Cross Message Passing Graph Neural Network

Zeyu Zhang
School of Computer Information and Engineering
Jiangxi Normal University
Nanchang, China
zzyjxnu@jxnu.edu.cn

Zheng Liu
Jiangsu Key Laboratory of BDSIP
Nanjing University of Posts and Telecommunications
Nanjing, China
zliu@njupt.edu.cn

Qiyun Zhou
School of Computer Information and Engineering
Jiangxi Normal University
Nanchang, China
zhou_qi_yun@qq.com

Yanwen Qu[*]
School of Computer Information and Engineering
Jiangxi Normal University
Nanchang, China
qu_yw@jxnu.edu.cn

***Abstract*—Most Graph Convolutional Networks (GCNs) used for graph classification task can fit into the Message Passing Neural Networks (MPNNs) framework. However, traditional MPNNs don't consider global information in the message passing phase in which the node embeddings are updated. In this paper, we propose a new model called Cross Message Passing Graph Neural Network (CMPGNN). The new model consists of two message passing phases, of which one is the local message passing phase used for node embeddings updating and another is the global message passing phase used for graph feature updating. Several convolutional layers are stacked together in the local message passing phase, in which different convolutional layers are used to update the node embeddings at different time steps. Each convolutional layer updates node embeddings not only according to the outputs calculated at the previous convolutional layer but also to the graph feature calculated at the previous time step. A readout layer shared by all time steps is used in the global message passing phase. At each time step, after the node embeddings are updated, the readout layer aggregates the embeddings of all nodes by using a global gated network, and feeds the aggregation result into a Gated Recurrent Unit (GRU) to update the graph feature.The above two message passing phases are executed alternatively. After all time steps, the graph feature obtained by the readout layer is fed into a MultiLayer perception for graph classification. We evaluate the performance of CMPGNN on 6 graph classification datasets. Experimental results show that compared with other 10 baselines, CMPGNN achieves the highest accuracy on 4 of the 6 benchmark datasets.**

Keywords—graph classification, message passing neural network, graph convolutional network, recurrent neural network

I. INTRODUCTION

Deep learning methods have shown great success in Euclidean or grid-structure data. However, there are also many non-grid data in real world, such as graph data. Graph learning can be applied to many areas, such as social network analysis, knowledge graph, bio-informatics, chem-informatics, natural language processing, and computer vision. Due to the structure complexity of graph data and the unordered of node set, traditional deep learning methods can't be directly applied to graph data. In recent years, researchers have developed many different kinds of Graph Neural Networks (GNNs) for graph learning, among which, Graph Convolutional Networks (GCNs) [1], Graph Attention Network (GAT) [2], PATCHSAN [3], Sortpool [4], GraphSAGE [5], Graph Isomorphism Network (GIN) [6] etc., have achieved state-of-the-art performance on many tasks such as node classification, link prediction and graph classification. In this paper, we focus on the graph classification task.

Gilmer et al. find that many GNNs based graph classification approaches can fit into a general framework called the Message Passing Neural Network (MPNNs) [7]. Message Passing framework consists of two phases: the message passing phase which is used for updating node embeddings and the graph feature extraction phase (readout phrase). In fact, many GNNs which are not originally designed for graph classification tasks can also fit into the message passing phase, such as GAT. Message passing phase updates node embeddings iteratively by propagating message at different time steps. As for GCNs, each time step corresponds to a independent graph convolutional layer. Adopting the programming framework introduced in [8], the message passing phase at each time step can be divided into three sub-operations: message generation sub-operation, message aggregation sub-operation, and node embedding updating sub-operation. Readout phase aggregates the embeddings of all nodes to extract graph feature. The differences between different models lay in the design of the sub-operations and the readout layer.

The above methods have achieved state-of-the-art performance in many tasks, especially in graph classification task, but there is much room for improvement. On the one hand, at each time step of the message passing phase, although some MPNNs, can generate or aggregate messages selectively, they only consider the local information of each node and its neighbor nodes, and do not consider the global information of the graph. On the other hand, in traditional MPNNs, the readout layer follows the message passing phase, which means the graph feature extraction is performed only after the last time step of the message passing phase. The graph feature will not be iteratively updated as the time step increases, so it is unable to take full advantage of current graph feature at each time step. Therefore, we propose a new model named the Cross Message Passing Graph Neural Network (CMPGNN) for graph classification task. The new model also follows the MPNNs framework. But different from traditional MPNNs, CMPGNN contains two message passing embeddings updating: one is the local message passing phase used for node embeddings updating; the other is the global message passing phase for graph feature updating.The above two message passing phases are executed alternatively which makes it possible to combine the global information into

