Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6665588, 9 pages
https://doi.org/10.1155/2021/6665588

Research Article

Multihop Neighbor Information Fusion Graph Convolutional Network for Text Classification

Fangyuan Lei,[1,2] **Xun Liu**,[3] **Zhengming Li**,[4] **Qingyun Dai**,[1,2] **and Senhong Wang**[5]

[1]*Guangdong Key Provincial Laboratory of Intellectual Property & Big Data, Guangdong Polytechnic Normal University, Guangzhou 510665, China*
[2]*School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou 510665, China*
[3]*Department of Electronics, Software Engineering Institute of Guangzhou, Guangzhou 510990, China*
[4]*Industrial Training Center, Guangdong Polytechnic Normal University, Guangzhou 510665, China*
[5]*School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China*

Correspondence should be addressed to Xun Liu; liuxun.stf@gmail.com

Received 27 December 2020; Revised 21 March 2021; Accepted 24 April 2021; Published 3 May 2021

Academic Editor: Andrea Semaničová-Feňovčíková

Graph convolutional network (GCN) is an efficient network for learning graph representations. However, it costs expensive to learn the high-order interaction relationships of the node neighbor. In this paper, we propose a novel graph convolutional model to learn and fuse multihop neighbor information relationships. We adopt the weight-sharing mechanism to design different order graph convolutions for avoiding the potential concerns of overfitting. Moreover, we design a new multihop neighbor information fusion (MIF) operator which mixes different neighbor features from 1-hop to k-hops. We theoretically analyse the computational complexity and the number of trainable parameters of our models. Experiment on text networks shows that the proposed models achieve state-of-the-art performance than the text GCN.

1. Introduction

Text classification problem is a fundamental problem in many natural language processing (NLP) applications, such as text mining, spam detection, summarization, and question-answering system [1–5]. Many deep learning approaches such as convolutional neural networks [6], recurrent neural networks (RNN) [7], and long short-term memory (LSTM) [8] are introduced to text classification.

Text could be constructed on a typical graph-structured network, and graph networks have natural advantages for processing such data. Scarselli et al. [9] proposed a graph neural network, which was widely used for text classification [10–12] and other NLP tasks [13]. Graph convolutional network (GCN) [10], which is the extension of the CNN on graph data, has shown good performance on text classification than the traditional CNN [14]. Yao et al. [15] proposed a text GCN to apply document nodes and weighted edges to construct the text network graph, and their model outperformed the state-of-the-art text classification methods.

When the messages pass through the graph of the text network, the node's output is affected by not only the directly connected nodes but also the k-hop nodes [16]. To obtain more neighbor node information, GCN models can expand the receptive field by stacking multiple layers. However, GNN models and GCN often suffer from the oversmoothing issue [17, 18]. On the contrary, the representation ability of the shallow network structure is clearly insufficient.

To address the above issue, we propose a multihop neighbor information fusion graph convolutional network for text classification based on the GCN. In our model, we propose a novel negative minimum value fusion operator to fuse multihop neighbor information (MIF). To reduce the computational complexity, we share the trainable weight [19] among the multihop neighbor nodes. Our experimental results show that our models achieve state-of-the-art

performance in several citation text datasets with lower computational complexity.

The contributions of this paper are as follows. First, we propose a novel negative minimum value fusion operator, which fuses the feature information of multihop neighbors. Second, we propose high-efficiency graph convolutional network-based MIF to successfully capture k-hop neighbors for nodes' classification of the text dataset.

The remainder of this paper is organized as follows. In Section 2, the related works are reviewed. In Section 3, our methods are proposed. In Section 4, the experimental results are presented. Finally, we draw the concluding remarks in Section 5.

2. Related Work

In this section, we will describe the related work about the graph convolutional network text classification, and we also introduce the related work about the multihop neighbor information of graph convolutional networks.

2.1. Graph Convolutional Network. Gori et al. [20] first proposed the concept of the graph neural network (GNN), which was based on the recurrent neural network architecture. Micheli [21] developed the GNN by random walk on the graph network [18]. Morris et al. [22] proved that the graph neural network and 1D Weisfeiler-Leman had the same ability to decompose nonisomorphic graphs.

Graph convolutional networks (GCNs) were developed from convolutional neural networks [23]. However, it is difficult to apply the GCN for large-scale graphs due to the high computational burden of eigenvalue decomposition [23]. Defferrard et al. [10] proposed a localized filter using Chebyshev polynomial. Kipf and Welling [24] proposed vanilla GCN, which achieves state-of-the-art classification performance on the citation network. Niepert et al. [25] proposed PATCHY-SAN to capture the information from locally connected regions. Hamilton et al. [26] developed a set of aggregate functions by sampling nodes in the neighbor to address the limitation of transductive learning. Monti et al. [27] contributed a unified framework for generalizing convolution to non-Euclidean domains. Velickovic et al. [28] leveraged masked attention to propose a graph attention network (GAT). Ding et al. [29] developed GAT and achieved better classification performance.

