Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 32(7): 4398–4415.
DOI: 10.3934/era.2024198
Received: 12 May 2024
Revised: 25 June 2024
Accepted: 01 July 2024
Published: 11 July 2024

Research article

A Dual-channel Progressive Graph Convolutional Network via subgraph sampling

Wenrui Guan[1,2] **and Xun Wang**[1,*]

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

[2] School of Automation, Jiangsu University of Science and Technology, Zhenjiang 212100, China

* **Correspondence:** Email: wx_just@126.com.

Abstract: Graph Convolutional Networks (GCNs) demonstrate an excellent performance in node classification tasks by updating node representation via aggregating information from the neighbor nodes. Note that the complex interactions among all the nodes can produce challenges for GCNs. Independent subgraph sampling effectively limits the neighbor aggregation in convolutional computations, and it has become a popular method to improve the efficiency of training GCNs. However, there are still some improvements in the existing subgraph sampling strategies: 1) a loss of the model performance caused by ignoring the connection information among different subgraphs; and 2) a lack of representation power caused by an incomplete topology. Therefore, we propose a novel model called Dual-channel Progressive Graph Convolutional Network (DPGCN) via sub-graph sampling. We construct subgraphs via clustering and maintain the connection information among the different subgraphs. To enhance the representation power, we construct a dual channel fusion module by using both the geometric information of the node feature and the original topology. Specifically, we evaluate the complementary information of the dual channels based on the joint entropy between the feature information and the adjacency matrix, and effectively reduce the information redundancy by reasonably selecting the feature information. Then, the model convergence is accelerated through parameter sharing and weight updating in progressive training. We select 4 real datasets and 8 characteristic models for comparison on the semi-supervised node classification task. The results verify that the DPGCN possesses superior classification accuracy and robustness. In addition, the proposed architecture performs excellently in the low labeling rate, which is of practical value to label scarcity problems in real cases.

Keywords: semi-supervised learning; Graph Convolutional Network; subgraph sampling; dual channel; clustering

1. Introduction

Graph-structured data is a non-Euclidean data that represents entity relationship through nodes and edges [1]. It has become the fundamental form of carrying a complex structure in various domains, such as the user directed connection in social networks and gene interactions in biology. In recent years, there has been seen a surge of research interests in utilizing Graph Neural Networks (GNNs) to handle graph-structured data [2]. As an important branch of GNNs, a Graph Convolutional Network (GCN) has demonstrated an excellent performance on downstream tasks such as node classification, graph classification, and link prediction by introducing convolutional operations [3].

GCN updates node representation by aggregating the neighbor nodes' information. In this way, the feature information propagates over network topology to node embedding, and then the node embedding learned as such is used in classification tasks [4]. However, GCN performs the convolutional operation by iteratively calculating whole nodes layer by layer, which makes the training process challenging [5]. The neighbor nodes grow exponentially in size, especially with larger or denser graph data.

To address the above problems, independent subgraph sampling that can efficiently enhance the training is widely used in GCN variants [6, 7]. Subgraph sampling divides the entire graph into a number of subgraphs by utilizing various algorithms, such as random wandering [8], node-ordered sampling [9], etc. Then, the GCN utilizes the subgraphs for batch learning to obtain the node embedding representation. In this way, the GCN performs convolution operation on smaller scale data, thereby reducing the demand for computing resource.

In general, subgraph sampling involves two core processes: the subgraph construction method and the subgraph training strategy. The construction method determines how to quickly and efficiently extract the essential information from the graph. The training strategy determines how to exploit subgraphs to achieve results comparable to the entire graph training [10]. However, the existing GCN variants based on subgraph sampling still suffer from two significant challenges.

- Challenge of model performance: Subgraph sampling exploits incomplete neighbor information during back-propagation, which compromises the accuracy of gradient estimation. Therefore, the model performances of the GCNs based on subgraph sampling suffers from the loss of accuracy, robustness, and convergence speed.
- Challenge of representation: The incomplete topology of the subgraph is susceptible to bias with downstream tasks, which makes the embedding representation deviate from the ground truth. Thus, only utilizing the topology of subgraphs can cause the compromise of the model representation.

To address the challenge of the model's performance, some sampling strategies randomly backfill a part of the connection information. For example, GraphSAINT [8], which is a random wandering sampling strategy, enables to capture higher-order dependencies among nodes while preserving the local structural information of the graphs, while Cluster-GCN [11] adopts a subgraph construction method based on graph clustering, which preserves the connection information by randomly adding edges among different clusters. It is noticed that the above strategies reduce the bias of the gradient estimation by randomly augmenting the node information for the subgraphs. Although random backfilling can improve the generalization ability of the model, it can also lead to instability and slow

convergence during the training process.

To address the challenge of representation, GCN variants construct the multi-channel architecture to further enhance the representation. For example, AM-GCN [12] extracts the node feature for constructing the multi-channel architecture, which is able to learn the suitable importance weights when fusing the topology and node feature information. The essence of multi-channel architecture depends on the complementary information among the multi-channel, which provides a more comprehensive perspective for the model [13, 14]. It can be inferred that improving the complementary information is an effective way to enhance the model's representation. However, the existing models rarely attempt to quantify the amount of information complementarity among the multi-channel.