[*] Corresponding author

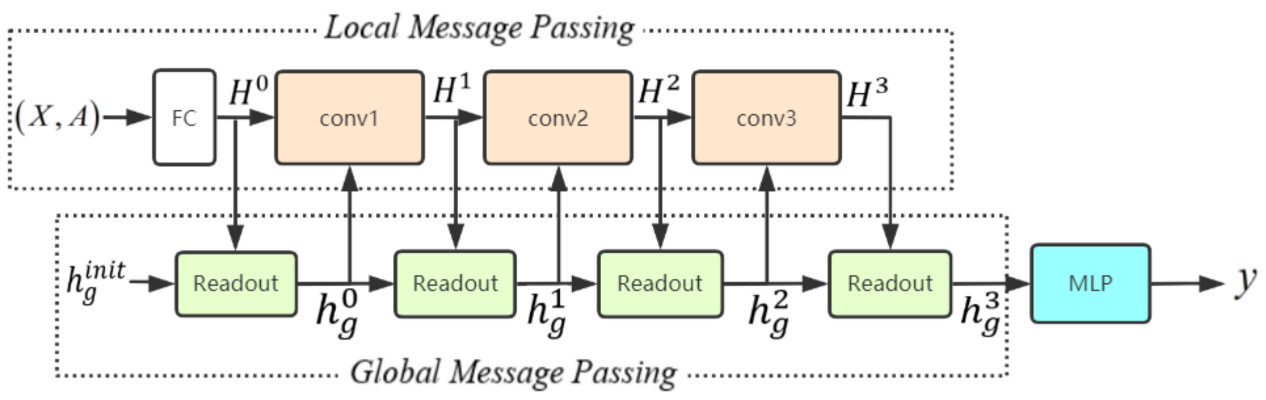

Fig. 1. Framework of the CMPGNN. The upper part is the local message passing process, and the lower part is the global message passing process.

the node embeddings updating by using the graph feature.

Our contributions of this paper are as follows: (1) We present a new convolution operation for node embeddings updating, which incorporates global information and gate mechanism; (2) A novel message passing process named the cross message passing is proposed, the new process updates the node embeddings and global feature alternatively; (3) Based on the cross message passing process, we propose an end-to-end graph neural network-CMPGNN for graph classification[a], the new network is permutation invariant; (4) We test and evaluate CMPGNN under 6 graph classification datasets, and experimental results show that it achieves state-of-the-art performance on benchmarks.

The remainder of this paper is structured as follows: Section 2 provides an overview of related research. Section 3 presents the framework and design details of the new model, and the experimental results of the new model on 6 benchmark datasets are given in Section 4. Finally, Section 7 concludes the paper, and discusses our future work.

II. Related Work

In this section, we will give a brief introduction to the MPNNs framework and the GCNs.

A graph $G=(V,A,E,X)$, where $V=\{v_i|i=1,\dots,N\}$, N denotes the number of nodes, E is the set of edges, $X\in R^{N\times D}$ represents the input feature vector of node v_i. $v_i\in V$ denotes a node and $e_{ij}=(v_i,v_j)\in E$ denotes an edge. In this paper ,we only consider the simple graph, that is, A is $N\times N$ binary symmetric matrix with $A_{ij}=1$ if $e_{ij}\in E$ and $A_{ij}=0$ if $e_{ij}\notin E$. The original graphs in datasets have no self-loops.

For graphs with node labels or attributes, each row of X can be represented as the one-hot encoding of the node label or as a vector of attributes. For graphs without node label and attributes, the one-hot encoding of node degree is used as a node's input feature, where degree $D_{ii}=\sum_j A_{ij}$. For node v, $N(v)$ is used to represent the neighbor nodes set.

A. MPNNs Framework

Many graph neural networks for graph classification tasks follow the message passing framework which consists in a message passing phase and a readout phase [7]. At the t-th time step of the message passing phase, the message generation, message aggregation, and node embeddings updating sub-operations are performed sequentially as follows:

$$m_{ij}^t = M_t\left(h_i^{t-1}, h_j^{t-1}, h_{ij}\right),\ v_j\in v_i\cup N(v_i) \tag{1}$$

$$a_i^t = \square_t\left(\left\{m_{ij}^t | v_j\in v_i\cup N(v_i)\right\}\right) \tag{2}$$

$$h_i^t = U_t\left(h_i^{t-1}, a_i^t\right) \tag{3}$$

where h_i^{t-1} is the hidden state of v_i at time step t-1. $M_t()$ is message generation function. m_{ij}^t is the message passed from node v_j to v_i. $\square_t$ () is the message aggregation function, which can be *MAX*, *MEAN* or *SUM*. $U_t()$ denotes the node embeddings updating function. In some datasets, edge e_{ij} has corresponding input edge feature h_{ij} . So MPNNs framework also considers h_{ij} when performing the message generation sub-operation. Here, we don't consider input edge features.

