbb{I}\left(u = \arg\max_{n \in \mathcal{N}(v)} \mathbf{h}_{n,i}^{(l)}\right) \tag{6}$$

where $\mathbb{I}$ is the indicator function. We combine all these aggregation results via a linear combination with learnable weights followed by a multilayer perceptron at the end to get information from both high- and low-order neighbors

$$\mathbf{h}_v^{(l+1)} = \text{MLP}\left(\sum_{i=0}^{K} w_i \mathbf{h}_{v,i}^{(l)}\right). \tag{7}$$

Next, we show the equivalence between the high-order scheme and the generalized WL-1 algorithm.

Theorem 1: Given the input graph $\mathcal{G}$ and an L-layer GCN with high-order graph convolution scheme, as described in (4) and (7), the output of this GCN is equivalent to applying a generalized WL-1 algorithm on $\mathcal{G}$ with L iterations.

Proof: To prove Theorem 1, we first show that high-order graph convolution schemes have the same form as the generalized WL-1 algorithm in a single graph convolution layer. At the lth layer, we expand the aggregation result $\mathbf{h}_{v,i}^{(l)}$ as

$$\begin{aligned}\mathbf{h}_{v,0}^{(l)} &= \mathbf{h}_{v}^{(l)} \\ \mathbf{h}_{v,i}^{(l)} &= \sum_{u\in\mathcal{N}(v)} c_{u,v,i-1}^{(l)}\mathbf{h}_{u,i-1}^{(l)} \\ &= \sum_{u\in\mathcal{N}^i(v)} c_{u,v,i,0}^{(l)}\mathbf{h}_{u}^{(l)} \end{aligned} \tag{8}$$

where $c_{u,v,i,0}^{(l)} = \sum_{p\in\mathcal{P}}\prod_{j=0}^{i-1} c_{n_j,n_{j+1},j}^{(l)}$ is the sum of products of all constant c in paths from u to v with length i, $p = (u, n_1, \ldots, n_{i-1}, v)$ is a path with length i from u to v, $n_0 = u$, and $n_i = v$. $\mathcal{P} = \{p \mid p = (u, n_1, \ldots, n_{i-1}, v)\}$ is the set all such paths. Assume that there are K aggregations in total, and combining (7), we rewrite the final expression as

$$\begin{aligned}\mathbf{h}_{v}^{(l+1)} &= \mathrm{MLP}\left(\sum_{i=0}^{K} w_i\mathbf{h}_{v,i}^{(l)}\right) \\ &= \mathrm{MLP}\left(\sum_{u\in\mathcal{N}^K(v)}\left(\sum_{i=0}^{K} w_i c_{u,v,i,0}^{(l)}\right)\mathbf{h}_{u}^{(l)}\right) \\ &= \mathrm{MLP}\left(\sum_{u\in\mathcal{N}^K(v)} w_{u,v}\mathbf{h}_{u}^{(l)}\right) \end{aligned} \tag{9}$$

where $w_{u,v} = \sum_{i=0}^{K} w_i c_{u,v,i,0}^{(l)}$. Equation (9) is exactly the form of generalized WL-1 algorithm. Hence, to perform high-order graph convolution on graphs is to perform an iteration of the generalized WL-1 algorithm. This proves Theorem 1. □

D. Spectral Analysis

The equivalence between high-order graph convolution schemes and the generalized WL-1 algorithm reveals that high-level schemes have the ability to learn structure-aware representations on graphs. To make this conclusion more concrete, we perform spectral analysis on these high-order schemes under GCN [7] settings, and the analysis explains their ability from a graph signal processing perspective, which provides another view to understand this property. Specifically, assume that the propagation matrix used in aggregation is the symmetric normalized adjacency matrix in GCN [7]; we have the following theorem.

Theorem 2: Given a GCN with high-order graph convolution scheme and propagation matrix as $\mathbf{S} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{A}$, each graph convolution layer in this GCN corresponds to a polynomial filter on the spectral domain.

Proof: To prove this theorem, we first show that using propagation matrix $\mathbf{S}^k$ to perform aggregation is equivalent to applying low-pass filtering with fixed parameters. The propagation matrix used in GCN [7] is an approximation with first-order Chebyshev filter. To eliminate numerical instabilities and exploding/vanishing gradients, the author propose the renormalization trick, where the propagation matrix is changed from $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ to $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, where $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{A}$. Thus, the normalized Laplacian is also changed from $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ to $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. The spectral filters corresponding to $\mathbf{S}^k$ are then a polynomial of the eigenvalues of $\tilde{\mathbf{L}}$

$$\hat{g}(\tilde{\lambda}_i) = (1 - \tilde{\lambda}_i)^k \tag{10}$$

where $\tilde{\lambda}_i$ is the eigenvalue of $\tilde{\mathbf{L}}$. Since all eigenvalues of $\tilde{\mathbf{L}}$ lie between 0 and 2 [38] and the renormalization trick can make the largest eigenvalue shrink from 2 to approximately 1.5 and then eliminate the effect of negative coefficients [27, Th. 1], the propagation matrix $\mathbf{S}^k$ acts as a low-pass filter.