Recent works have been proposed for capturing k-hop neighbor information of nodes [30–33]. Zhou and Li [33] proposed a new high-order convolution operator to capture k-hop neighbor information and developed adaptive filtering to adjust the weights of the operator. Based on the motif graph attention mechanism, Lee et al. [34] proposed motif convolutional networks to capture k-hop interactive information. Mao et al. [35] proposed a Siamese framework to capture the k-hop information in the brain network. Abu-El-Haija et al. [36–38] proposed several versions of the mix-hop convolution to mix these features of different order graph convolutions using a fully connected layer.

2.2. Text Classification. Recently, deep learning models are introduced into text classification, which achieve far better performance than traditional models.

With the development of deep learning techniques, increasingly deep learning models are applied for text classification. Kim [39] developed several variants of the CNN model for the text classification. Recurrent neural networks (RNNs) [2, 7] are widely applied for text classification, showing better results than traditional models.

With the development of graph network models, many researchers developed more GCN-based classification methods [26, 40–42]. Zhaoa et al. [42] proposed the SDGCN model to capture the interdependencies hidden in the data. Liu et al. [43] developed TensorGCN to aggregate intragraph and intergraph information of the text graph. However, Text GCN [15] has a large number of parameters and high computational complexity. We will propose a novel GCN-based model to solve the issue in the Text GCN [15].

3. Method

In this section, we review the definition of related graph notations and analyse the layer-wise propagation model of the GCN in detail. Then, we develop a novel information propagation method to capture and fuse the multihop neighbor information. We propose two novel frameworks to capture the rich information of the text network. Finally, we analyse the computational complexity and parameter quantities of our models.

3.1. Notations' Definition. We assume graph signal $X \in R^{n\times c}$ could be characterized by the node feature matrix of the graph, where n and c represent the number of nodes and feature dimensions, respectively. Let A be the adjacency matrix representing its edge connection. We define the normalized Laplacian matrix L as $L = I_n - D^{-(1/2)}AD^{-(1/2)}$, where I_n and D denote the identity matrix and the degree matrix of the graph, respectively.

The popular convolutional propagation model [24] is as follows:

$$H^{(l+1)} = \mathrm{ReLU}\left(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}H^{(l)}\theta^{(l)}\right), \tag{1}$$

where $\widetilde{A}$ is the adjacency matrix with self-loops, namely, $\widetilde{A} = A + I_n$. The degree matrix $\widetilde{D}$ could be written as $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, where $H^{(l)}$ denotes the propagation matrix. If $l = 0$, then $H^{(0)} = X$, which means the input signal is connected to the network. The trainable weight matrix $\theta^{(l)}$ could be optimized by gradient descent. We repeat the application of the convolutional model to get the vanilla GCN [24] framework.

$$Y = \mathrm{softmax}\left(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}\mathrm{Re}LU\left(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}X\theta^{(0)}\right)\theta^{(1)}\right), \tag{2}$$

where $\theta^{(0)}$ and $\theta^{(1)}$ present different weight matrices. The classification function is softmax. The convolutional operator is essentially a linear combination of its own vertices

and one-hop neighbourhood vertices to make the same category of vertex features similar. Stacking two convolutional layers makes the vertices of the same category more closely connected and further eases the classification task. However, when more layers are applied, the vertices of different categories will be mixed and become indistinguishable, which is excessive smoothing [17, 18].

3.2. Multihop Neighbor Information Fusion with the Graph Convolutional Operator

Definition 1 (multihop neighbor information fusion (MIF)).

It is assumed that matrix $\widetilde{A}$ denotes the regularized adjacency matrix of graph G, where $\widehat{A} = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$. If $\mathbf{A} \in R^{n\times n}$, then $\widehat{A}^{(k)} \in R^{n\times n}$. The power matrix of $\widehat{A}$ is $\widehat{A}^{(k)}, \widehat{A}^{(k-1)}, \ldots, \widehat{A}^{(1)}, \widehat{A}^{(0)}$, where $\widehat{A}^{(0)}$ denotes the identity matrix. The multihop neighbor information fusion operator is to fuse the k-hop neighbor information with the element-wise topological information which is preserved. The MIF operator is defined as follows:

$$Z^{(l)} = -\min\left(\widehat{A}^{(1)}H^{(l)}W_l, \widehat{A}^{(2)}H^{(l)}W_l, \ldots, \widehat{A}^{(k)}H^{(l)}W_l\right), \tag{3}$$

where $\widehat{A}^{(k)}$ denotes the k-hop regularized adjacency matrix and $\widehat{A}^{(k)}H^{(l)}W_l$ represents the k-hop neighbor information.