In the readout phase, graph feature extraction is performed to obtain the graph feature:

$$h_g = Readout\left(\left\{h_i^T | v_i\in V\right\}\right) \tag{4}$$

where T is the length of the time steps of the message passing phase. Global pooling operation, such as summation or average, is usually used as the readout function.

B. Graph Neural Networks

In recent years, varieties of Graph Convolutional Networks have been proposed by researchers. GCNs can be divided into spectral based methods and spatial based methods. Among them, the spectral based methods includes: Bruna et al. first define convolution in the Fourier domain using the graph Laplacian matrix, and generate non-spatially localized filters [9]. Defferrard et al. use the Chebyshev expansion of the graph Laplacian to avoid an explicit use of the graph Fourier basis [10]. Kipf et al. further simplify spectral approach by using simple filters operating on 1-hop neighborhoods of the graph [1].

[a] https://github.com/earverse/CMPGNN

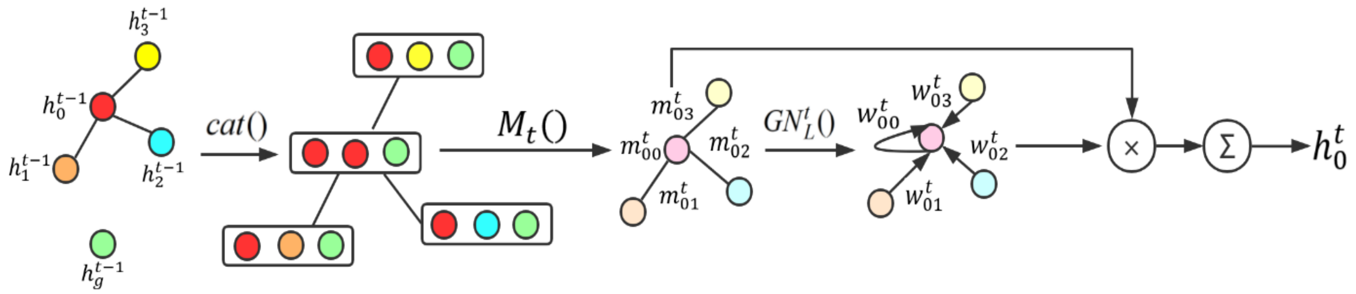

Fig. 2. Schematic diagram of the convolutional layer in the local message passing phase of the CMPGNN. This figure takes node v_0 as an example to show its embedding updating process at time step t. The convolutional layer first calculates the messages $\{m_{0j}^t\}$ passed from nodes $\{v_j\}$ to v_0. When calculating $\{m_{0j}^t\}$, the graph feature h_g^{t-1} obtained at time step t-1 will be taken into account. Then $\{m_{0j}^t\}$ will be aggregated according to the weights $\{w_{0j}^t\}$ calculated by the local gated network GN_L^t () , Finally, the node embedding of v_0 is updated.

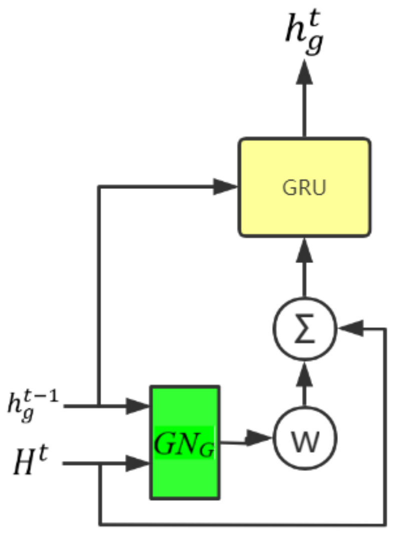

Fig. 3. Schematic diagram of the readout layer in the global message passing phase of the CMPGNN.

Spatial based methods include GraphSAGE [5], GAT [2], GIN [6] etc.

