



MulStepNET: stronger multi-step graph convolutional networks via multi-power adjacency matrix combination

Xun Liu¹ · Fangyuan Lei^{2,3} · Guoqing Xia¹

Received: 3 September 2020 / Accepted: 22 June 2021 / Published online: 6 August 2021
© The Author(s) 2021

Abstract

Graph convolutional networks (GCNs) have become the de facto approaches and achieved state-of-the-art results for circumventing many real-world problems on graph-structured data. However, these networks are usually shallow due to the over-smoothing of GCNs with many layers, which limits the expressive power of learning graph representations. The current methods of solving the limitations have the bottlenecks of high complexity and many parameters. Although Simple Graph Convolution (SGC) reduces the complexity and parameters, it fails to distinguish the feature information of neighboring nodes at different distances. To tackle the limits, we propose MulStepNET, a stronger multi-step graph convolutional network architecture, that can capture more global information, by simultaneously combining multi-step neighborhoods information. When compared to existing methods such as GCN and MixHop, MulStepNET aggregates neighborhoods information at more distant distances via multi-power adjacency matrix while fitting fewest parameters and being computationally more efficient. Experiments on citation networks including Pubmed, Cora, and Citeseer demonstrate that the proposed MulStepNET model improves over SGC by 2.8, 3.3, and 2.1% respectively while keeping similar stability, and achieves better performance in terms of accuracy and stability compared to other baselines.

Keywords Graph convolutional networks · High complexity · Simple Graph Convolution · Multi-power adjacency matrix · Multi-step neighborhoods information · Multi-step graph convolutional network

1 Introduction

Graph convolutional networks (GCNs) (Zhang et al. 2018b; Kipf and Welling 2017; Li et al. 2018; Yao et al. 2019) show natural advantages for dealing with many real-world problems which can be modeled as graph networks (Bruna et al. 2014; Hamilton et al. 2017; Monti et al. 2017; Defferrard

et al. 2016). Each convolution in these GCNs is based on one-step neighborhood aggregation scheme. GCNs directly apply multiple graph convolution layers to obtain multi-step neighborhoods information and learn graph representations. Limited to the over-smoothing (Li et al. 2018) of GCNs with more layers, GCNs are difficult to leverage the hierarchical property of convolutional neural networks (CNNs) such as AlexNet (Krizhevsky et al. 2012) and MACNN (Lai et al. 2020). Therefore, GCNs are usually shallow and generally do not exceed four layers (Zhou et al. 2018). For instance, there are only two layers in GCN (Kipf and Welling 2017). This shallow mechanism limits the expressive power of learning graph representations and label propagation (Sun et al. 2020). To address the limitations, researchers propose some approaches which are summarized in the following two-fold.

1. To improve the expressive power, Li et al. (2019b) propose DeepGCNs with very deep networks using deep CNNs concepts such as residual connections (He et al. 2016) and dense connections (Huang et al. 2017). Nev-

✉ Fangyuan Lei
leify@126.com

Xun Liu
liuxun.stf@gmail.com

Guoqing Xia
xgq@sise.com.cn

¹ Department of Electronics, Software Engineering Institute of Guangzhou, Guangzhou 510990, China

² School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou 510665, China

³ Guangdong Provincial Key Laboratory of Intellectual Property Big Data, Guangdong Polytechnic Normal University, Guangzhou 510665, China

ertheless, deep networks with many parameters are extremely hard to train on large graphs.

- High-order graph convolution models (Abu-El-Haija et al. 2019a, b; Luan et al. 2019) such as MixHop (Abu-El-Haija et al. 2019b) directly capture the interaction of neighboring nodes at different distances to achieve performance improvement. As the number of order increases, the parameters of these models will increase and these models become more complex. This is hard to train and may suffer from overfitting.

In order to reduce excess complexity and parameters, a recent surge of interest has focused on Simple Graph Convolution (SGC) (Wu et al. 2019). Wu et al. (2019) have shown that by repeatedly removing the nonlinearities between graph convolution layers of GCN and collapsing normalized adjacency matrices between consecutive layers, the complexity of GCN is reduced. In addition, they significantly reduce parameters through reparameterizing weight matrices into a single weight matrix. Nevertheless, SGC has difficulty in distinguishing the feature information of neighboring nodes at various distances, which limits the expressive power.

To address the limits, we propose a novel architecture of stronger multi-step graph convolutional network (MulStepNET). Figure 1 shows the background and need for designing our MulStepNET. As illustrated in Fig. 2, our MulStepNET leverages multi-step neighborhoods information by constructing multi-power adjacency matrix with simple grouping and attention mechanism and applies the attention mechanism to flexibly adjust the contributions of neighboring nodes at various distances. Based on the multi-power adjacency matrix, we design a stronger multi-step graph convolution to aggregate more nodes features and learn global graph structure. Further, we build an one-layer network model with the fewest parameters to avoid overfitting and reduce complexity. Meanwhile, the model with large k -steps (zero-step to k -step) graph convolution widens the receptive field and improves the learning ability. Our contributions are as follows:

- To enlarge the receptive field, we construct a multi-power adjacency matrix with simple grouping and attention mechanism by combining the adjacency matrices of different powers. Based on the multi-power adjacency matrix, we develop a new multi-step graph convolution that can flexibly adjust the weights of neighboring nodes at different distances. This may improve the learning ability.
- To the best of our knowledge, it is the first work to propose an one-layer architecture with larger steps graph convolution. Compared with prior methods, in terms of complexity and parameters, our architecture performs as