We now demonstrate that the high-order graph convolution scheme corresponds to a learnable linear combination of low-pass fixed filters with different orders, which forms a polynomial filter in the spectral domain. Suppose that a graph signal with single channel is $\mathbf{f} \in \mathbb{R}^n$; under the GCN [7] aggregation scheme, we have

$$\mathbf{S}^k\mathbf{f} = \tilde{\mathbf{U}}\hat{g}_k(\tilde{\Lambda})\tilde{\mathbf{U}}^\top\mathbf{f} \tag{11}$$

where $\tilde{\mathbf{U}}$ is the matrix of eigenvectors of $\tilde{\mathbf{L}}$, $\tilde{\Lambda}$ is the diagonal matrix of eigenvalues of $\tilde{\mathbf{L}}$, and $\hat{g}_k(\tilde{\Lambda})$ is also a diagonal matrix with ith element be $(1-\tilde{\lambda}_i)^k$. As shown in (7), the high-order convolution scheme combines all aggregation results at the end with a linear combination

$$\begin{aligned}\sum_{k=0}^{K} w_k\mathbf{S}^k\mathbf{f} &= \sum_{k=0}^{K} w_k\tilde{\mathbf{U}}\hat{g}_k(\tilde{\Lambda})\tilde{\mathbf{U}}^\top\mathbf{f} \\ &= \tilde{\mathbf{U}}\left(\sum_{k=0}^{K} w_k\hat{g}_k(\tilde{\Lambda})\right)\tilde{\mathbf{U}}^\top\mathbf{f} \end{aligned} \tag{12}$$

where w_k is the learnable weight. Thus, we get a new convolution kernel $\sum_{k=0}^{K} w_k\hat{g}_k(\tilde{\Lambda})$ whose ith element is a polynomial of eigenvalue $\tilde{\lambda}_i$ of $\tilde{\mathbf{L}}$. Since high-order schemes do not contain interchannel information exchange, this conclusion can be easily extended to multidimensional situations. Therefore, the high-order graph convolution scheme corresponds to a polynomial filter on the spectral domain. □

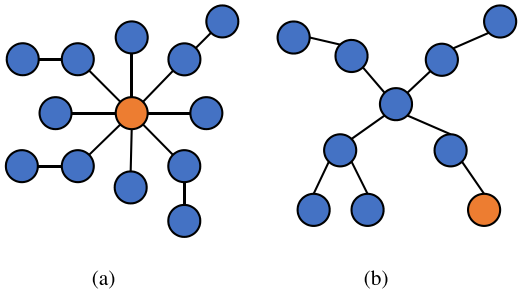


Fig. 2. Subgraphs with different structures (star and tree). Different subgraph structures around the central node (denoted in orange) result in very different neighborhood (denoted in blue) sizes. See the first paragraph in Section III-E. (a) Star. (b) Tree.

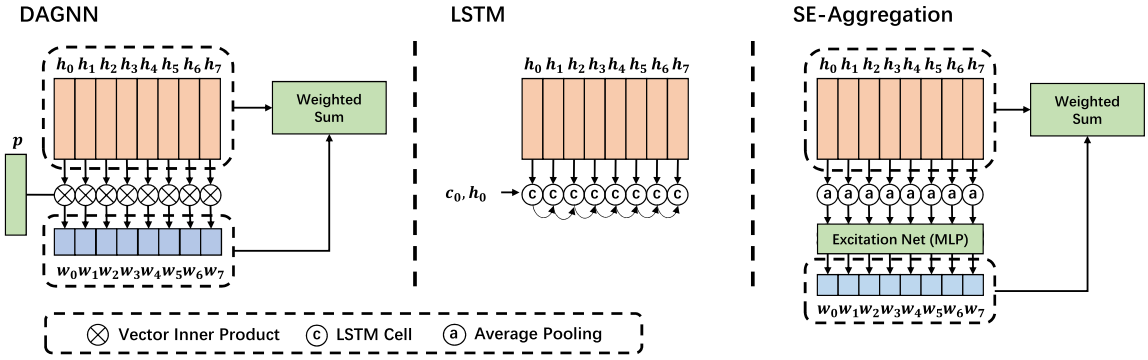


Fig. 3. Comparison between SE-aggregation and other adaptive combination methods used in high-order graph convolutional networks. DAGNN does not consider the interdependencies between input features. Our model can adaptively recalibrate features by explicitly modeling interdependencies (rather than the temporal dependency in the LSTM combination method) between them and generating weight for each feature vector according to these interdependencies.

However, current high-order models have two limitations: 1) lack mechanisms to generate individual feature combinations for each node and 2) fail to properly model the relationship between information from different distances. Next, we will discuss these limitations in detail and give our solution.

E. SE-Aggregation Module

Most existing high-order schemes use handcrafted combination methods to collect information from various distances. For example, max-pooling and concatenation are used as combination methods in JK-Net [18]. However, graphs are non-Euclidean data, and different nodes in graphs may need different combination methods. This is because different subgraph structures result in very different neighborhood sizes. For example, as illustrated in Fig. 2, the central node in a star subgraph only needs one aggregation to gather all useful information. Therefore, this central node may give very large weights to features from one-hop neighbors, while features from multihop neighbors will be assigned with small weights. On the contrary, the leaf node in a tree subgraph may give all its neighbors similar weights to obtain enough information from its neighborhood. Although there exist methods that can generate unique combination schemes for different nodes, these methods are either based on sequence (e.g. LSTM in JK-Net) or independent of features from other hops (the projection vector in DAGNN [20]), which are not suitable for the high-order graph convolution. When the network tries to collect features from a long range of nodes on the graph with different orders, the number of aggregation results in one layer (i.e. K in Section III-C) will become extremely large. If we treat the set of aggregation results from nodes with different orders $\{\mathbf{h_0}, \mathbf{h_1}, \ldots, \mathbf{h_K}\}$ as a sequence, pure sequence-based methods, such as LSTM, will have poor performance compared with attention-based method on long sequences [39]. On the other hand, if we generate a combination weight for each aggregation result independently and adaptively, the linear combination on this large set $\{\mathbf{h_0}, \mathbf{h_1}, \ldots, \mathbf{h_K}\}$ with unrelated weights generated dynamically will perform like noise because of the central limit theorem since the generated weights are independent of each other, and we cannot guarantee that the range of generated weights is bounded.