The spectral based methods rely on the eigen-decomposition of the Laplacian matrix, that means they can not be applied to a graph with a different structure [11]. The spatial based methods perform convolution operation directly on the graph domain, which is similar to the convolution operation on the image. However, unlike images, the number of neighbor nodes is different for each node, and the node set is unordered. Therefore, the spatial based graph convolution operation aggregates the embedding of each node and its neighbor nodes as a new embedding by using the adjacency matrix. In order to expand the receptive field of each node, multiple convolutional layers are usually stacked. After message passing phase, a readout layer is adopted to extract the graph features for the graph classification task.

In the CMPGNN proposed in this paper, multiple spatial convolutional layers are stacked to update the node embeddings during the local message passing phase. And the graph feature is iteratively updated based on the GRU [12] during the global message passing phase.

III. Preposed Method

A. Framework Description

We first introduce the design inspiration of the model, and then the framework of the model.

As illustrated in (1) , for the node v_i , the message m_{ij}^t generated by the neighbor node $v_j \in v_i \cup N(v_i)$ in the message generation sub-operation only relates to v_i^{t-1}, v_j^{t-1} and h_{ij} at time step t. That means only local information is utilized. In order to effectively introduce global information into the node embedding updating, a possible approach is to calculate the graph feature at each time step as the global information. In the traditional MPNNs, as illustrated in (4), the graph feature h_g is usually calculated according to the node embeddings obtained at the last convolutional layer. Recently, some scholars suggested that in order to make full use of the information at different scales, the node embeddings at all layers should be concated to obtain the graph feature [6]. However, if such approach is adopted to calculate the graph feature at each time step, we need to employ different readout layers at different time steps because of the input dimensions are various. A possible way is to use the GRU as the readout layer, which shares parameters at all time steps and has long short-term memory characteristics.

On the other hand, (2) shows that in the message aggregation sub-operation, the messages $\{m_{ij}^t\}$ generated by nodes in $v_i \cup N(v_i)$ are usually averaged or weighted and summed to gain the aggregation information a_i^t . For example, GAT assigns more weights to important messages in the aggregation process [2]. However, when two nodes v_i and $v_i^{'}$ have different numbers of neighbor nodes but have the same neighbor node embedding distribution, a_i^t and $a_{i'}^t$ may be the same after the aggregation sub-operation. In order to solve the above problems, some scholars suggest using the *SUM* function for aggregation, such as the GIN [6], but unimportant information may be introduced which is opposite to the intuition of the GAT. In order to leverage the advantages of both GAT and GIN, a new graph convolution method is proposed in this paper. In the new method, at each convolutional layer such as the t-th layer (the layer used at the time step t during the message passing phase), a local gated network GN_L^t is employed to compute a local passing coefficient for each dimension of $\{m_{ij}^t\}$, and the element wise product of the m_{ij}^t and corresponding coefficient vector is adopted as the valuable information passing from v_j to v_i. Then the aggregation sub-operation is performed by summing up all valuable information from local region. In order to avoid introducing redundant information, the L2 Norm of the local passing coefficient vector will be used as the regularization term in the loss function.

Based on the above ideas, we propose a new model named the Cross Message Passing Graph Neural Network, which is used for graph classification task. Fig. 1 shows the framework of CMPGNN. The CMPGNN consists of two cross-over message passing phases and one MLP layer for classification. The upper part of Fig. 1 shows the local message passing phase used for the node embeddings updating, and the lower part shows the global message passing phase used for the graph feature updating. The local message passing phase consists of a Fully Connected layer which is used for transforming initial node feature, and several convolutional layers. The t-th convolutional layer $conv_t$ is used to calculate the node embedding $H^t \in R^{N \times D_t}$ at the time step t, where D_t represents the dimension of the node embedding and the i_{th} row of H^t is the node embedding h_i^t of v_i. The global message passing phase consists of a readout layer which shares parameters at all time steps. The structure of the readout layer is shown in Fig. 3, it contains a global gated network GN_G and a GRU network. The graph feature is the hidden state of the GRU network. At each time step, the graph feature updating and the node embeddings updating are performed alternatively. As shown in Fig. 1, at the time step t, the network first performs the local message passing phase to update the node embeddings H^t, and then performs the global message passing phase to update the graph feature h_g^t.