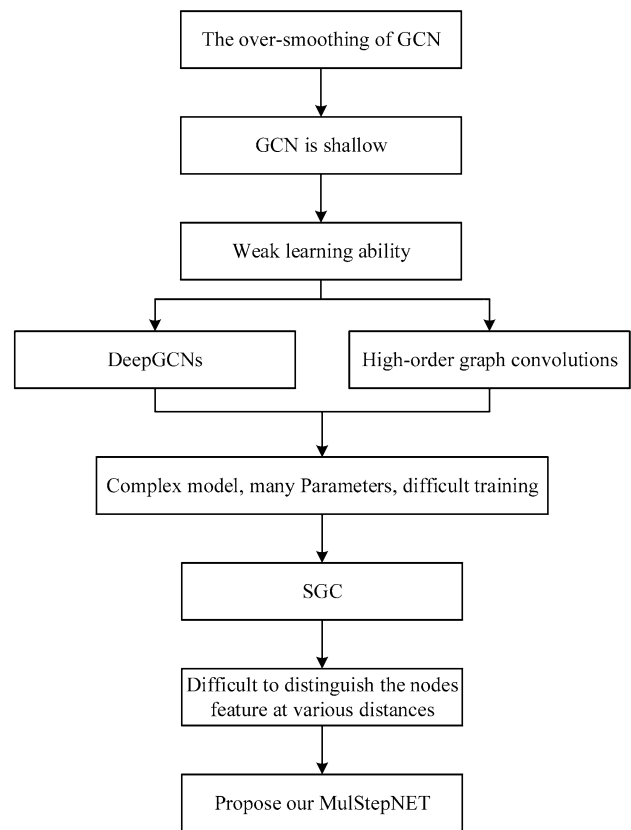


Fig. 1 Our MulStepNET background

well as SGC and outperforms other methods while capturing more nodes and global information.

- We conduct extensive experiments on node classification tasks. Experimental results show that the proposed method compares favorably against state-of-the-art approaches in terms of classification performance and stability.

2 Preliminaries and related works

Graph convolutional networks are successfully applied to non-Euclidean and Euclidean applications and are quickly evolving (Wang and Ye 2018; Yao et al. 2019; Kampffmeyer et al. 2019; Chen et al. 2018; Guo et al. 2019; Yu and Qin 2020). We mainly review the work most relevant to our approach.

An undirected graph \mathcal{G} with n vertices and e edges is denoted as $\mathcal{G} = (\epsilon, \nu, A)$, where ϵ and ν are respectively the set of edges and vertices. The edge relationships of \mathcal{G} can be described by adjacency matrix $A \in \mathbb{R}^{n \times n}$. We introduce $X \in \mathbb{R}^{n \times c_0}$ to denote node feature matrix with c_0 features per node.

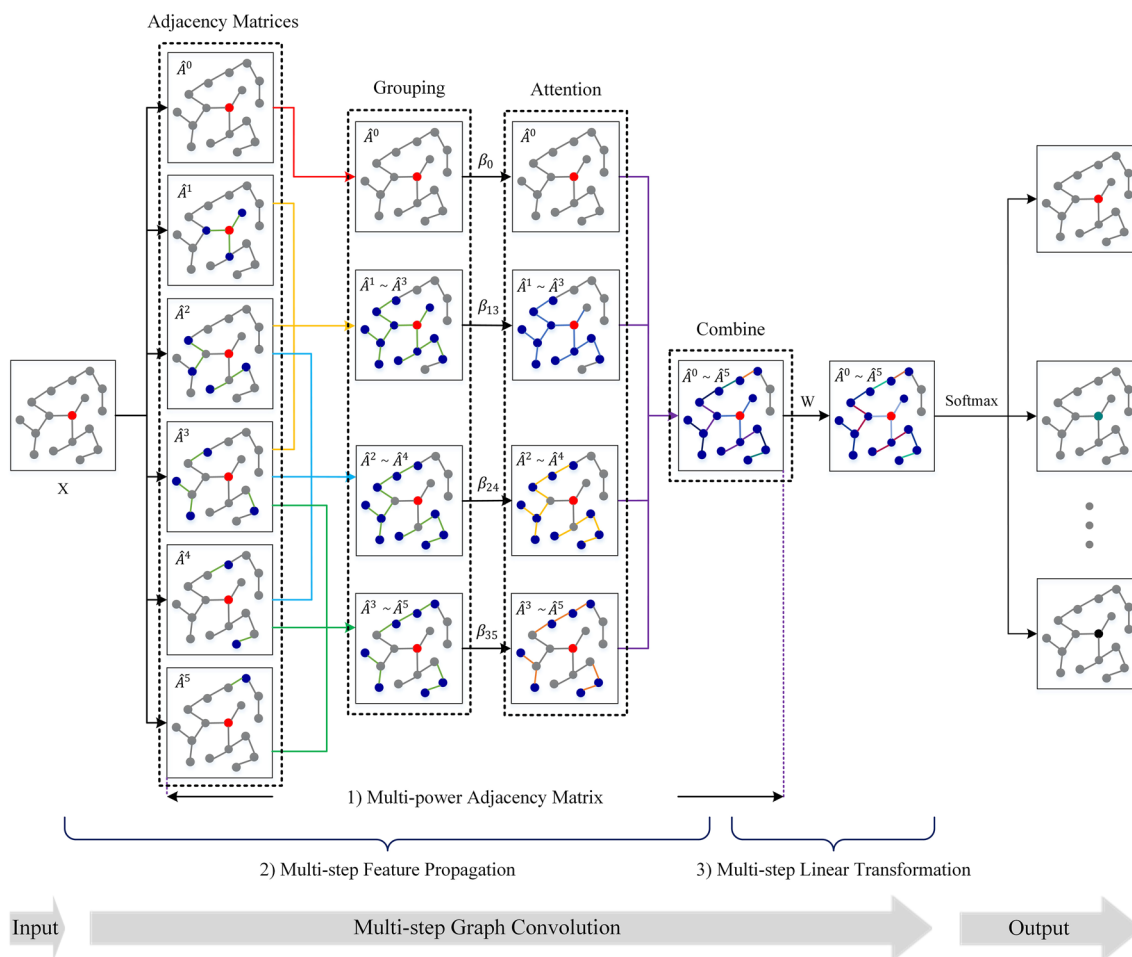


Fig. 2 Our MulStepNET architecture. This figure describes the architecture when k is 5. Our MulStepNET consists of an input layer, a multi-step graph convolution layer, and an output layer. The multi-

step graph convolution includes three stages: multi-power adjacency matrix, multi-step feature propagation, and multi-step linear transformation