To solve these issues, we need a new method based on the attention-like mechanism to model the relationship between features with different orders, and we also hope that this new method is computationally efficient and can generate unique combination scheme for each node on graphs. We propose a new adaptive combination method, coined the SE-aggregation method, which is inspired by SENet [24]. The SE block in SENet can adaptively recalibrates channelwise feature responses by explicitly modeling interdependencies between channels and generating weight for each channel according to these interdependencies, which is exactly what we want. To show the difference between the SE module and previous adaptive combination methods, we illustrate them in Fig. 3. As illustrated in Fig. 4, our method uses a similar SE mechanism to obtain combination weights of different aggregation results. Given results $\{\mathbf{h}_{v,0}, \mathbf{h}_{v,1}, \ldots, \mathbf{h}_{v,K}\}$ of node v, the SE-aggregation first squeezes these aggregation results via average pooling along the node state channel and then concatenates the squeezed results to obtain a vector $\mathbf{z}_v = [z_{v,0}, z_{v,1}, \ldots, z_{v,K}]^\top$ where $z_{v,i}$ is the average of all elements in $\mathbf{h}_{v,i}$. To enlarge the capacity of our network, the average pooling can be replaced by a projection vector $\mathbf{t}$, where $z_{v,i} = \mathbf{t}^\top \mathbf{h}_{v,i}$. Then, we feed $\mathbf{z}_v$ into a two-layer perceptron acting as a gating mechanism with a tan$h$ activation

$$\mathbf{a}_v = \tan h(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z}_v)) \tag{13}$$

where δ refers to the ReLU function [40], $\mathbf{W}_1 \in \mathbb{R}^{r(K+1)\times(K+1)}$, and $\mathbf{W}_2 \in \mathbb{R}^{(K+1)\times r(K+1)}$. r is the reduction ratio to limit model complexity and aid generalization. Finally, we use $\mathbf{a}_v$ as the weight to combine all aggregation results

$$\mathbf{h}'_v = \sum_{i=0}^{K} a_{v,i} \mathbf{h}_{v,i} \tag{14}$$

where $a_{v,i}$ is the ith element of $\mathbf{a}_v$ and $\mathbf{h}'_v$ is the output feature of node v. The output of SE module $\mathbf{a}_v$ gives networks the ability to generate different combination schemes for each node.

F. Comparison With Other High-Order Networks

Table I shows the difference between our SE-aggregation method and common combination methods used by other

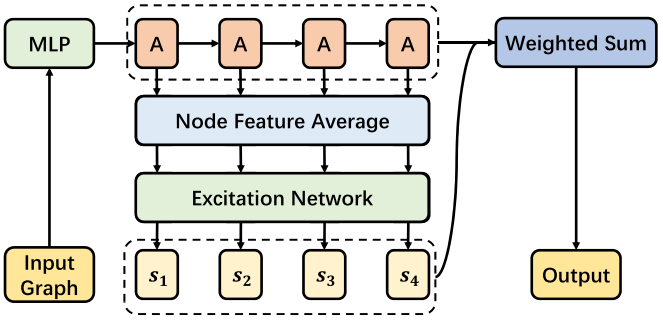


Fig. 4. High-order graph convolutional network with the proposed SE-aggregation module. Different aggregation (denoted by A) results will be fed to an SE module, and the output weights will be used to compute a weighted sum of aggregation results.

TABLE I
COMPARISON BETWEEN OUR SE-AGGREGATION METHOD AND COMBINATION METHODS USED IN OTHER HIGH-ORDER GRAPH CONVOLUTION MODELS

	Unique Combination	Modeling Relationship	Processing Large Region	Computationally Efficient
APPNP	×	×	✓	✓
concatenation	×	✓	×	×
mean/max pooling	×	×	×	✓
LSTM	×	✓	×	×
DAGNN	✓	×	×	✓
SE-aggregation	✓	✓	✓	✓

high-order graph convolutional networks, where APPNP and DAGNN mean the combination methods used by these two models. We list our observations and discussions on this comparison as follows. First, all handcrafted combination methods cannot generate individual combination schemes for each node since they use a fixed combination method for all nodes. Second, APPNP, pooling, and DAGNN methods cannot model the relationship between features from a different distance because of the lack of a corresponding mechanism. As mentioned before, although the LSTM combinator considers the relationship between features from different distances, it cannot process large node regions on graphs with many aggregation results due to the natural defect of recurrent networks in dealing with long sequences. The method used in APPNP can only extract a fixed point feature state since it uses the PageRank aggregation scheme. In addition, the method used in DAGNN may generate noise on a large node region since it does not consider the relationship between features from different distances. The same problem also appears in the pooling method where the network cannot distinguish the importance of features come from different distances. Moreover, it is almost computationally impossible for the concatenation method to process a large node region. Also, the LSTM combinator is not computationally efficient due to the temporal dependency of recurrent networks.

G. Self-Attention Aggregation

We also try a self-attention-based combination method, as a more complex version of our SE-aggregation method. The SE mechanism can be viewed as a simplified and channelwise version of self-attention; using self-attention over aggregation results instead of the SE method may bring more performance gain. Moreover, we need to process many aggregation results from different distances if we are going to extract information from a large region on graphs. If we treat these aggregation results as a long sequence, currently, the most powerful tool for this scenario is the Transformer [25], which is a model based on self-attention. Therefore, it is very valuable to explore this method.

Given aggregation results $\mathbf{X} \in \mathbb{R}^{N\times K\times F}$, where N is the number of nodes in the graph, K is the number of aggregations, and F is the dimension of node features, we obtain two new tensor $\mathbf{Q}$ and $\mathbf{K}$ by feeding $\mathbf{X}$ to two different linear layers. Different from the normal self-attention module, we use $\mathbf{X}$ as the input value $\mathbf{V}$ directly since we already have linear layers before. Then, this module will compute combination weights for each channel via a softmax function over inner products and output the combination result $\mathbf{X}'$

$$\mathbf{X}' = \text{softmax}\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{F}\right)\mathbf{X}. \tag{15}$$

Intuitively, this variant of our proposed method can achieve better performance on large graphs and deeper models with more aggregation results since self-attention can model long-range dependencies in sequences very well. This can be confirmed in ablation studies, as shown in Section IV-C. However, using self-attention also means higher computational cost compared with the original SE-aggregation.