Proposition 1. *The multihop neighbor information fusion operator is a topological preserved operator.*

Proof. If $\mathbf{A}^{(k)} \in \mathrm{R}^{n\times n}$, then $\mathbf{A}^{(0)}, \mathbf{A}^{(1)} \ldots, \mathbf{A}^{(k)} \in R^{n\times n}$, and the k-hop neighbor information in convolutional layer l has the same dimension, $\widehat{A}^{(1)}H^{(l)}W_l, \widehat{A}^{(2)}H^{(l)}W_l, \ldots, \widehat{A}^{(k)}H^{(l)}W_l \in R^{n\times r1}$. As defined in formula (3), the MIF operator is the element-wise operation. Therefore, the dimension of $Z^{(l)}$ is equal to the dimension of $\widehat{A}^{(i)}H^{(l)}W_l$. The MIF operator preserves topological information.

The MIF operator is an information aggregation layer that is used to mix k-order adjacency information. The procedure of the calculation of MIF is as follows:

(1) Calculating the minimum value of these features from different order graph convolutions:

$$P = \min\left(\widehat{A}^{(1)}H^{(l)}W_l, \widehat{A}^{(2)}H^{(l)}W_l, \ldots, \widehat{A}^{(k)}H^{(l)}W_l\right). \tag{4}$$

We give a living example to show that the MIF operator works. It is assumed that $k = 3$, namely, we obtain the maximum 3-hop neighbor information. We assume $P_1 = \widehat{A}^{(1)}H^{(l)}W_h = \begin{bmatrix} 1 & 7 \\ -1 & 0 \end{bmatrix}$, $P_2 = \widehat{A}^{(2)}H^{(l)}W_l = \begin{bmatrix} -2 & 2 \\ 3 & 4 \end{bmatrix}$, and $P_3 = \widehat{A}^{(3)}H^{(l)}W_l = \begin{bmatrix} 4 & 2 \\ -3 & -2 \end{bmatrix}$; then, the result is as follows:

$$P = \min\left(P_1, P_2, P_3\right) = \min\left[\left\{P_1^{(i,j)}, P_2^{(i,j)}, P_3^{(i,j)}\right\}\right] = \begin{bmatrix} -2 & 2 \\ -3 & -2 \end{bmatrix}. \tag{5}$$

(2) The output of the MIF operator is defined as negative for each element of P, namely, $Z = -P = \begin{bmatrix} 2 & -2 \\ 3 & 2 \end{bmatrix}$.

Following DIFFPOOL [44], we use the topological preserved operator MIF to improve the performance. □

3.3. MIF Propagation Model. To address the limitations of the Text GCN, we propose a propagation model of multihop neighbor information fusion graph convolution as follows:

$$H^{(l+1)} = \mathrm{ReLU}\left(\mathrm{MIF}\left(\widehat{A}^{(1)}H^{(l)}W_l, \widehat{A}^{(2)}H^{(l)}W_l, \ldots, \widehat{A}^{(k)}H^{(l)}W_l\right)\right), \tag{6}$$

where $\mathrm{MIF}(\cdot)$ is defined in formula (4), $\widehat{A} = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$, and $\widehat{A}^{(k)}$ denotes that $\widehat{A}$ multiplies itself by k times, where $\widehat{A}^{(j)}H^{(l)}W_l$ is the j-hop graph convolution, and W_l is the weight parameter matrix. In the MixHop model [36–38], Abu-El-Haija et al. adopted different weights for different $\widehat{A}^{(k)}$. To reduce the computational complexity, we share the weight in the same convolutional layer in the multihop convolutional operator.

In formula (6), the convolutional layers combine the multihop neighbor information from 1-hop graph convolution with k-hop graph convolution. The calculation procedure is summarized in Algorithm 1. The MIF has advantages compared to Text GCN. MIF implements feature aggregation on nodes and their k-hop neighbor nodes. Therefore, MIF contains multihop neighbor information, in which it captures more information than Text GCN. In summary, MIF merges multihop neighbor information features while avoiding the extra parameter number. Those nodes in the same category are more closely connected. Furthermore, the MIF operator suppresses excessive weighting features while retaining features with small weight values, which may prevent gradient disappearance and gradient explosion problems.

3.4. Graph Convolutional Network Based on MIF. In Figure 1, we propose a two-layer graph convolutional neural network using the MIF layer. The first layer is the graph convolutional layer with MIF, and the convolutional layer is represented as follows:

$$H = \sigma\left(MIF\left(\widehat{A}^{(1)}XW_1, \widehat{A}^{(2)}XW_1, \ldots, \widehat{A}^{(k)}XW_1\right)\right), \tag{7}$$

where W_1 is the weight parameter matrix between the input layer and hidden layer.