Similar to CNNs, the convolution of GCN (Kipf and Welling 2017) is to learn the feature representation of nodes over multiple graph convolution layers. At layer j , we denote the adjacency matrix A_j and the node hidden representation H_j as input, the output node representation H_{j+1} can be written as:

$$H_{j+1} = \text{ReLU} \left(\tilde{D}_j^{-\frac{1}{2}} \tilde{A}_j \tilde{D}_j^{-\frac{1}{2}} H_j W_j \right), \tag{1}$$

where \tilde{A}_j denotes a new adjacency matrix with self-loops, with $\tilde{A}_j = A_j + I_j$. I_j and \tilde{D}_j are identity matrix and degree matrix respectively. We describe H_1 ($j=1$) as original input feature matrix X and use gradient descent to train weight matrix W_j . Stacking the layer twice, the two-layer GCNs can be described as:

$$Y_{\text{GCN}} = \text{softmax} \left(\tilde{D}_j^{-\frac{1}{2}} \tilde{A}_j \tilde{D}_j^{-\frac{1}{2}} \text{ReLU} \left(\tilde{D}_j^{-\frac{1}{2}} \tilde{A}_j \tilde{D}_j^{-\frac{1}{2}} X W_1 \right) W_2 \right), \tag{2}$$

where W_1 and W_2 are different weight matrices, softmax is a normalization classifier.

Abu-El-Haija et al. (2019b) propose high-order graph convolution (HGC) model to improve expressive power by mixing multi-hop neighborhoods information. We normalize the adjacency matrix \tilde{A}_j into normalized adjacency matrix \hat{A}_j , with $\hat{A}_j = \tilde{D}_j^{-\frac{1}{2}} \tilde{A}_j \tilde{D}_j^{-\frac{1}{2}}$. Then the model is as follow:

$$H_{j+1} = \text{ReLU}(\hat{A}_j^1 H_j W_1 | \hat{A}_j^2 H_j W_2 | \dots | \hat{A}_j^k H_j W_j), \tag{3}$$

where \hat{A}_j^k is the k power of \hat{A}_j , and $|$ denotes column concatenation. Lei et al. (2020) develop HGC model and reduce the parameters of Abu-El-Haija et al. (2019b) via weight sharing mechanism.

We write a K -layer GCN in general form as:

$$\hat{Y} = \text{softmax}(\hat{A}_j \text{ReLU}(\dots \hat{A}_j \text{ReLU}(\hat{A}_j X W_1) W_2 \dots) W_K). \tag{4}$$

According to the hypothesis of Wu et al. (2019), for the K -layer GCN, we remove all ReLU functions and reparameterize all weight matrices (W_1, W_2, \dots, W_K) into a single weight matrix W via $W = W_1 W_2 \dots W_K$. The K -layer GCN becomes:

$$\hat{Y} = \text{softmax}(\hat{A}_j^K XW), \quad (5)$$

where \hat{A}_j^K denotes K power of \hat{A}_j . The model is called as Simple Graph Convolution (SGC) (Wu et al. 2019). Although the model has fewer computations and parameters, the model can not distinguish the features information of neighboring nodes at different distances. This restricts the ability of learning graph representations. There are many other simple or linear models (Thekumparampil et al. 2018; Cai and Wang 2018; Eliav and Edith 2018). By applying these models to the tasks of Li et al. (2019a) and Al-Sharif et al. (2020), these models are more powerful.

Recently, many graph attention models (Veličković et al. 2018; Thekumparampil et al. 2018; Zhang et al. 2018a) try to assign suitable weights based on node features in the graph and achieve better performance on graph learning tasks. Nevertheless, these models with attention mechanism bring the concerns of high complexity and nuisance parameters. There are many other researches (Zhou et al. 2018; Wu et al. 2020) for comprehensive review.

3 The proposed method

We are committed to developing a method that can simultaneously capture neighboring nodes information at different distances and global graph structure while fitting few parameters and being computationally efficient. In this section, we propose a novel one-layer MulStepNET architecture. Further, we introduce our multi-step graph convolution that can simultaneously capture neighboring nodes information at more distant distances and analyze the computational complexity and parameters.

3.1 The overall architecture

In CNNs, we increase the expressivity of extracting features via many and deeper convolutional layers, which enlarges the scale of receptive field. However, GCN with multiple layers (exceed 2 layers) can suffer from over-smoothing (Li et al. 2018), which hurts classification performance on graph learning tasks. To improve the performance, an effective way is to use high-order graph convolutions with different weight matrices to gather more neighborhoods information at various distances (Abu-El-Haija et al. 2019b). As the number of order increases, the parameters will significantly increase and the model will lead to redundant computation. This is

hard to train and brings potential concerns of overfitting. For SGC (Wu et al. 2019), although the simplified mechanism speeds up model training and avoids the excess complexity, SGC fails to adjust and distinguish the contributions of neighboring nodes at different distances. This limits the learning ability. Motivated by the above analyses, we propose our MulStepNET architecture (Fig. 2). A key innovation of MulStepNET is a novel multi-step graph convolution layer. Instead of aggregating the information of one-step neighboring nodes, our multi-step graph convolution combines these neighborhoods information at different distances and captures the high-order interaction between nodes.

We summarize the main differences of our MulStepNET and the models most relevant to our approach as follows. In SGC, the model applies the K power of the normalized adjacency matrix \hat{A} to obtain the neighboring nodes that are K -hops away. In our MulStepNET, we design a multi-power adjacency matrix (the k powers of the \hat{A} , namely 0 power to k power of the \hat{A} , $k > K$, see Sect. 3.2) to obtain the neighboring nodes that are k -hops away. In addition, we use a new attention mechanism to distinguish the contributions that may be important for classification. In MixHop, (1) the model constructs high-order graph convolution using different weight matrices to capture the feature information of neighboring nodes at different distances; (2) the model uses column concatenation to combine these information; (3) the model has a two-layer structure. In our MulStepNET, (1) we propose the multi-step graph convolution based on single weight matrix to capture the feature information of neighboring nodes at more distant distances; (2) we consider these information by the multi-power adjacency matrix, rather than the column concatenation; (3) we construct an one-layer structure with fewer computations and parameters; (4) we can adjust the contributions via the attention mechanism.