H. Operation Decoupling Perspective

In fact, a high-order graph convolution scheme can also be explained from the model structure perspective. We decouple

TABLE II
STATISTICS OF GRAPH DATASETS USED IN THE ARTICLE

Dataset	Type	# Nodes	# Edges
CiteSeer	citation network	3,327	4,732
Cora	citation network	2,708	5,429
PubMed	citation network	19,717	44,338
Coauthor CS	co-authorship	18,333	44,324
Coauthor Physics	co-authorship	34,493	247,962
Amazon Computers	co-purchase	13,381	245,778
Amazon Photo	co-purchase	7,487	119,043
PPI	biology network	56,944	818,716

TABLE III
RESULTS ON THE PPI DATASET IN TERMS OF CLASSIFICATION ACCURACY (%)

	PPI
GraphSage	61.2
VR-GCN	97.8
GAT	97.3
JK-Net	97.6
Cluster-GCN	99.3
SE-aggregation	**99.4 ± 0.1**

TABLE IV
RESULTS ON CITATION DATASETS WITH FIXED SPLITS IN TERMS OF CLASSIFICATION ACCURACY AND F1-SCORE (%)

	Cora	CiteSeer	PubMed	Coauthor CS	Coauthor Physics	Amazon Computers	Amazon Photo
				Accuracy			
LogReg	50.2 ± 0.1	62.4 ± 0.1	69.5 ± 0.2	90.6 ± 0.4	89.6 ± 0.7	66.8 ± 1.5	79.7 ± 1.0
MLP	60.2 ± 0.7	59.6 ± 1.0	72.1 ± 0.5	90.9 ± 0.3	91.1 ± 0.5	65.4 ± 1.1	77.6 ± 1.7
LabelProp	72.0 ± 0.4	53.7 ± 0.7	70.9 ± 0.2	76.8 ± 1.6	82.6 ± 2.4	70.3 ± 5.1	70.5 ± 7.4
LabelProp NL	73.0 ± 0.4	53.6 ± 0.8	70.1 ± 0.3	81.2 ± 1.1	87.5 ± 1.2	72.3 ± 1.8	78.0 ± 4.5
GCN	81.2 ± 0.2	70.8 ± 0.4	79.0 ± 0.6	91.1 ± 0.3	92.9 ± 0.7	82.3 ± 0.8	91.1 ± 0.5
GAT	83.3 ± 0.3	70.9 ± 0.6	78.2 ± 0.4	90.2 ± 0.4	92.1 ± 0.9	77.3 ± 2.8	85.2 ± 3.4
APPNP	83.4 ± 0.5	71.5 ± 0.3	80.2 ± 0.4	91.5 ± 0.3	93.3 ± 0.5	82.6 ± 1.3	91.6 ± 0.5
MoNet	80.1 ± 0.6	67.8 ± 1.0	78.5 ± 1.4	90.8 ± 0.6	92.5 ± 0.9	83.5 ± 2.2	91.2 ± 1.3
SGC	81.5 ± 0.2	71.2 ± 0.2	79.1 ± 0.3	90.7 ± 0.2	93.1 ± 0.6	82.4 ± 1.1	90.3 ± 0.5
GraphSage-mean	81.1 ± 0.5	71.0 ± 0.7	78.8 ± 0.4	90.8 ± 0.3	92.7 ± 0.3	82.2 ± 1.2	90.4 ± 1.2
GraphSage-maxpool	73.9 ± 2.0	52.3 ± 1.3	76.6 ± 0.7	83.2 ± 0.8	90.2 ± 1.3	78.0 ± 1.7	89.7 ± 1.3
GraphSage-meanpool	78.9 ± 1.0	65.1 ± 0.9	78.3 ± 0.4	90.5 ± 0.3	91.1 ± 1.2	80.4 ± 1.5	90.1 ± 0.7
JK-Net-concat	80.9 ± 0.9	69.6 ± 3.2	77.6 ± 0.8	89.1 ± 0.4	91.4 ± 1.6	81.9 ± 2.1	89.5 ± 1.5
MixHop	81.9 ± 0.7	71.2 ± 0.3	77.4 ± 0.5	88.9 ± 0.3	89.4 ± 0.8	81.5 ± 2.4	88.6 ± 1.2
IncepGCN	81.2 ± 0.3	69.8 ± 0.4	77.6 ± 0.4	88.2 ± 0.5	90.2 ± 1.1	81.1 ± 2.5	89.7 ± 1.1
DAGNN	84.2 ± 0.5	73.2 ± 0.7	80.6 ± 0.2	93.5 ± 0.2	94.3 ± 0.3	83.1 ± 1.2	91.2 ± 0.7
SE-aggregation	**85.2 ± 0.4**	**73.6 ± 0.5**	**80.7 ± 0.2**	**93.7 ± 0.2**	**94.5 ± 0.2**	**84.1 ± 1.3**	**92.3 ± 0.8**
				Macro F1-score			
LogReg	49.4 ± 0.1	59.9 ± 0.1	69.2 ± 0.1	87.3 ± 0.5	86.5 ± 0.7	63.4 ± 0.9	77.8 ± 1.2
MLP	58.5 ± 0.5	56.8 ± 0.8	72.5 ± 0.5	87.3 ± 0.4	88.7 ± 0.4	62.1 ± 1.4	76.1 ± 1.7
LabelProp	71.5 ± 0.3	51.1 ± 0.7	69.4 ± 0.2	73.4 ± 1.4	74.6 ± 3.5	70.8 ± 2.2	68.8 ± 5.6
LabelProp NL	72.8 ± 0.3	51.3 ± 0.8	68.6 ± 0.3	77.5 ± 1.1	83.9 ± 1.0	72.5 ± 1.2	77.6 ± 2.9
GCN	80.1 ± 0.2	67.7 ± 0.4	78.5 ± 0.3	88.3 ± 0.5	90.9 ± 0.8	81.8 ± 0.8	89.1 ± 0.6
GAT	82.3 ± 0.3	66.5 ± 0.4	77.9 ± 0.4	87.9 ± 0.4	90.1 ± 0.8	77.6 ± 0.9	83.2 ± 1.4
APPNP	82.3 ± 0.4	68.3 ± 0.3	79.3 ± 0.4	88.9 ± 0.3	91.1 ± 0.6	82.3 ± 0.9	89.8 ± 0.6
MoNet	79.7 ± 0.5	64.7 ± 0.9	77.7 ± 1.3	87.3 ± 0.5	90.2 ± 0.7	81.2 ± 2.4	89.4 ± 1.5
SGC	80.2 ± 0.3	67.5 ± 0.2	78.2 ± 0.3	88.0 ± 0.2	91.0 ± 0.8	81.4 ± 1.1	88.5 ± 0.7
GraphSage-mean	79.8 ± 0.4	67.3 ± 0.5	78.7 ± 0.4	88.2 ± 0.3	90.2 ± 0.5	80.9 ± 1.1	88.8 ± 1.2
GraphSage-maxpool	72.6 ± 1.8	50.1 ± 1.5	75.9 ± 0.8	78.9 ± 0.8	89.1 ± 0.9	75.8 ± 1.8	87.6 ± 1.2
GraphSage-meanpool	77.8 ± 1.1	62.1 ± 1.1	77.8 ± 0.4	87.6 ± 0.4	89.7 ± 1.0	79.0 ± 1.7	88.3 ± 0.5
JK-Net-concat	79.2 ± 0.8	65.9 ± 2.6	77.2 ± 0.8	85.8 ± 0.6	88.9 ± 1.6	80.2 ± 1.5	87.4 ± 1.7
MixHop	80.7 ± 0.6	67.2 ± 0.4	77.2 ± 0.6	85.0 ± 0.3	88.4 ± 0.7	79.3 ± 2.2	86.4 ± 1.3
IncepGCN	80.2 ± 0.3	65.8 ± 0.3	76.4 ± 0.5	84.2 ± 0.4	87.9 ± 1.2	79.4 ± 2.4	87.7 ± 1.2
DAGNN	83.0 ± 0.4	68.1 ± 1.0	79.8 ± 0.2	91.1 ± 0.2	92.4 ± 0.3	82.3 ± 1.1	89.8 ± 0.6
SE-aggregation	**83.5 ± 0.4**	**68.4 ± 0.8**	**80.0 ± 1.0**	**92.0 ± 0.2**	**92.6 ± 0.3**	**83.0 ± 0.7**	**90.4 ± 0.6**