The second layer is the traditional graph convolution. We set the nonlinear activation function σ between the two layers as ReLU and achieve multiple classifications via softmax after the second layer. The network extends the 1-hop graph convolution to k-hop graph convolution to capture multihop neighbor interactive information. The output of our model is expressed as follows:

$$Z = \text{softmax}\left(\widehat{A}\left(\text{ReLU}\left(MIF\left(\widehat{A}^{(1)}XW_1, \widehat{A}^{(2)}XW_1, \ldots, \widehat{A}^{(k)}XW_1\right)\right)\right)W_2\right), \tag{8}$$

where W_2 represents the weight parameter matrix between the hidden layer and output layer. The trainable weight parameters W_1 and W_2 would be updated by gradient descent.

In the preliminary network design, we compare how many convolutional layers and hops fit to our model. The two-convolutional-layer network shows better performance than the three and more convolutional layer network. When we implement the multihop neighbor information fusion, we observe that $k = 2$ is better for most text networks, while $k = 3$ is better for a few networks. In further experiments, when $k \geq 4$, the classification results would decrease. Moreover, the larger the k value, the higher the computational cost. Therefore, we only discuss the cases of $k = 2$ and $k = 3$ in our models.

When $k = 2$, our MIF operator fuses the 1-hop graph convolutional layer and 2-hop graph convolutional layer. The 2-hop MIF graph convolutional layer (MIFGC-2) is as follows:

$$Z = \text{softmax}\left(\widehat{A}^{(1)}\left(\text{ReLU}\left(MIF\left(\widehat{A}^{(1)}XW_1, \widehat{A}^{(2)}XW_1\right)\right)\right)W_2\right). \tag{9}$$

When $k = 3$, the 3-hop MIF graph convolutional layer (MIFGC-3) fuses from 1-hop to 3-hop convolutional layer. The NMGC-3 model is as follows:

$$Z = \text{softmax}\left(\widehat{A}^{(1)}\left(ReLU\left(MIF\left(\widehat{A}^{(1)}\text{XW}_1, \widehat{A}^{(2)}\text{XW}_1, \widehat{A}^{(3)}\text{XW}_1\right)\right)\right)W_2\right). \tag{10}$$

The cross-entropy loss is utilized as our model loss function:

$$\mathscr{L} = -\sum_{l \in x_L}\sum_{m=1}^{M} Y_{lM} \ln Z_{lM}, \tag{11}$$

where x_L denotes the nodes set with labels and M represents the number of classes. Y_{lM} denotes the real labels of tag nodes, and Z_{lM} denotes the probability value between 0 and 1 predicted by softmax.

3.5. Computational Complexity and Parameters. Because the actual running time is sensitive to hardware and implementations, we follow He and Sun [45] to adopt the theoretical time complexity to show the complexity rather than the actual running time. For large-scale graph networks, it is a huge challenge to directly calculate $\widehat{A}^{(k)}$. Therefore, we calculate $\widehat{A}^{(k)}XW_1$ with right-to-left multiplication. For example, if $k = 2$, we calculate $\widehat{A}^{(2)}XW_1$ as $\widehat{A}(\widehat{A}(XW_1))$. $\widehat{A}$ is usually a sparse matrix with m nonzero entries. In formulas (7)–(10), our graph convolutional layers adopt the weight-sharing mechanism. Therefore, the calculation procedure is efficient.

Since different hop graph convolutions share the same weight in the same layer, the parameter quantities are consistent with the 1-hop graph convolution. It is assumed that $\widehat{A} \in R^{n \times n}$, where n is the number of nodes; $X \in R^{n \times r_0}$, where r_0 is the feature dimensions, $W_1 \in R^{r_0 \times r_1}$, where r_1 represents the number of hidden neurons in the 1st layer, and $W_2 \in R^{r_1 \times r_2}$, where r_2 represents the hidden neurons in the 2nd layer. Then, output dimension in the 1st layer as the same, namely, $\widehat{A}XW_1 \in R^{n \times r_1}$, $\widehat{A}^2XW_1 \in R^{n \times r_1}$, and $\widehat{A}^kXW_1 \in R^{n \times r_1}$. Therefore, in the first convolutional layer of our proposed model, the computational complexity is $O(k \times m \times r_0 \times r_1)$, and the trainable parameters are $O(r_0 \times r_1)$. The whole computational complexity of our proposed model is $O(k \times m \times r_0 \times r_1 + m \times r_1 \times r_2)$ with $O(\sum_{i=1}^{2} r_{i-1} \times r_i)$ trainable parameters. The node feature dimension is far large than the neural number, namely, $r_0 \gg r_2$. Therefore, the computational complexity of our network frameworks approximates $O(k \times m \times r_0 \times r_1)$, and trainable parameters approximate $O(r_0 \times r_1)$, respectively. It matches the computational complexity and parameters of vanilla GCN [24]. Similarly, Text GCN [15] takes $O(1 \times m \times r_0 \times r_1)$ computational complexity and $O(r_0 \times r_1)$ trainable parameters.