3.2 Multi-step graph convolution layer

Multi-step graph convolution has three stages: multi-power adjacency matrix, multi-step feature propagation, and multi-step linear transformation. We introduce each stage in detail as follows.

Multi-power adjacency matrix We use adjacency matrix A to describe the edge weights between nodes that are one-step away in the graph. The adjacency matrix A fails to denote the edge weights of k -step ($k > 1$) neighboring nodes. The information propagation of graph is propagated among edges in the graph. We introduce A^k (the k power of A) to describe the k -step edge weights which indicate the relationship of k -step neighboring nodes in the graph. In order to better capture more nodes and global graph information, we combine different powers of A into a single multi-power adjacency matrix \hat{A}_{ok} .

$$\hat{A}_{ok} = \beta_0 \hat{A}^0 + \beta_{13}(\hat{A}^1 + \hat{A}^2 + \hat{A}^3) + \beta_{24}(\hat{A}^2 + \hat{A}^3 + \hat{A}^4) + \dots + \beta_{(k-2)k}(\hat{A}^{k-2} + \hat{A}^{k-1} + \hat{A}^k), \tag{6}$$

where \hat{A}_{ok} denotes the multi-power adjacency matrix. \hat{A}^1 is normalized adjacency matrix, with $\hat{A}^1 = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} = \tilde{D}^{-\frac{1}{2}} (A + I) \tilde{D}^{-\frac{1}{2}}$. Here I and \tilde{D} denote identity matrix and the degree matrix of \tilde{A} , with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. We provide a higher weight to nodes's own features via \hat{A}^0 because the own features may be more important, with $\hat{A}^0 = I$. Furthermore, we utilize $\hat{A}^2, \hat{A}^3, \dots, \hat{A}^k$ to obtain two-step and larger step neighbors information in the graph. In order to consider more similarity of adjacent powers of \hat{A} and distinguish the difference between other powers of \hat{A} , we divide different powers of \hat{A} as multiple simple groups including $\{\hat{A}^0\}$, $\{\hat{A}^1, \hat{A}^2, \hat{A}^3\}$, $\{\hat{A}^2, \hat{A}^3, \hat{A}^4\}$, ..., $\{\hat{A}^{k-2}, \hat{A}^{k-1}, \hat{A}^k\}$ and flexibly adjust the weights of these groups via a series of attention multipliers $\beta_0, \beta_{13}, \beta_{24}, \dots, \beta_{(k-2)k} \in R$. By adjusting the weights, we can adjust the contributions of neighboring nodes at different distances. Each group except \hat{A}^0 leverages and shares the information of same groups. This can be regarded as an attention mechanism with simple grouping. We combine all groups with these attention multipliers to obtain much information and learn global graph topology.

Theorem 1 *Multi-power adjacency matrix is an operator of preserving graph topology.*

Proof Let $\hat{A}^0 \in R^{n \times n}$ (n nodes), then $\hat{A}^0 = I \in R^{n \times n}, \hat{A}^1, \hat{A}^2, \dots, \hat{A}^k \in R^{n \times n}$, and $\hat{A}_{ok} = \beta_0 \hat{A}^0 + \beta_{13}(\hat{A}^1 + \hat{A}^2 + \hat{A}^3) + \beta_{24}(\hat{A}^2 + \hat{A}^3 + \hat{A}^4) + \dots + \beta_{(k-2)k}(\hat{A}^{k-2} + \hat{A}^{k-1} + \hat{A}^k) \in R^{n \times n}$. Equation (6) shows that multi-power adjacency matrix is element-wise operation. Obviously, the spatial location of \hat{A}_{ok} is the same as \hat{A} , thus preserving the topology of the graph.

Equation (6) and Theorem 1 show that our multi-power adjacency matrix can capture more nodes and global graph information while preserving the graph topology.

Multi-step feature propagation Given the node feature matrix X , the feature propagation of graph convolution in GCN is defined as follows:

$$H_1 = \hat{A}X. \tag{7}$$

Equation (7) shows that the feature propagation propagates node information to one-step neighboring nodes as well as the node itself via \hat{A} . However, the feature propagation fails to obtain two-step and larger step neighboring nodes information. To circumvent the limits, we design a novel multi-step feature propagation scheme in Eq. (8).

$$H_{ok} = \hat{A}_{ok}X. \tag{8}$$

Instead of propagating one-step neighboring nodes information, our multi-step feature propagation can capture the information of k -steps nodes while keeping larger receptive field.

Multi-step linear transformation After the multi-step feature propagation, a multi-step linear transformation is applied to the H_{ok} by $H_{ok}W$.

$$Y_{ok} = H_{ok}W, \tag{9}$$

where W denotes weight matrix which is shared among all nodes.

Based on Eqs. (8) and (9), we conclude that our multi-step graph convolution takes the following form:

$$Y_{ok} = \hat{A}_{ok}XW. \tag{10}$$

Our algorithm is summarized in Algorithm 1. Existing convolutions generally only aggregate neighborhoods information at up to four-steps distances, however our convolution with large k -steps can simultaneously capture neighborhoods information at more distant distances. Specifically, compared with GCN's convolution that can only capture one-step neighboring nodes information, our multi-step graph convolution can capture the high-order interaction information between neighboring nodes. Compared with SGC's convolution that can not distinguish the features information of neighboring nodes at various distances, our multi-step graph convolution can adjust the contributions of these features information and obtain more nodes information and better learn global graph topology.