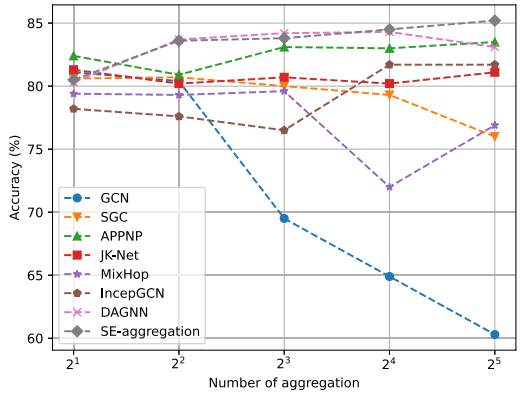


Fig. 5. Results on the Cora dataset with different depths (the number of aggregations) in terms of classification accuracy (%).

the aggregation and nonlinearity from a normal graph convolutional layer and recombine them by stacking layers of the same type together. By doing so, we can build more flexible network architectures by adjusting the ratio of the number of aggregations and nonlinearities according to the task at hand. By stacking multiple aggregations, the model can obtain a large receptive field, and the discrimination power can also be fulfilled by nonlinearities.

IV. EXPERIMENTS AND RESULTS

In this section, we conduct extensive experiments on node classification tasks to evaluate the superiority of our proposed SE-aggregation module. We first introduce datasets and experimental settings that we utilized. Next, we compare our method with other prior state-of-the-art methods to show the effectiveness of our method. After that, we conduct an ablation study and compare our method with other combination methods to elaborate the contribution of the SE-aggregation module. Finally, we study the performance variation with respect to network depth and training set size to further verify our method.

A. Experimental Setup

1) Datasets: We conduct experiments on seven semisupervised node classification datasets: citation networks [41] (Cora, CiteSeer, and PubMed), coauthorship networks [42] (Coauthor CS and Coauthor Physics), and copurchase networks [42] (Amazon Computers and Amazon Photo). We also evaluate our method on the PPI dataset [29], which is an inductive and multilabel classification task. Dataset statistics are summarized in Table II.

2) Baselines: In this article, we call methods where only first-order neighbors are used in aggregation the first-order

TABLE V
RESULTS ON CITATION NETWORKS WITH DIFFERENT DEPTH AND COMBINATION METHODS FOR HIGH-ORDER GRAPH CONVOLUTION IN TERMS OF CLASSIFICATION ACCURACY (%)
(OOM: OUT OF MEMORY)