B. Convolution Layer

The details of the convolutional layer will be introduced below. Referring to the MPNNs framework, the t-th convolutional layer is composed of a message generation function, a message aggregation function, and a node updating function. Assuming that node embeddings $\{h_i^{t-1}\}$ and graph feature h_g^{t-1} have been obtained at the time step t-1, for each nodes pair v_i and v_j, the convolutional layer first calculates the message m_{ij}^t passed from node v_j to v_i according to the following equation, where v_j belongs to the neighborhood of v_i with self-loop. We use LeakyRelu [13] function as the activation function:

$$m_{ij}^t = LeakyRelu(LeakyRelu(cat(h_i^{t-1} || h_j^{t-1} || h_g^{t-1})w_1^t + b_1^t)w_2^t + b_2^t) \quad (5)$$

Then we preform the message aggregation for v_i. In order to select valuable information, we first calculate the local passing coefficients of m_{ij}^t in each dimension according to the local gated network GN_L^t of the t-th layer, and then the element wise product of the m_{ij}^t and corresponding coefficient vector w_{ij}^t is calculated to obtain a_i^t:

$$w_{ij}^t = GN_L^t\left(m_{ij}^t\right) = sigmoid\left(m_{ij}^t w_l^t + b_l^t\right) \quad (6)$$

$$a_i^t = \sum_j w_{ij}^t \odot m_{ij}^t \quad (7)$$

Finally, a_i^t is passed into the update function to get the node embedding h_i^t, here $h_i^t = a_i^t$.

The $\{w_1^t, b_1^t, w_2^t, b_2^t, w_l^t, b_l^t\}$ in (5) and (6) are the parameters needed to be trained in the t-th convolutional layer.

Given a node v_0 and its neighbor nodes $\{v_1, v_2, v_3\}$, Fig. 2 shows the schematic calculation diagram of the convolution operation for node v_0 From left to right, message generation operation, local gated network operation, and summation aggregation operation are performed sequentially.

C. Readout Layer and Classification Layer

Fig. 3 shows the calculation flow of the readout layer. It consists of two parts, the lower part is a global gated network GN_G followed by global weighted sum operation, and the upper-right part is a GRU. The readout layer is shared by all time steps. Assuming that the node embeddings H^t and the graph feature h_g^{t-1} are available. First, for any node v_i, the coefficient vector w_i^t is calculated according to the following equation:

$$w_i^t = GN_G\left(h_i^t, h_g^{t-1}\right) = sigmoid\left(cat\left(h_i^t \,\|\, h_g^{t-1}\right)w_g + b_g\right) \quad (8)$$

where $\{w_g, b_g\}$ are the parameters needed to be trained.

Next, the aggregated information provided by all nodes is calculated according to (9):

$$a_g^t = \sum_i w_i^t \odot h_i^t \quad (9)$$

Then, a_g^t and h_g^{t-1} are sent to the GRU to update graph feature h_g^t:

$$h_g^t = GRU\left(h_g^{t-1}, a_g^t\right) \quad (10)$$

Finally, after T time steps, the hidden state h_g^T of the GRU is sent into a classification layer which consists of two fully connected layers to predict the probability distribution of labels y:

$$\mathrm{y} = softmax\left(LeakyRelu\left(BN\left(h_g^t\right)w_c^1 + b_c^1\right)w_c^2 + b_c^2\right) \quad (11)$$

where $\{w_c^1, b_c^1, w_c^2, b_c^2\}$ are the parameters needed to be trained, BN represents the Batch Normalization layer.

D. Loss Function

The loss function L is calculated as follows:

$$L = L_{CE} + \lambda_1 L_2 + \lambda_2 L_L + \lambda_3 L_G \quad (12)$$

L is composed of four parts λ_1, λ_2 and λ_3 are hyper parameters. L_{CE} represents the cross entropy loss. L_2 represents the L2 regularization term of the model parameters. L_L is used to constrain the L2 norm of the local passing coefficients calcula-

TABLE I. Dataset Information

Datasets	Properties				
	Number of Graphs	*Number of Classes*	*Average Number of Nodes*	*Average Number of Edges*	*Node Labels*
MUTAG	188	2	17.93	19.79	√
PTC	344	2	14.29	14.69	√
PROTEINS	1113	2	39.06	72.82	√
IMDB-Binary	1000	2	19.77	96.53	×
IMDB-Multi	1500	3	13.00	65.94	×
COLLAB	5000	3	74.49	2457.78	×

Table II. Average Classification Accuracy In Percent, And The Standard Deviation (Behind ±)