Algorithm 1 Multi-step graph convolution

Input: \hat{A}, X
Parameter: W
Output: Y_{ok}
 1: Let $i = 0$.
 2: **while** $i \leq k$ **do**
 3: $\hat{A}^i = \hat{A}^i$.
 4: $i += 1$.
 5: **end while**
 6: Calculate \hat{A}_{ok} according to Eq. (6).
 7: $H_{ok} = \hat{A}_{ok}X$.
 8: $Y_{ok} = \hat{H}_{ok}W = \hat{A}_{ok}XW$.
 9: **return** Y_{ok}

3.3 Output layer

Similar to GCN, we predict the label of nodes using a softmax classifier. The output prediction $Y_{MulStepNET}$ can be expressed as:

$$Y_{MulStepNET} = \text{softmax}(\text{ReLU}(Y_{ok})). \tag{11}$$

We follow the loss function from Kipf and Welling (2017).

3.4 Analysis of complexity and parameters

Since the calculation of H_{ok} requires no weight, we calculate $H_{ok} = \hat{A}_{ok}X$ in a feature preprocessing step. That is, we can regard H_{ok} as a fixed feature extractor, then our calculation is very efficient. As described in Sect. 3.2, if $\hat{A} \in \mathbb{R}^{n \times n}$, $X \in \mathbb{R}^{n \times c_0}$, $W \in \mathbb{R}^{c_0 \times c_1}$ (c_1 filters). Then $\hat{A}_{ok} \in \mathbb{R}^{n \times n}$, $H_{ok} \in \mathbb{R}^{n \times c_0}$, $Y_{ok} = H_{ok}W \in \mathbb{R}^{n \times c_1}$. In our model, c_1 is the number of classes. The proposed MulStepNET architecture takes $O(n \times c_0 \times c_1)$ computational time and $O(c_0 \times c_1)$ parameters. The complexity and parameters are the same as SGC. Obviously, our MulStepNET takes fewer computations and parameters due to without hidden layer, when compared to GCN. To the best of our knowledge, our MulStepNET and SGC achieve the best performance in terms of computational time and parameters.

4 Experiments

In this section, we perform experiments on citation network datasets to evaluate the performance of our MulStepNET in terms of prediction accuracy and stability. We compare our MulStepNET against recent state-of-the-art approaches including graph networks and high-order graph convolutions in terms of classification accuracy, complexity, and parameters on benchmark datasets. Further, we conduct experiments using different k to investigate the relationship between model width and prediction accuracy. In addition, we prove that own node features are more important for prediction. Lastly, we investigate the influence of attention multipliers and show the significant trends of the proposed method.

4.1 Datasets

To prove the expressive power of the proposed MulStepNET to learn global graph topology, we evaluate on citation network datasets chosen from benchmarks commonly used on semi-supervised node classification tasks. The datasets are: Pubmed, Cora, and Citeseer (Kipf and Welling 2017). Following Kipf and Welling (2017) and Yang et al. (2016), we summarize the statistics of the datasets as shown in Table 1.

4.2 Baselines and experimental setting

In the performance comparison on semi-supervised node classification tasks, we consider baselines based on graph networks as well as recent high-order graph convolution methods. These graph networks are: Diffusion Convolutional Neural Networks (DCNN) (Atwood and Towsley

Table 1 Dataset statistics

Dataset	Classes	Nodes	Features	Edges	Label rate
Pubmed	3	19,717	500	44,338	0.003
Cora	7	2708	1433	5429	0.052
Citeseer	6	3327	3703	4732	0.036

2016), Gated Graph Neural Networks (GGNN) (Li et al. 2015), Graph Convolutional Networks with Chebyshev (ChebNet) (Defferrard et al. 2016), Message Passing Neural Networks (MPNN) (Gilmer et al. 2017), Graph Convolutional Network (GCN) (Kipf and Welling 2017), and Graph Attention Networks (GAT) (Veličković et al. 2018). These high-order graph convolution methods include Adaptive Lanczos Network (AdaLNet) (Liao et al. 2019), Learned MixHop (MixHop-learn) (Abu-El-Haija et al. 2019b), Lanczos Network (LNet) (Liao et al. 2019), and Simple Graph Convolution (SGC) (Wu et al. 2019).

We run all experiments using Adam optimizer with the learning rate. We now report the hyperparameters that are optimized repeatedly. Instead of using dropout in GCN, we remove dropout and set learning rate to 0.01 for Pubmed and Citeseer. To improve stability and prediction accuracy, the learning rate and dropout are set to 0.0003 and 0.95 on Cora. Besides, we apply 0.005, 0.0005, 0.0008 L2 regularization factors to avoid over-fitting on Pubmed, Cora, and Citeseer. We train 290, 43,000, 2000 epochs on Pubmed, Cora, Citeseer, respectively. We fine-tune the highest power (k power) of the adjacency matrix \hat{A} to achieve better prediction performance on all datasets. We set k to 21, k to 8, k to 4 on Pubmed, Cora, and Citeseer, respectively. In our MulStepNET, we set these attention multipliers $\beta_0, \beta_{13}, \beta_{24}, \dots, \beta_{(k-2)k}$ to suitable values, with $\beta_0 = 1.5$, $\beta_{13} = 0.8$, $\beta_{24} = 0.09$, $\beta_{35} = 0.2$, $\beta_{46} = 0.2$, $\beta_{79} = 0.4$, $\beta_{11-13} = 0.25$, $\beta_{12-14} = 0.25$, $\beta_{13-15} = 0.2$, $\beta_{14-16} = 0.2$, $\beta_{15-17} = 0.5$, other $\beta = 0.3$ on Pubmed, with $\beta_0 = 1.5$, $\beta_{13} = 1.7$, $\beta_{57} = 0.6$, $\beta_{68} = 0.4$, other $\beta = 0.3$ on Cora, with $\beta_0 = 0.8$, $\beta_{13} = 0.925$, $\beta_{24} = 0.5$ on Citeseer. With the above parameter settings, early stop is not adopted in all experiments due to very stable experimental results.