Number of Aggregations		16	32	64	128	256
Cora	Constant	82.8	83.2	83.0	82.9	83.1
	Random	84.3	84.5	84.4	84.0	83.0
	Poly	84.0	84.1	84.1	84.2	83.6
	Max	78.6	78.2	75.6	73.7	72.4
	APPNP	82.4	82.1	82.8	82.4	82.5
	DAGNN	84.1	83.3	82.7	80.2	72.1
	SE-aggregation	**84.5**	**85.0**	**84.9**	**85.0**	84.7
	Self-attention	84.4	84.7	84.6	84.5	**84.8**
CiteSeer	Constant	71.4	71.2	71.4	71.4	71.7
	Random	71.4	70.8	71.3	70.2	68.8
	Poly	72.1	72.2	72.2	71.9	71.7
	Max	62.3	59.4	60.1	59.3	58.4
	APPNP	71.0	70.9	70.4	71.1	70.3
	DAGNN	**73.1**	72.5	72.0	70.4	70.1
	SE-aggregation	**73.1**	**72.8**	**72.6**	72.1	71.9
	Self-attention	72.9	72.2	**72.6**	**72.4**	**72.3**
PubMed	Constant	78.5	76.5	74.0	73.0	71.9
	Random	80.1	80.2	80.2	80.1	79.1
	Poly	80.0	80.1	80.2	80.1	80.3
	Max	75.2	75.7	75.4	74.2	73.6
	APPNP	79.4	79.2	79.2	79.5	78.7
	DAGNN	**80.2**	80.4	80.5	80.1	78.3
	SE-aggregation	**80.2**	80.3	80.5	80.5	**80.4**
	Self-attention	80.1	**80.8**	**80.6**	**80.6**	OOM

methods. We implement our model and baselines using deep graph library (DGL) [43]. We use the following first-order methods as baselines: multilayer perceptron (MLP), logistic regression (LR), label propagation (LabelProp), normalized Laplacian label propagation (LabelProp NL) [44], ChebyNet [6], GCN [7], graph attention network (GAT) [26], mixture model network (MoNet) [45], Cluster-GCN [30], VR-GCN [46], and GraphSage [29]. For high-order graph convolution schemes, we use these methods as baselines: APPNP [31], SGC [27], JK-Net-concat [18], IncepGCN [16], [17], MixHop [19], and DAGNN [20]. For a fair comparison, we do not use some specific preprocessing tricks in original articles (e.g. only use the largest connected component). We tune the parameters of each baseline individually to get the results.

3) Training Details: For networks with SE-aggregation, we tune the following hyper-parameters: 1) the number of aggregations $K \in \{4, 5, \ldots, 32\}$; 2) weight decay $\in [0, 1e{-}2]$; 3) dropout rate $\in [0, 0.8]$; 4) hidden dimensional $\in \{32, 64\}$; 5) reduction ratio of SE module $r \in [0.5, 4.0]$; and 6) use projection vector or average pooling in the SE module. On citation networks, we train our model for 400 epochs and apply early stop if the validation loss does not improve in 100 epochs. On coauthorship and copurchase, we train 1000 epochs and apply early stop with 200 epochs. We use a multimodel architecture for the PPI dataset where several high-order graph convolutional networks with SE-aggregation modules are stacked, and their results will be concatenated at the end to obtain the final activation. We use 28 stacked networks with $K = 3$. We train our model on PPI for 8000 epochs and apply early stop if the loss does not improve in 1000 epochs.

B. Accuracy Results

1) Citation Networks: The accuracy and macro F1-score on citation networks are given in Table IV. Following the data split in [41], 20 nodes per class are used for training, 500 nodes are used for validation, and 1000 nodes are used for the test. For each model that we implemented, we conduct ten runs and report the mean and standard deviation. As shown in Table IV, models with SE-aggregation outperform by significant margins. Compared with other high-order graph convolution schemes (JK-Net, MixHop, IncepGCN, and DAGNN), high-order graph convolution with SE-aggregation can achieve much better results on citation networks, which demonstrates the superiority of our method. We also find that networks with MLP before aggregations (e.g. DAGNN and our network) perform better than networks with MLP behind aggregations (e.g. JK-Net, MixHop, and IncepGCN). We conjecture that the MLP may affect the result of the aggregation process and bring the noise to node features after aggregation.

2) Coauthorship and Copurchase Networks: Results on coauthorship and copurchase networks are summarized in Table IV. We follow the setting in [42]. We sample 20 labeled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the test set. As shown in Table IV, models with SE-aggregation outperform other baselines.

3) PPI: The accuracy of the PPI dataset is summarized in Table III. Since not all baselines perform well on this large and inductive dataset, we only choose GraphSage, GAT, and JK-Net as baselines. We also add two methods that are proposed for large graph training, VR-GCN, and Cluster-GCN, as baselines. We directly use results reported in original articles, and these articles do not report the F1-score, so we only compare accuracy here. Our model performs better than all these baselines. Experiments on inductive multigraph datasets also show our model's ability to generalize across graphs.

C. Ablation Study

To verify that the SE-aggregation method can truly help high-order convolutional networks to learn effective combination weights, we conduct ablation studies on three citation network datasets and compare SE-aggregation with other common combination methods used in high-order graph convolutional networks. Also, to further show that SE-aggregation has a better ability to process large regions on graphs with more aggregation results, we conduct ablation studies on various aggregation depths from 16 to 256. We conduct each experiment for ten runs and report the mean result and choose a mean-pooling combination, which simply calculates the channelwise mean value of aggregation results as the baseline. To show the superiority of the SE-aggregation, we compare it with other combination methods used by JK-Net [18], IncepGCN [16], [17], MixHop [19], and DAGNN [20]. Specifically, we choose the following methods for comparison.

1) *Random:* We randomly generate combination weights for all aggregation results and output a weighted sum of these aggregation results.