	Methods	Datasets					
		MUTAG	*PTC*	*PROTEINS*	*IMDB-Binary*	*IMDB-Multi*	*COLLAB*
Kernel	WL[21,22,23]	80.72 ± 3.00	57.97 ± 0.49	74.68 ± 0.49	73.40 ± 4.63	49.33 ± 4.75	**79.02 ± 1.77**
	GK[21,22,23]	81.58 ± 2.11	57.26 ± 1.41	71.67 ± 0.55	65.87 ± 0.98	43.89 ± 0.38	72.84 ± 0.28
	RW[22,23]	83.68 ± 1.66	57.85 ± 1.30	74.22 ± 0.42	--	--	--
	SP[22,23]	85.79 ± 2.51	58.24 ± 2.44	75.07 ± 0.54	--	--	--
GNN	PATCHYSAN[22,23]	**88.90 ± 4.37**	62.29 ± 5.68	75.00 ± 2.51	71.00 ± 2.29	45.23 ± 2.84	72.60 ± 2.15
	DCNN[22,23]	66.98	56.60 ± 2.89	61.29 ± 1.60	49.06 ± 1.37	33.49 ± 1.42	52.11 ± 0.71
	Sortpool[22,23]	85.83 ± 1.66	58.59 ± 2.47	75.54 ± 0.94	70.03 ± 0.86	47.83 ± 0.85	73.76 ± 0.49
	DGK[22,23]	82.66 ± 1.45	60.08 ± 2.55	75.68 ± 0.54	66.96 ± 0.56	44.55 ± 0.52	73.09 ± 0.25
	GraphSAGE[24]	74.90 ± 8.70	--	73.80 ± 3.60	72.40 ± 3.60	--	79.70 ± 1.70
	GIN-0[24]	85.70 ± 7.70	--	72.10 ± 5.10	72.80 ± 4.50	--	79.30 ± 2.70
	GIN-ε[24]	83.40 ± 7.50	--	72.60 ± 4.90	72.10 ± 5.10	--	79.80 ± 2.40
	CMPGNN	87.76 ± 7.80	**65.11 ± 9.61**	**75.92 ± 3.77**	**73.70 ± 2.83**	**50.13 ± 3.38**	77.38 ± 2.29
	CMPGNN-gi	86.54 ± 8.91	59.67 ± 7.77	74.03 ± 3.45	72.60 ± 4.63	46.93 ± 2.67	76.03 ± 5.90

ted by the local gated network GN_L^t in each convolutional layer, so as to reduce the redundant information or noise. Similar to L_L, L_G is used to constrain the L2 norm of the passing coefficients of the global gated network GN_G.

Assuming each minibatch has M graphs, the details of L_L and L_G are described below:

$$L_L = \frac{1}{M}\sum_{m=1}^{M} L_{m,L} \tag{13}$$

where $L_{m,L}$ represents the regularization term of the m-th graph in L_L, and the formula of $L_{m,L}$ is shown in (14). Here, in order to avoid excessive symbol definition, the mark 'm' is omitted in the symbols appeared in the right part of the formula:

$$L_{m,L} = \sum_{t\text{-}1}^{T} \frac{1}{N_v} \sum_{v_i \in V} \sum_{j \in v_i \cup N(v_i)} \frac{\|w_{ij}^t\|_2}{D_{ii}} \tag{14}$$

$$L_G = \frac{1}{M}\sum_{m=1}^{M} L_{m,G} \tag{15}$$

where $L_{m,G}$ represents the regularization term of the m_{th} graph in L_G, and the formula of $L_{m,G}$ is shown in (16). Similar to the (14), the mark 'm' is omitted:

$$L_{m,G} = \sum_{t=1}^{T} \frac{1}{N_v} \sum_{v_i \in V} \| w_{ij}^t \|_2 \tag{16}$$

IV. Experiments

In order to evaluate the performance of our model, we compare it with 10 baseline algorithms, including 4 graph kernel based methods and 6 deep learning based methods, on 6 graph classification datasets.

A. Datasets

The benchmark include three bio-informatics datasets: MUTAG, PTC and PROTEINS, and three social network datasets: IMDB-Binary, IMDB-Multi and COLLAB. The social network datasets do not have node labels, therefore we follow the convention, using one-hot encoding of node degree as the input feature for each node. Bio-informatics datasets use one-hot encoding of node label as the input feature for each node. Table I lists relevant information for the 6 benchmark datasets [14].

B. Experimental Configuration

In this paper we compare the classification accuracy of the CMPGNN with 10 baselines: PATCHSAN [3] , Sortpool [4] (Also named DGCNN [4, 22, 23]) , GraphSAGE [5] , GIN [6], Graphlet Kernel(GK) [15] , Weisfeiler-Lehman subtree kernel (WL) [16], Shortest-Path kernel (SP) [17], Random Walk kernel (RW) [18], Deep Graph Kernel (DGK) [19], Diffusion-CNN (DCNN) [20] . Since GAT was not designed for graph classification task in its paper, we did not include it in the evaluation. In addition, in order to analyze the role of global message passing, we constructed a CMPGNN network with global information removed (CMPGNN-gi) to perform ablation