4.3 Results analysis

Based on Eqs. (12) and (13), we obtain the average test accuracy \overline{acc} and standard deviation S of the proposed models.

$$\overline{acc} = \frac{\sum_{i=1}^m acc_i}{m}, \quad (12)$$

where \overline{acc} is the test accuracy of the i th run, and m is the number of runs.

$$S = \sqrt{\frac{\sum_{i=1}^m (acc_i - \overline{acc})^2}{m - 1}}. \tag{13}$$

We compare our MulStepNET against graph networks and high-order graph convolution methods on citation networks, and the results are summarized in Table 2. Based on the results, we observe that our MulStepNET achieves the best performance including classification accuracy and stability among the state-of-the-art approaches (except SGC) on all datasets. Our MulStepNET obtains the highest prediction accuracy of 81.1, 83.7, 73.4% and very low standard deviation of 0.0, 0.1, 0.0% on Pubmed, Cora, and Citeseer respectively. In terms of accuracy comparison on all datasets, our MulStepNET improves over GCN by 2.7, 2.7, and 4.4%, improves over GAT by 2.7, 0.8, and 1.2%, and improves over SGC by 2.8, 3.3, and 2.1% respectively. Compared with SGC, our MulStepNET achieves competitive

performance in terms of stability while significantly improving accuracy. Due to the smaller standard deviation in the proposed model, our MulStepNET outperforms other baselines by a large margin in terms of stability. These results demonstrate the effectiveness of MulStepNET for capturing nodes information and global graph topology.

Similar to He and Sun (2015), we use theoretical time complexity to describe the complexity, rather than the actual running time, since the actual running time is sensitive to hardware and implementations. Table 3 shows the results of the proposed MulStepNET and competing methods in terms of complexity and parameters (see Sect. 3.4 for analysis). In terms of complexity and parameters, our MulStepNET achieves as good performance as SGC, and consistently outperforms other methods. These results show the superiority for designing one-layer model.

To investigate the influence of model width and \hat{A}^0 , we conduct these experiments using different k for our MulStepNET-a and MulStepNET-b on Pubmed, Cora, and Citeseer datasets respectively. The results are summarized in Fig. 3. We observe that our MulStepNET-a can improve

Table 2 Comparison with graph networks and high-order graph convolution methods on citation networks (in percent)

Method	Pubmed	Cora	Citeseer
DCNN (Atwood and Towsley 2016)	76.8 ± 0.8	79.7 ± 0.8	69.4 ± 1.3
MPNN (Gilmer et al. 2017)	75.6 ± 1.0	78.0 ± 1.1	64.0 ± 1.9
GGNN (Li et al. 2015)	75.8 ± 0.9	77.6 ± 1.7	64.6 ± 1.3
ChebNet (Defferrard et al. 2016)	69.8 ± 1.1	78.0 ± 1.2	70.1 ± 0.8
GCN* (Kipf and Welling 2017)	79.0 ± 0.4	81.4 ± 0.4	70.9 ± 0.5
GCN (Kipf and Welling 2017)	79.0	81.5	70.3
GAT (Veličković et al. 2018)	79.0 ± 0.3	83.0 ± 0.7	72.5 ± 0.7
AdaLNet (Liao et al. 2019)	78.1 ± 0.4	80.4 ± 1.1	68.7 ± 1.0
LNet (Liao et al. 2019)	78.3 ± 0.3	79.5 ± 1.8	66.2 ± 1.9
MixHop-learn (Abu-El-Haija et al. 2019b)	80.8 ± 0.58	81.9 ± 0.40	71.4 ± 0.81
SGC (Wu et al. 2019)	78.9 ± 0.0	81.0 ± 0.0	71.9 ± 0.1
MulStepNET (ours)	81.1 ± 0.0	83.7 ± 0.1	73.4 ± 0.0

Bold represents the best experimental results

To compare the stability, we report the average accuracy and standard deviation of GCN (see SGC), which is denoted as GCN*. We list the results that are averaged over 10 runs

Table 3 Comparison of computational complexity and the number of parameters

Method	Complexity	Parameters
GCN (Kipf and Welling 2017)	$O(n \times c_0 \times 16)$	$O(c_0 \times 16)$
MixHop-learn (Abu-El-Haija et al. 2019b)	$O(2 \times n \times c_0 \times 20)$	$O(c_0 \times 60)$
SGC (Wu et al. 2019)	$O(n \times c_0 \times 3)$ (Pubmed)	$O(c_0 \times 3)$ (Pubmed)
	$O(n \times c_0 \times 7)$ (Cora)	$O(c_0 \times 7)$ (Cora)
	$O(n \times c_0 \times 6)$ (Citeseer)	$O(c_0 \times 6)$ (Citeseer)
MulStepNET (ours)	$O(n \times c_0 \times 3)$ (Pubmed)	$O(c_0 \times 3)$ (Pubmed)
	$O(n \times c_0 \times 7)$ (Cora)	$O(c_0 \times 7)$ (Cora)
	$O(n \times c_0 \times 6)$ (Citeseer)	$O(c_0 \times 6)$ (Citeseer)

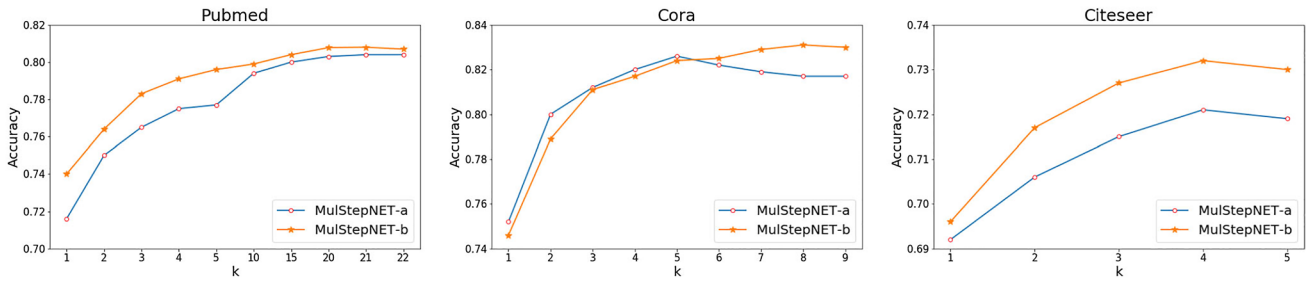


Fig. 3 Influence of model width (the highest power of \hat{A} , namely k) and own features (\hat{A}^0) on node classification accuracy. MulStepNET-a denotes MulStepNET without simple grouping, attention mecha-