4. Experiment

We will evaluate our NMGC-2 and NMGC-3 on text networks and compare our methods with the classic method and deep learning methods, such as the embedding model, CNN-based, LSTM-based, and GCN-based. We analyse the terms of computational complexity and trainable parameters in detail. We investigate the impact of network framework parameters and training epochs on classification accuracy.

4.1. Datasets. We test our methods on five benchmark corpora datasets including R52 and R8 of Reuters-21578, 20-Newsgroups (20NG), Ohsumed, and Movie Review (MR). According to the preprocessing steps by Yao et al. [15], we process the text datasets and use the documents and words as nodes to build the text graph. In Table 1, the statistics characters of the datasets are described in detail.

4.2. Baseline and Experimental Settings. We compare with the following baseline methods as in Yao et al. [15], i.e., CNN with randomly initialized word vectors (CNN-rand) [39], CNN with pretrained word vectors (CNN-pretrain) [39],

input: $\widehat{A}, H^{(l)}, W$
output: H
for $i = 1: k$
$T_i = \widehat{A}^i H^{(l)}$
$P_i = T_i W = \widehat{A}^i H^{(l)} W$
end
$\mathbf{Z}^{(l)} = -\min(P_1, P_2, \ldots, P_k)$
$H = \mathrm{ReLU}(\mathbf{Z}^{(l)})$

Algorithm 1: MIF operation.

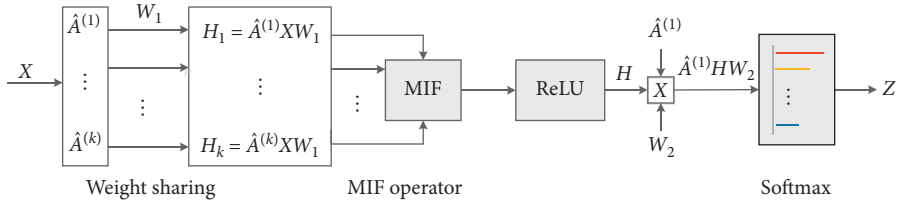

Figure 1: The proposed model.

predictive text embedding (PTE) [46], LSTM framework (LSTM) [2], LSTM framework with pretrained word embeddings (LSTM-pretrain) [2], fast text classifier (fastText) [47], fast text classifier with bigrams (fastText-bigrams) [47], label embedding model with attention (LEAM) [48], simple word-embedding model (SWEM) [49], graph CNN with spline filter (GCNN-S) [23], graph CNN with Fourier filter (GCNN-F) [50], and graph CNN with Chebyshev filter (GCNN-C) [10].

We tune a series of hyperparameters' (learning rate, dropout rate, hidden units, and epochs) values to determine the best hyperparameters of our model on text networks. The hyperparameters are reported in Tables 2 and 3. In our NMGC-2 and NMGC-3 models, we set $L2$ regularization factor as 0 and use Adam [51] to optimize the learning rate, following Yao et al. [15].

4.3. Results' Analysis. We compare our NMGC-2 and NMGC-3 with other baseline methods in terms of test accuracy. As shown in Table 4, the proposed model NMGC-2 or NMGC-3 achieves the highest classification performance on datasets R52, R8, 20NG, and Ohsumed. Specifically, our NMGC-2 obtains the best accuracy of 94.35%, 97.31%, and 69.21% on datasets R52, R8, and Ohsumed, respectively, whereas our NMGC-3 obtains the highest accuracy of 86.68%on 20NG.

The success of NMGC-2 and NMGC-3 is mainly due to the following three aspects. (1) Our NMGC-2 and NMGC-3 models have the capability to capture the relations in terms of word-word and document-word in the datasets. (2) Our NMGC-2 and NMGC-3 make full use of the advantages of the GCN. We implement the feature information aggregation on the node and its 1-hop neighbor information (each layer) so that the node (word-word and document-word) features in the same cluster are similar, which are easy to classify. (3) Our NMGC-2 and NMGC-3 capture more and richer feature information from 1-hop to k-hop neighbors, which may circumvent the limitations of the GCN.

On dataset MR, CNN-pretrain [39] achieves the highest classification result of 77.75%, which shows the model consecutive and short-distance semantics because CNN-pretrain [39] and LSTM-pretrain [2] model consecutive word sequences, while our NMGC-2 and NMGC-3 ignore the word orders. Another reason is that our NMGC-2 and NMGC-3 models are difficult to propagate the information among the nodes (word-word and document-word) on MR with few edges.

Compared to Text GCN [15], our NMGC-2 and NMGC-3 are better by a large margin in most cases, which demonstrates the effectiveness of our method in terms of capturing high-order interactive information.