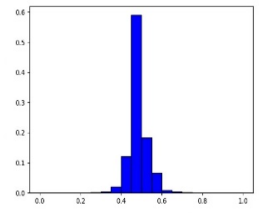
Fig. 4 (a). IMDB-Multi

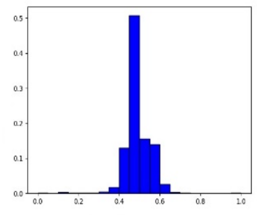
Fig. 4 (b). IMDB-Binary

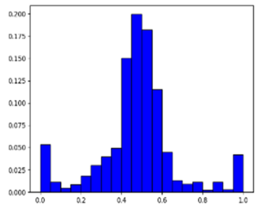
Fig. 4 (c). COLLAB

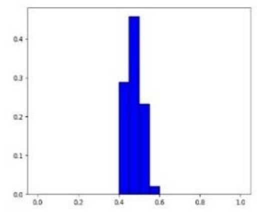
Fig. 4 (d). PTC

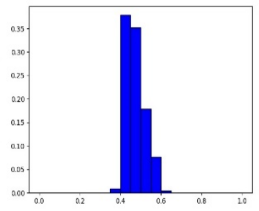
Fig. 4 (e). PROTEINS

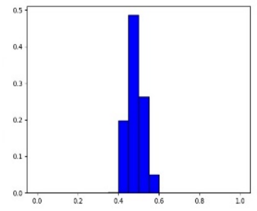
Fig. 4 (f). MUTAG

Fig. 4. The histogram of the local passing coefficients obtained on the test samples of each dataset. The X axis represents the range of the coefficient, and the Y axis represents the probability distribution on each bin.

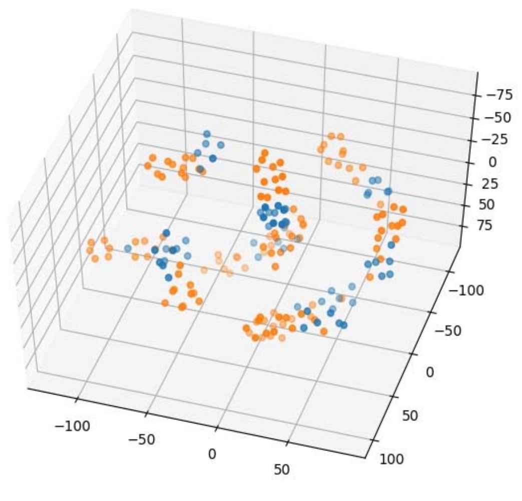
Fig. 5. The visualization result of the graph features obtained on the MUTAG dataset by the CMPGNN.

analysis.

For our method, we use the Adam optimizer with an initial learning rate of 0.001, and the learning rate is decayed by half every 20 epochs. We stack three convolutional layers in the local message passing phase. Other hyper-parametersare as follows: the number of training epochs is 100; the candidate set of the dimension of node embeddings is {4,6,8,12,16}, and all the convolutional layers have the same dimension of node embeddings; the candidate set of the dimension of graph feature is {4,6,8,12,16}; batchsize is 20. All of the results are obtained under 10 fold cross validation.

C. *Results and Analysis*

We use the testing results reported in related literature [21, 22, 23, 24] for baselines comparison. Table II lists the experimental results of the CMPGNN, CMPGNN-gi and baselines. We observes that CMPGNN performs well on both bio-informatics and social network datasets, and it achieves the highest average accuracy on PTC, PROTEINS, IMDB-Binary and IMDB-Multi datasets. In addition, the new model ranks in the top 2 on the MUTAG dataset. By comparing the accuracy rates of CMPGNN-gi and CMPGNN on all datasets, it can be found that the performance of CMPGNN decreases when the global information is removed, which means that the global information plays an important role in CMPGNN.

We also analyze the distribution of local passing coefficients calculated in the CMPGNN. Fig. 4 lists the histograms of the local passing coefficients obtained on the test sets of the 6 datasets when the cross-validation is performed. As can be seen from Fig. 4, most of the coefficients are concentrated between 0.4 and 0.6, which means during the local message passing process most nodes selects nearly half of the amount of information passed by their neighbors.

According to Table I and Table II, we find that CMPGNN is more suitable for datasets with large number of samples and small average degree, compared with other GNN baselines.

For the MUTAG dataset, we collected the graph features of the test samples obtained at the last time step, and use t-SNE to visualize the global features. As illustrated in Fig. 5, we find that after dimensionality reduction, the data are linearly separable in each cluster.