nism, and \hat{A}^0 , then $\hat{A}_{0k} = \hat{A}^1 + \hat{A}^2 + \dots + \hat{A}^k$. MulStepNET-b denotes MulStepNET without simple grouping and attention mechanism, then $\hat{A}_{0k} = \hat{A}^0 + \hat{A}^1 + \hat{A}^2 + \dots + \hat{A}^k$

Table 4 Comparison of MulStepNET with and without \hat{A}^0

Method	Pubmed	Cora	Citeseer
MulStepNET without \hat{A}^0	80.0 ± 0.0	82.6 ± 0.0	72.0 ± 0.1
MulStepNET (ours)	81.1 ± 0.0	83.7 ± 0.1	73.4 ± 0.0

Bold represents the best experimental results

We list the results that are averaged over 10 runs

Table 5 Comparison of MulStepNET with and without attention multipliers

Method	Pubmed	Cora	Citeseer
MulStepNET without β	80.0 ± 0.1	83.0 ± 0.1	72.7 ± 0.1
MulStepNET (ours)	81.1 ± 0.0	83.7 ± 0.1	73.4 ± 0.0

Bold represents the best experimental results

β is the set of attention multipliers. In MulStepNET without attention multipliers, Pubmed dataset is slightly sensitive to initializations. We run the model 10 times and report the results for top 8 runs (sort by the results)

accuracy as model goes wider (increase k) until the width (k) of 21, 5, and 4 on Pubmed, Cora, and Citeseer respectively. Similarly, the width becomes 21, 8, and 4 for MulStepNET-b on respective datasets. Since MulStepNET-a and MulStepNET-b with wider model may mix the features from different clusters, they hurt the performance. Compared

to MulStepNET-a, MulStepNET-b with \hat{A}^0 significantly improves performance in most cases. This proves that the own features are more important for classification. Based on the comparison results, we determine the best model width and demonstrate the importance of own node features for designing model.

We conduct experiments based on whether MulStepNET has \hat{A}^0 . Table 4 lists the comparison results. We can observe from the results that MulStepNET outperforms MulStepNET without \hat{A}^0 by margins of 1.1, 1.1 and 1.4% on Pubmed, Cora, and Citeseer respectively, which further demonstrates the contribution of own node features to performance improvement.

To investigate the influence of attention multipliers, we remove all attention multipliers from MulStepNET while keeping other settings. Table 5 lists the comparison results between MulStepNET with and without attention multipliers. The results show that MulStepNET performs better. This demonstrates the benefits of the attention multipliers.

Figure 4 shows the classification accuracy of MulStepNET, MulStepNET without \hat{A}^0 , and MulStepNET without attention (β) on all three datasets, respectively. It is obviously to see that MulStepNET outperforms MulStepNET without \hat{A}^0 and MulStepNET without attention in terms of average accuracy and the overall trend of MulStepNET is better than MulStepNET without \hat{A}^0 and

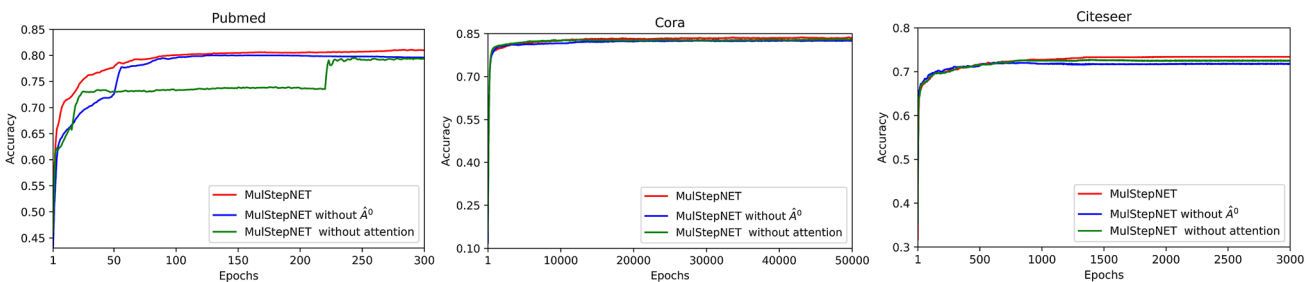


Fig. 4 Results of MulStepNET, MulStepNET without \hat{A}^0 , and MulStepNET without attention on citation network datasets

MulStepNET without attention. On Citeseer, the accuracy of MulStepNET is 1.9% (average value) and 1.0% (average value) higher than the classification result of MulStepNET without \hat{A}^0 and MulStepNET without attention, respectively. This further verifies the contribution of own node features and the attention multipliers to performance improvement.

5 Conclusion

In this paper, we propose a stronger multi-step graph convolutional network architecture, MulStepNET, on graph-structured data. Notably, our MulStepNET can obtain more nodes features information and enable adequate information propagation via multi-power adjacency matrix. Further, by precomputing the fixed feature extractor H_{ok} , our computations are more efficient than GCN. Experiments on several graph classification benchmarks show natural advantages for capturing node features and entire graph structure information. We observe that our MulStepNET with the fewest parameters achieves better performance as compared to baselines. In the future, we would apply the proposed model to more application areas such as social networks.