TABLE VI
RESULTS WITH DIFFERENT TRAINING SET SIZES ON CORA IN TERMS OF CLASSIFICATION ACCURACY (%)

#Training nodes per class	1	2	3	4	5	10	20	30	40	50	100
MLP	30.3	35.0	38.3	40.8	44.7	53.0	59.8	63.0	64.8	65.4	64.0
GCN	34.7	48.9	56.8	62.5	65.3	74.3	79.1	808	82.2	82.9	84.7
GAT	45.3	58.8	66.6	68.4	70.7	77.0	80.8	82.6	83.4	84.0	86.1
APPNP	44.7	58.7	66.3	71.2	74.1	79.0	81.9	83.2	83.8	84.3	85.4
SGC	43.7	59.2	67.2	70.4	71.5	77.5	80.4	81.3	81.9	82.1	83.6
DAGNN	**58.4**	**67.7**	**72.4**	75.5	76.7	80.8	83.7	84.5	85.6	86.0	87.1
SE-aggregation	56.1	65.1	71.5	**75.8**	**76.9**	**82.4**	**83.8**	**85.1**	**85.7**	**86.4**	**88.5**

2) *Poly:* The combination weight for the ith aggregation result is ab^i where a and b are two hyperparameters controlling the importance of low- and high-order aggregation results.
3) *Max:* The max-pooling combination that outputs the maximum value on each channel.
4) *APPNP:* The PageRank [32] style combination scheme used in APPNP [31].
5) *DAGNN:* The combination scheme used in DAGNN [20]. Using the innerproduct of aggregation result and a parameter vector as the combination weight.
6) *Self-Attention:* Using the combination method, as mentioned in Section III-G.

We do not compare the concatenation and LSTM combination methods because these two methods are inefficient in computation and raise an out-of-memory error in most instances.

The results are summarized in Table V. Our method outperforms the baseline and all other combination methods (except our variant method) in all these three datasets. This shows the superiority of our proposed method. In addition, these results also confirm our intuition in Section III-G that the self-attention variant can achieve better performance on large graphs (the PubMed dataset) and deeper networks with more aggregation results.

D. Model Depth

To study how our method is affected by the oversmoothing problem, we conduct experiments for our method on the Cora dataset with different numbers of aggregations. Intuitively, a model without the oversmoothing problem should achieve better performance with a deeper network since it has access to more information from distant nodes. Therefore, a good model design should be able to make better use of information from these neighbors, which means that it can achieve more performance gain from the model depth. We conduct each experiment for ten runs and report the mean result. The reduction ratio of the SE module r is set to be 2, and we apply dropout with a rate of 0.8 for all models. The results are summarized in Fig. 5. As shown in the results, GCN is affected severely by the oversmoothing problem, and oversmoothing has a slight impact on SGC and MixHop. On the contrary, APPNP, JK-Net, IncepGCN, DAGNN, and our SE-aggregation are not affected by the oversmoothing problem in this aggregation number region. Moreover, among models that are not affected by oversmoothing, SE-aggregation achieves the best performance as the network goes deeper. Particularly, SE-aggregation can have an accuracy of 85.2% with depth equals 32, which is higher than the accuracy of all other models no matter the networks are deep or shallow. Although some models can have better performance than SE aggregation when the depth is shallow, these experimental results still show the superiority of the design of our SE aggregation comparing with other high-order methods since SE-aggregation can achieve more performance gain from the model depth. As discussed in Section III-E, high-order graph convolution schemes that do not consider individual combination methods for each node (e.g. JK-Net, MixHop, and IncepGCN) or do not properly consider the coupling effects between representations from different distances (e.g. DAGNN) perform worse when the network goes deeper. By contrast, our model with SE-aggregation considers node-individual combination methods and the coupling effects between representations from different distances simultaneously, thereby benefiting from the depth. Finally, the oversmoothing problem has a slight impact on MixHop, while our method is not affected by the oversmoothing problem within the region of hyperparameters used in our experiments.

TABLE VII
RESULTS WITH A DIFFERENT REDUCTION RATIO r OF THE SE MODULE ON CITATION NETWORKS IN TERMS OF CLASSIFICATION ACCURACY (%)

r	0.5	1.0	2.0	3.0	4.0	Best r
Cora	83.8	84.6	84.6	84.4	84.0	r=2.21, acc=84.7
CiteSeer	72.0	72.1	72.7	72.7	73.1	r=2.38, acc=73.3
PubMed	75.9	79.8	79.8	79.6	79.6	r=1.52, acc=80.2

We also conduct experiments on the PPI dataset with different numbers of stacked networks and different numbers of aggregations per network. The results are given in Fig. 6. It shows that deeper models can achieve better performance when combined with our SE-aggregation method.

E. Training Set Size

The size of the training set is important in semisupervised learning tasks. To explore how our model performs with different training set sizes, we conduct experiments with different training set sizes and compare results with several representative baselines. We follow the setting in [20] where ten aggregations are used in high-order GCNs. We directly

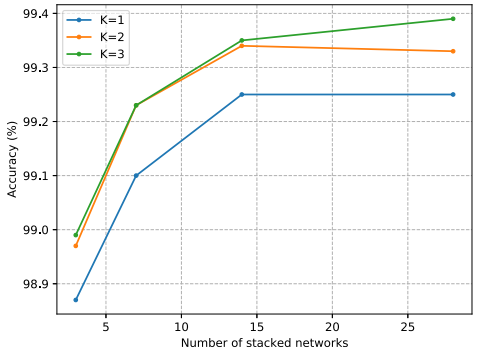


Fig. 6. Results on the PPI dataset using our SE-aggregation model with different numbers of stacked networks and numbers of aggregations per network (K) in terms of classification accuracy (%).

use the result reported in [20] for baselines. For each model that we implemented, we conduct 100 runs on random training/validation/test splits for every training set size on the Cora dataset. The results are summarized in Table VI. Our model achieves the best performance when the number of nodes in the training set per class is greater than 3. The reason why our model performs worse on small training set size may be that our method is more complex than other high-order graph convolution schemes and requires more training data.

F. Reduction Ratio of SE Module

Similar to the SENet [24], we use a ratio parameter r to control model complexity. To explore how this parameter affects our model, we conduct experiments with different r on citation networks. Specifically, we conduct each experiment for ten runs and report the mean result. For each model, we use 16 aggregations and set the dimension of hidden states as 64. In addition, to find the optimal value of r, we conduct a hyperparameter search via the Bayesian optimization with the domain of r equal to $[0.5, 4.0]$. The results are summarized in Table VII. Experimental results show that, by tuning r, we can obtain a proper model complexity, and it aids generalization on testset with a smaller structural risk.