4.4. Hidden Units' Analyses of Our Method. To evaluate the relationship between hidden units and the model performance, we use different hidden units to conduct experiments. We choose a representative set of hidden units as our comparative experiments in balancing computational complexity and classification performance. The results are summarized in Table 5. Specifically, our NMGC-2 uses the best hidden units of 128, 64, 128, 128, and 64 to achieve the higher accuracy on R52, R8, 20NG, Ohsumed, and MR, respectively, whereas our NMGC-3 uses the best hidden units of 128, 128, 128, 128, and 32 to achieve the better accuracy on R52, R8, 20NG, Ohsumed, and MR, respectively. We note that our NMGC-2 is always better than our NMGC-3 in many cases for different hidden units. This is most likely that our NMGC-3 suffers from oversmoothing in most cases. However, NMGC-3 outperforms NMGC-2 on short MR with few edges because NMGC-3 considers higher-order node information and propagates more label information to the entire graph network. In most cases, when the number of hidden units decreases, the accuracy of

Table 1: Dataset statistics.

Dataset	#training	#test	#docs	#classes	#words	#nodes	#length
R52	6532	2568	9100	52	8892	17,992	69.82
R8	5485	2189	7674	8	7688	15,362	65.72
20NG	11,314	7532	18,846	20	42,757	61,603	221.26
Ohsumed	3357	4043	7400	23	14,157	21,557	135.82
MR	7108	3554	10,662	2	18,764	29,426	20.39

Table 2: The hyperparameters in NMGC-2.

Dataset	Learning rate	Dropout rate	Hidden units	Epochs
R52	0.005	0.4	128	800
R8	0.01	0.4	64	200
20NG	0.005	0.5	128	330
Ohsumed	0.005	0.4	128	410
MR	0.01	0.4	64	40

Table 3: The hyperparameters in NMGC-3.

Dataset	Learning rate	Dropout rate	Hidden units	Epochs
R52	0.005	0.5	128	1300
R8	0.015	0.4	128	180
20NG	0.01	0.5	128	265
Ohsumed	0.01	0.6	128	210
MR	0.01	0.4	32	85

Table 4: Test accuracy on five text datasets. The benchmark results were reported by Yao et al. [15]. The accuracy values are the average result of 10 runs.

Method	R52	R8	20NG	Ohsumed	MR
CNN-rand [39]	85.37	94.02	76.93	43.87	74.98
CNN-pretrain [39]	87.59	95.71	82.15	58.44	**77.75**
PTE [46]	90.71	96.69	76.74	53.58	70.23
LSTM [2]	85.54	93.68	65.71	41.13	75.06
LSTM-pretrain [2]	90.48	96.09	75.43	51.10	77.33
fastText [47]	92.81	96.13	79.38	57.70	75.14
fastText-bigrams [47]	90.99	94.74	79.67	55.69	76.24
LEAM [48]	91.84	93.31	81.91	58.58	76.95
SWEM [49]	92.94	95.32	85.16	63.12	76.65
GCNN-S [23]	92.74	96.80	—	62.82	76.99
GCNN-F [50]	93.20	96.89	—	63.04	76.74
GCNN-C [10]	92.75	96.99	81.42	63.86	77.22
Text GCN [15]	93.56 ± 0.18	97.07 ± 0.10	86.34 ± 0.09	68.36 ± 0.56	76.74 ± 0.20
NMGC-2 (ours)	94.35 ± 0.06	97.31 ± 0.09	86.61 ± 0.06	69.21 ± 0.17	76.21 ± 0.25
NMGC-3 (ours)	93.83 ± 0.16	97.16 ± 0.10	86.68 ± 0.18	68.20 ± 0.35	76.36 ± 0.40

our method decreases, and more epochs are required to train our network.

4.5. Computational Complexity and Trainable Weight Parameters. We design the weight-sharing mechanism to share the weight in the proposed convolutional layer, which reduces the number of parameters. When using weight sharing, our calculations are very efficient. This naturally reduces the computational complexity. We design the weight-sharing mechanism to avoid overfitting caused by many parameters.

As shown in Table 6, we compare our method with Text GCN [15] in terms of computational complexity and the number of trainable weight parameters. The detailed derivation process of Comp. and Params. is analysed in Section 3.5. In Table 6, we observe that our NMGC-2 and NMGC-3 could match the computational complexity of Text GCN [15]. In particular, for MR, NMGC-3 is about 1 time less than Text GCN [15] in terms of computational complexity. Interestingly, our models have the least parameters on all datasets because we use the weight-sharing mechanism on these convolutions from 1-hop graph convolution to k-hop graph convolution and a smaller number of hidden units.

Table 5: Comparison of hidden units.