Acknowledgements This work was partly supported by the Guangdong Provincial Key Laboratory of Intellectual Property and Big Data (2018B030322016), Special Projects for Key Fields in Higher Education of Guangdong, China (2020ZDZX3077), the National Natural Science Foundation of China (U1701266), the Characteristic Innovation Projects of Ordinary Universities in Guangdong Province, China (2020KTSCX212, 2019KTSCX245), the Projects of South China Institute of Software Engineering of Guangzhou University (ky202012).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

Abu-El-Haija S, Kapoor A, Perozzi B, Lee J (2019a) N-GCN: multi-scale graph convolution for semi-supervised node classification. In: Uncertainty in artificial intelligence, pp 841–851

- Abu-El-Haija S, Perozzi B, Kapoor A, Harutyunyan H, Alipourfard N, Lerman K, Steeg G, Galstyan A (2019b) MixHop: higher-order graph convolution architectures via sparsified neighborhood mixing. In: International conference on machine learning, pp 21–29
- Al-Sharif Z, Al-Saleh M, Alawneh L, Jararweh Y, Gupta B (2020) Live forensics of software attacks on cyber-physical systems. *Future Gener Comput Syst* 108:1217–1229
- Atwood J, Towsley D (2016) Diffusion-convolutional neural networks. In: Neural information processing systems, pp 1993–2001
- Bruna J, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and locally connected networks on graphs. In: International conference on learning representations, pp 1–14
- Cai C, Wang Y (2018) A simple yet effective baseline for non-attribute graph classification. [arXiv:1811.03508](https://arxiv.org/abs/1811.03508)
- Chen J, Ma T, Xiao C (2018) FastGCN: fast learning with graph convolutional networks via importance sampling. In: International conference on learning representations, pp 1–15
- Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Neural information processing systems, pp 3844–3852
- Eliav B, Edith C (2018) Bootstrapped graph diffusions: exposing the power of nonlinearity. *ACM Measur Anal Comput Syst*, 2(1):1–19
- Gilmer J, Schoenholz S, Riley P, Vinyals O, Dahl G (2017) Neural message passing for quantum chemistry. In: International conference on learning representations, pp 1263–1272
- Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: AAAI conference on artificial intelligence, pp 922–929
- Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Neural information processing systems, pp 1024–1034
- He K, Sun J (2015) Convolutional neural networks at constrained time cost. In: IEEE conference on computer vision and pattern recognition. IEEE, pp 5353–5360
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition. IEEE, pp 770–778
- Huang G, Liu Z, Weinberger K (2017) Densely connected convolutional networks. In: IEEE conference on computer vision and pattern recognition. IEEE, pp 4700–4708
- Kampffmeyer M, Chen Y, Liang X, Wang H, Zhang Y, Xing E (2019) Rethinking knowledge graph propagation for zero-shot learning. In: IEEE conference on computer vision and pattern recognition. IEEE, pp 11487–11496
- Kipf T, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations, pp 1–14
- Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. In: Neural information processing systems, pp 1097–1105
- Lai Z, Chen R, Jia J, Qian Y (2020) Real-time micro-expression recognition based on ResNet and atrous convolutions. *J Ambient Intell Human Comput*, pp 1–12
- Lei F, Liu X, Dai Q, Ling B, Zhao H, Liu Y (2020) Hybrid low-order and higher-order graph convolutional networks. *Comput Intelli Neurosci*. <https://doi.org/10.1155/2020/3283890>

- Li Y, Tarlow D, Brockschmidt M, Zemel R (2015) Gated graph sequence neural networks. In: International conference on learning representations, pp 1–20
- Li Q, Han Z, Wu X (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI conference on artificial intelligence, pp 3538–3545
- Li D, Deng L, Gupta B, Wang H, Choi C (2019a) A novel CNN based security guaranteed image watermarking generation scenario for smart city applications. *Inf Sci* 479:432–447
- Li G, Muller M, Thabet A, Ghanem B (2019b) DeepGCNs: can GCNs go as deep as CNNs? In: IEEE/CVF international conference on computer vision, pp 9267–9276
- Liao R, Zhao Z, Urtasun R, Zemel R (2019) LanczosNet: multi-scale deep graph convolutional networks. In: International conference on learning representations, pp 1–18
- Luan S, Zhao M, Chang X, Precup D (2019) Break the ceiling: stronger multi-scale deep graph convolutional networks. In: Neural information processing systems, pp 10943–10953
- Monti F, Boscaini D, Masci J, Rodola E, Svoboda J, Bronstein M (2017) Geometric deep learning on graphs and manifolds using mixture model CNNs. In: CVPR, pp 5115–5124
- Sun K, Zhu Z, Lin Z (2020) Multi-stage self-supervised learning for graph convolutional networks. In: AAAI conference on artificial intelligence, pp 5892–5899
- Thekumparampil K, Wang C, Oh S, Li L (2018) Attention-based graph neural network for semisupervised learning. [arXiv:1803.03735](https://arxiv.org/abs/1803.03735)
- Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: International conference on learning representations, pp 1–12
- Wang X, Ye Y (2018) Zero-shot recognition via semantic embeddings and knowledge graphs. In: IEEE conference on computer vision and pattern recognition. IEEE, pp 6857–6866
- Wu F, Zhang T, Souza J, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: International conference on learning representations, pp 6861–6871
- Wu Z, Pan S, Chen F, Long G, Zhang C, Yu P (2020) A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 32(1):4–24
- Yang Z, Cohen W, Salakhutdinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: International conference on learning representations, pp 40–48
- Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: AAAI conference on artificial intelligence, pp 7370–7377
- Yu W, Qin Z (2020) Graph convolutional network for recommendation with low-pass collaborative filters. In: International conference on learning representations, pp 1–13
- Zhang J, Shi X, Xie J, Ma H, King I, Yeung D (2018a) GaAN: gated attention networks for learning on large and spatiotemporal graphs. In: Uncertainty in artificial intelligence, pp 339–349
- Zhang M, Cui Z, Neumann M, Chen Y (2018b) An end-to-end deep learning architecture for graph classification. In: AAAI conference on artificial intelligence, pp 4438–4445
- Zhou J, Cui G, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2018) Graph neural networks: a review of methods and applications. *AI Open* 1:57–81

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.