V. Conclusion

In this article, we perform a theoretical analysis on high-order graph convolution to explore the mechanism behind these methods. Based on the theoretical analysis, we point out two weaknesses of existing high-order graph convolution methods: 1) lack of mechanisms to generate individual combination methods for each node and 2) lack of proper modeling of the relationship between representations from different distances.

To solve these problems, we propose a new adaptive combination method, coined SE aggregation, for high-order graph convolution. Our new combination method can learn to generate a unique combination method for each node in graphs. The SE module also helps networks to model the interaggregation relationship properly. Extensive experiments on node classification tasks show the reasonability of our analysis and the superiority of our proposed method compared with other high-order graph convolution methods.

Further improvements of our method are possible as future works. For example, self-supervised learning with joint training can be adopted to modify the training loss of our model and give our method the ability to fully exploit the unlabeled nodes. In addition, we can use methods such as selective kernel [47] to improve the SE module used in our model. Moreover, efficient training methods on super large-scale graphs can also be explored.

Acknowledgment

The authors would like to thank the anonymous reviewers and editors for the valuable comments.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.
[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.
[3] 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, Jan. 2009.
[4] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–20.
[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.
[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.
[8] 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.
[9] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4509–4517.
[10] P. Riba, A. Fischer, J. Llados, and A. Fornes, "Learning graph distances with message passing neural networks," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 2239–2244.
[11] Z. Li and J. Tang, "Weakly supervised deep matrix factorization for social image understanding," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 276–288, Jan. 2017.
[12] Z. Li, J. Tang, and T. Mei, "Deep collaborative embedding for social image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2070–2083, Sep. 2019.
[13] Z. Li, J. Tang, L. Zhang, and J. Yang, "Weakly-supervised semantic guided hashing for social image retrieval," *Int. J. Comput. Vis.*, vol. 128, nos. 8–9, pp. 2265–2278, Sep. 2020.
[14] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.
[15] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-GCN: Multi-scale graph convolution for semi-supervised node classification," in *Proc. Conf. Uncertainty Artif. Intell. (UAI)*, 2019, p. 310.
[16] A. Kazi *et al.*, "Inceptiongcn: Receptive field aware graph convolutional network for disease prediction," in *Proc. Int. Conf. Inf. Process. Med. Imag.*, 2019, pp. 73–85.
[17] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–18.
[18] 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. 5449–5458.
[19] S. Abu-El-Haija *et al.*, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 21–29.

[20] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 338–348.
[21] B. Y. Weisfeiler and A. A. Leman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Tech. Inf.*, vol. 2 no. 9, pp. 2–16, 1968.
[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[23] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[25] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–12.
[27] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.
[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–17.
[29] 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.
[30] 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.
[31] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–15.
[32] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep. 1999-66, 1999.
[33] 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.
[34] L. Babai and L. Kucera, "Canonical labelling of graphs in linear average time," in *Proc. 20th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1979, pp. 39–46.
[35] J.-Y. Cai, M. Fürer, and N. Immerman, "An optimal lower bound on the number of variables for graph identification," *Combinatorica*, vol. 12, no. 4, pp. 389–410, Dec. 1992.
[36] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.
[37] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.
[38] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI, USA: American Mathematical Society, 1996.
[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[40] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[41] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 40–48.
[42] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2019, *arXiv:1811.05868*. [Online]. Available: https://arxiv.org/abs/1811.05868
[43] M. Wang *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2020, *arXiv:1909.01315*. [Online]. Available: https://arxiv.org/abs/1909.01315
[44] A. Z. O. Chapelle and B. Schölkopf, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.
[45] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5425–5434.
[46] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 941–949.
[47] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 510–519.

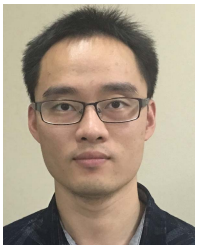
Tianqi Zhang received the B.S. degree (Hons.) in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2019, where he is currently pursuing the master's degree with the Department of Computer Science and Engineering.

He worked as a Research Intern with AWS AI Laboratory, Shanghai, and also a Research Intern with Microsoft Research Asia, Beijing, China. His current research interests include graph neural networks and ML system development.

Qitian Wu received the B.S. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, in 2018 and 2021, respectively.

He has first-authored several articles in the top-tier conferences, such as the International Conference on Machine Learning (ICML), the Conference on Neural Information Processing Systems (NeurIPS), the ACM Knowledge Discovery and Data Mining (KDD), the International World Wide Web Conferences (WWW), the International Joint Conference on Artificial Intelligence (IJCAI), the ACM International Conference on Information and Knowledge Management (CIKM), and the International Conference on Data Mining (ICDM). His research interests include machine learning and data mining, especially representation learning on graph-structured and user behavioral data, as well as developing label-efficient models for applications such as recommender systems and social network analysis.

He also served as a Reviewer for ICML, NeurIPS, the AAAI Conference on Artificial Intelligence (AAAI), and IJCAI.



Junchi Yan (Senior Member, IEEE) received the Ph.D. degree from the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2015.

He was a Senior Research Staff Member and a Principal Scientist with IBM Research, Shanghai, where he started his career in April 2011. He is currently an Associate Professor with Shanghai Jiao Tong University. His research interest is machine learning, especially for temporal and graph data.

Dr. Yan received the ACM China Doctoral Dissertation Nomination Award and the China Computer Federation Doctoral Dissertation Award. He serves as an Associate Editor for IEEE ACCESS and a Managing Guest Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), *Pattern Recognition*, and *Pattern Recognition Letters*. He also served as an Area Chair for ICPR 2020, ACM MM 2021, CVPR 2021, and the AAAI Conference on Artificial Intelligence (AAAI) 2022, and a Senior PC for the International Joint Conference on Artificial Intelligence (IJCAI) 2021 and the ACM International Conference on Information and Knowledge Management (CIKM) 2019.