Dataset	Model	Hidden units	Epochs	Test Acc.
R52	NMGC-2 (ours)	128	800	94.35 ± 0.06
		64	1200	94.23 ± 0.20
		32	2000	94.16 ± 0.14
	NMGC-3 (ours)	128	1300	93.83 ± 0.16
		64	1100	93.29 ± 0.26
		32	2000	92.85 ± 0.37
R8	NMGC-2 (ours)	128	150	97.25 ± 0.13
		64	200	97.31 ± 0.09
		32	330	97.30 ± 0.13
	NMGC-3 (ours)	128	180	97.16 ± 0.10
		64	220	96.92 ± 0.08
		32	300	96.69 ± 0.06
20NG	NMGC-2 (ours)	128	330	86.61 ± 0.06
		64	480	86.55 ± 0.12
		32	600	86.02 ± 0.12
	NMGC-3 (ours)	128	265	86.68 ± 0.18
		64	420	86.14 ± 0.19
		32	540	85.64 ± 0.13
Ohsumed	NMGC-2 (ours)	128	410	69.21 ± 0.17
		64	495	69.07 ± 0.28
		32	795	68.46 ± 0.24
	NMGC-3 (ours)	128	210	68.20 ± 0.35
		64	305	67.50 ± 0.24
		32	415	66.25 ± 0.35
MR	NMGC-2 (ours)	128	35	76.19 ± 0.43
		64	40	76.21 ± 0.25
		32	55	76.16 ± 0.26
	NMGC-3 (ours)	128	50	76.27 ± 0.32
		64	70	76.24 ± 0.11
		32	85	76.36 ± 0.40

Table 6: Comparison of computational complexity and the number of trainable weight parameters. Comp. and Params. denote the computational complexity and parameters, respectively. Constant numbers 1, 2, and 3 represent the hops of the graph convolutional network, and constant numbers 200, 64, 128, and 32 denote the number of hidden units.

Method	Comp.	Params.
Text GCN [15]	$O(1 \times m \times r_0 \times 200)$	$O(r_0 \times 200)$
NMGC-2 (ours)	$O(2 \times m \times r_0 \times 64)$ (R8 and MR datasets) $O(2 \times m \times r_0 \times 128)$ (other datasets)	$O(r_0 \times 64)$ (R8 and MR datasets) $O(r_0 \times 128)$ (other datasets)
NMGC-3 (ours)	$O(3 \times m \times r_0 \times 32)$ (MR dataset) $O(3 \times m \times r_0 \times 128)$ (other datasets)	$O(r_0 \times 32)$ (MR dataset) $O(r_0 \times 128)$ (other datasets)

5. Conclusion

In this work, we propose a new multihop neighbor information fusion graph convolutional network on graph-structured data. We develop a novel MIF operator to combine the graph convolution features of multihop neighbor information from 1-hop graph convolution to k-hops. Experiments on text networks suggest that our models are capable of encoding in terms of node features and global graph topology in a way useful for graph classification. In this setting, our models achieve performance improvement compared to other methods while being computationally efficient and with less trainable parameters. In future work, we plan to study different fusion schemes and extend our model to more datasets.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (U1701266, 61702117, and 61672008), the Guangdong Provincial Key Laboratory of Intellectual

Property and Big Data (2018B030322016), the Special Projects for Key Fields in Higher Education of Guangdong (2020ZDZX3077), and in part by Qingyuan Science and Technology Plan Project (Grant nos. 170809111721249 and 170802171710591).

References

[1] S. Lai, L. Xu, K. Liu et al., "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence*, Austin, TX, USA, January 2015.

[2] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, http://arxiv.org/abs/160505101.

[3] H. Tao, S. Tong, H. Zhao et al., "A radical-aware attention-based model for Chinese text classification," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, February 2019.

[4] R. Narayan, J. K. Rout, and S. K. Jena, "Review spam detection using semi-supervised technique," *Advances in Intelligent Systems and Computing*, Springer, Berlin, Germany, pp. 281–286, 2017.

[5] S. Reddy, D. Chen, and C. D. Manning, "Coqa: a conversational question answering challenge," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 249–266, 2019.

[6] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014.

[7] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, http://arxiv.org/abs/180301271.

[8] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Revisiting LSTM networks for semi-supervised text classification via mixed objective function," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6940–6948, 2019.

[9] F. Scarselli, M. Gori, A. C. Tsoi et al., "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, pp. 3844–3852, Springer, Berlin, Germany, 2016.

[11] W. Li, S. Li, S. Ma et al., "Recursive graphical neural networks for text classification," 2019, http://arxiv.org/abs/190908166.

[12] L. Huang, D. Ma, S. Li et al., "Text level graph neural network for text classification," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3444–3450, Hong Kong, China, November 2019.

[13] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, "Text classification algorithms: a survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[14] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: a survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[15] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7370–7377, 2019.