V. Conclusion

We propose a new graph neural network CMPGNN for graph classification. Different from traditional MPNNs which only include the message passing phase for node embeddings updating, the new model contains two message passing phases: the local message passing phase for node embedding updating and the global message passing phase for graph feature updating. The above two message passing phases are executed alternatively. We evaluate the performance of the new model on 6 benchmark datasets. Experimental results show that compared with other 10 baselines, CMPGNN achieves the top-1 accuracy on 4 of 6 datasets. Although in this paper the new model is only used for graph classification tasks, the message passing phase can be decoupled from the graph classification task, which can be used for node classification, link prediction and spatio-temporal graph prediction.

Future work include: 1) Studying how to construct regularization term to ensure that node embeddings with global information can well maintain the structural information of the graph; 2) Studying how to apply the new framework to graph classification models based on pooling methods, such as Sortpool [4], Diffpool [25] and SAGpooling [26]; 3) Studying how to use attention mechanism to extract differentiated global information for each node.

ACKNOWLEDGMENT

This research is partially supported by the National Science Foundation of China (Grant No. 61562041); Jiangsu Provincial Science Foundation of China (Grant No. BK20171447); Jiangsu Provincial University Natural Science Research of China(Grant No. 17KJB520024).

REFERENCES

[1] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017.

[2] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations*, 2018.

[3] M. Niepert, M. H. Ahmed and K. Kutzkov, "Learning convolutional neural networks for graphs," *International Conference on Machine Learning*, 2016, pp. 2014-2023.

[4] M. Zhang, Z. Cui, M. Neumann and C. Yixin, "An end-to-end deep learning architecture for graph classification," *National Conference on Artificial Intelligence*, 2018, pp. 4438-4445.

[5] W. L. Hamilton, Z. Ying and J. Leskovec, "Inductive representation learning on large graphs," *Neural Information Processing Systems*, 2017 , pp. 1024-1034.

[6] K. Xu, W. Hu, J. Leskovec and S. Jegelka, "How powerful are graph neural networks," *International Conference on Learning Representations*, 2019.

[7] J. Gilmer, S. S. Schoenholz, P. Riley, O. Vinyals and G. E. Dahl, "Neural message passing for quantum chemistry," *International Conference on Machine Learning*, 2017, pp. 1263-1272.

[8] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[9] J. Bruna, W. Zaremba, A. Szlam and Y. Lecun, "Spectral networks and locally connected networks on graphs," *International Conference on Learning Representations*, 2014.

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

[11] J. Zhou, G. Cui, Z. Zhang, et al. , "Graph neural networks: a review of methods and applications," *arXiv preprint arXiv:1812.08434,* 2018.

[12] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *Empirical Methods in Natural Language Processing*, 2014, pp. 1724-1734.

[13] A. L. Maas, A Y. Hannun, and A. Y. Ng. " Rectifier nonlinearities improve neural network acoustic models," *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[14] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel and M. Neumann, "Benchmark data sets for graph kernels," http://graphkernels.cs.tu-dortmund.de, 2016.

[15] N. Shervashidze, S. V. Vishwanathan, T. Petri, K. Mehlhorn and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 488-495.

[16] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn and K. M. Borgwardt, "Weisfeiler-Lehman graph kernels," *Journal of Machine Learning Research*, 2011, pp. 2539-2561.

[17] K. M. Borgwardt and H. Kriegel, "Shortest-path kernels on graphs," *International Conference on Data Mining* 2005, pp. 74-81.

[18] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi. Kondor and Karsten M. Borgwardt, "Graph kernels," *Journal of Machine Learning Research, 11*(2) , 2010, pp. 1201-1242.

[19] P. Yanardag and S. V. Vishwanathan, "Deep graph kernels," *Knowledge Discovery and Data Mining*, 2015, pp. 1365-1374.

[20] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," Neural Information Processing Systems, 2016, pp. 1993-2001.

[21] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," *International Conference on Machine Learning*, 2018, pp. 2186-2195.

[22] M. D. Mallea, P. Meltzer and P. J. Bentley, "Capsule neural networks for graph classification using explicit tensorial graph representations," *arXiv preprint arXiv:1902.08399,* 2019.

[23] S. Verma and Z. Zhang, "Graph capsule convolutional neural networks," *arXiv preprint arXiv:1805.08090,* 2018.

[24] M. Fey, and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428,* 2019.

[25] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Neural Information Processing Systems*, 2018 pp. 4805-4815.

[26] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," *International Conference on Machine Learning* 2019, pp. 3734-3743.