[16] Q. Zhu, B. Du, and P. Yan, "Multi-hop convolutions on weighted graphs," 2019, http://arxiv.org/abs/191104978v1.

[17] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, February 2018.

[18] J. Zhou, G. Cui, Z. Zhang et al., "Graph neural networks: a review of methods and applications," 2018, http://arxiv.org/abs/181208434.

[19] F. Lei, X. Liu, Q. Dai et al., "Hybrid low-order and higher-order graph convolutional networks," *Computational Intelligence and Neuroscience*, vol. 2020, Article ID 3283890, 9 pages, 2020.

[20] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*, pp. 729–734, IEEE, Montreal, Canada, July 2005.

[21] A. Micheli, "Neural network for graphs: a contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[22] C. Morris, M. Ritzert, M. Fey et al., "Weisfeiler and leman go neural: higher-order graph neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4602–4609, 2019.

[23] J. Bruna, W. Zaremba, A. Szlam et al., "Spectral networks and locally connected networks on graphs," 2013, http://arxiv.org/abs/13126203.

[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, http://arxiv.org/abs/160902907.

[25] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the International Conference on Machine Learning*, pp. 2014–2023, New York, NY, USA, June 2016.

[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1024–1034, Long Beach, CA, USA, July 2017.

[27] F. Monti, D. Boscaini, J. Masci et al., "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, Honolulu, HI, USA, July 2017.

[28] P. Veličković, G. Cucurull, A. Casanova et al., "Graph attention networks," 2017, http://arxiv.org/abs/171010903.

[29] M. Ding, J. Tang, and J. Zhang, "Semi-supervised learning on graphs with generative adversarial nets," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 913–922, ACM, Torino Italy, October, 2018.

[30] R. Milo, S. Shen-Orr, S. Itzkovitz et al., "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[31] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Estimation of graphlet counts in massive networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 44–57, 2018.

[32] R. A. Rossi, N. K. Ahmed, and E. Koh, "Higher-order network representation learning," in *Companion Proceedings of the Web Conference*, Lyon, France, April 2018.

[33] Z. Zhou and X. Li, "Convolution on graph: a high-order and adaptive approach," 2017, http://arxiv.org/abs/1706.09916.

[34] J. B. Lee, R. A. Rossi, X. Kong et al., "Higher-order graph convolutional networks," 2018, http://arxiv.org/abs/180907697.

[35] Y. Mao, Y. Shen, G. Qin et al., "Predicting the popularity of online videos via deep neural networks," *CoRR*, p. 10718, 2017, https://arxiv.org/abs/1711.10718.

[36] S. Abu-El-Haija, N. Alipourfard, H. Harutyunyan et al., "A higher-order graph convolutional layer," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal, Canada, December 2018.

[37] S. Abu-El-Haija, A. Kapoor, B. Perozzi et al., "N-gcn: multi-scale graph convolution for semi-supervised node classification," 2018, http://arxiv.org/abs/180208888.

[38] S. Abu-El-Haija, B. Perozzi, A. Kapoor et al., "MixHop: higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proceedings of the International Conference on Machine Learning*, pp. 21–29, Long Beach, CA, USA, June 2019.

[39] Y. Kim, "Convolutional neural networks for sentence classification," 2014, http://arxiv.org/abs/14085882.

[40] H. Peng, J. Li, Y. He et al., "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proceedings of the 2018 World Wide Web Conference*, pp. 1063–1072, Lyon, France, April 2018.

[41] A. A. Helmy, "A multilingual encoding method for text classification and dialect identification using convolutional neural network," 2019, http://arxiv.org/abs/190307588.

[42] P. Zhaoa, L. Houb, and O. Wua, "Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification," 2019, http://arxiv.org/abs/190604501.

[43] X. Liu, X. You, X. Zhang, J. Wu, and P. Lv, "Tensor graph convolutional networks for text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 5, pp. 8409–8416, 2020.

[44] Z. Ying, J. You, C. Morris et al., "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pp. 4805–4815, Montréal, Canada, December 2018.

[45] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5353–5360, Boston, MA, USA, June 2015.

[46] J. Tang, M. Qu, and Q. Mei, "Pte: predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174, ACM, Sydney, Australia, August 2015.

[47] A. Joulin, E. Grave, P. Bojanowski et al., "Bag of tricks for efficient text classification," 2016, http://arxiv.org/abs/160701759.

[48] G. Wang, C. Li, W. Wang et al., "Joint embedding of words and labels for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 2321–2331, Melbourne, Australia, July 2018.

[49] D. Shen, G. Wang, W. Wang et al., "Baseline needs more love: on simple word-embedding-based models and associated pooling mechanisms," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 440–450, Melbourne, Australia, July 2018.

[50] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, http://arxiv.org/abs/150605163.

[51] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, http://arxiv.org/abs/14126980.