In this paper, we propose a Dual-channel Progressive Graph Convolutional Network (DPGCN) via subgraph sampling. The DPGCN includes progressive subgraph sampling and a dual channel fusion module. It is noticed that the progressive network includes two senses: 1) progressiving merging of the clusters ensures that the connection information among the subgraphs is replenished; and 2) adopting a parameter-sharing strategy to progressively learning that leads to the acceleration of the convergence. Furthermore, to reduce the information overlap or redundancy in the dual-channel, we attempt to quantify the complementary information via the joint entropy between the original topology and the different feature topologies. Additionally, we compare the DPGCN with eight methods on four benchmark node classification datasets to verify the effectiveness and advantages of our proposed method. Our main contributions are as follows:

- We propose a novel progressive subgraph sampling method. This method effectively retains the connection information among the subgraphs, which improves the convergence speed.
- In order to construct dual-channel module with low information redundancy, we attempt to quantify the complementary information based on the joint entropy of different channel topologies (i.e., adjacency matrices).
- Our proposed model demonstrates an excellent performance in semi-supervised node classification tasks comparing with eight representative models.

2. Related works

The excellent performance of a GCN in downstream tasks prompts a lot of extended work, which focuses on the improvement of training efficiency and representation. Our proposed model is constructed to fulfill the above two targets. Specifically, we utilize progressive training via subgraph sampling to improve model's efficiency and to enhance the model's representation via the dual-channel architecture.

2.1. Subgraph sampling

The traditional GCN utilizes the entire graph for sequential iterative computation that leads to the complexity of the training process and a growth of the computational cost. To address the challenges, scholars proposed a series of independent subgraph sampling methods to reduce the computational cost while maintaining the essential properties of the graph data. For example, Cluster-GCN [11] adopts the METIS graph clustering method to divide the whole graph into multiple subgraphs, and then trains

the full batch on each subgraph, which effectively reduces the memory requirement and improves the training efficiency. GraphSAINT [8] adopts random walk sampling to solve the neighbor explosion and improves the hardware utilization by significantly reducing the data traffic to the slow memory. IANS-GCN [9] utilizes local center node sampling to isolate the high-order neighbors and builds a deeper network to learn more complex graph representations. Although these subgraph sampling methods already outperform traditional GCNs in classification accuracy, the weak connection information of the subgraphs still poses the challenges of stability and convergence speed.

2.2. Multi-channel learning

The core of multi-channel learning is utilizing the complementary information of different channels to obtain comprehensive embedding representation. For example, AM-GCN [12] extracts the specific and common embeddings from node features, topological structures, and their combinations simultaneously, and uses the attention mechanism to learn the adaptive importance weights of the embeddings. MAGCN [15] aggregates the node features from different hops of neighbors using the multi-view topology of the graph and an attention mechanism. In addition, MAGCN conducts a theoretical analysis of representation using a rigorous mathematical proof, which demonstrates its learning potential. PAGCN [16] leverages the principles of contrastive learning and attention mechanisms to efficiently learn and fuse the node representations of perturbation augmentation graphs and input graph. Adding multiple channels to enhance the model's representation is certainly a feasible strategy. However, blindly adding multiple channels is not enough to compensate for the efficiency loss caused by an increase in the parameters and computing resources. In view of this, we attempt to use joint entropy to evaluate the complementary information of the multiple channels, and then build a lightweight dual-channel fusion module.

3. A Dual-channel Progressive Graph Convolutional Network via subgraph sampling

3.1. Preliminaries

A graph data $G = (V, E, A)$ is composed of $N = |V|$ nodes and $|E|$ edges. The edges represent the relationship among nodes, which is represented by the adjacency matrix A. A is an N-dimensional sparse matrix. If nodes i and j are related, $A(i, j)$ is 1, otherwise it is 0. In addition, each node has an F-dimensional feature vector, where X is the feature matrix of the node.

3.2. Overview of framework diagram

The framework of the traditional subgraph sampling is shown in Figure 1. Its weak connection information of the subgraphs poses the challenges of stability and convergence speed.

The DPGCN framework is shown in Figure 2. The graph is reconstructed into dual-channel subgraphs through a clustering sampler. Then, the subgraphs are progressively trained to obtain the embedding representation. Finally, the dual-channel embedding representations are fused to obtain the classification result.

In detail, the sampling uses the clustering algorithm to divide the nodes into multiple clusters. Then, the clusters are sequentially merged to obtain the progressive set of nodes. Finally, the DPGCN extracts the sub-matrices from the adjacency and distance matrices for the topology channel and the

feature channels.

Progressive training adopts a parameter sharing strategy, which preserves the parameter of the previous subgraph to the next subgraph. Thus, only fine-tuning of the parameters in subsequent training can obtain a superior embedding representation. In addition, we progressively introduce updated sample weights in the loss function to revise the misclassified samples in subsequent training.

Compared with the traditional architecture, DPGCN has three main aspects: 1) it adopts a novel subgraph sampling strategy, which allows the connection information to be retained in the subsequent subgraphs; 2) it constructs a dual-channel architecture in subgraphs to fuse the topology and node features, which enhances the representation of subgraphs; and 3) it employs the progressive training strategy, including the updating of the sample weights, as shown by the yellow line. Constantly updating the samples allows the misclassified samples to be revised in the subsequent training. Moreover, subsequent training utilizes the parameters of the previous subgraph and only needs to be fine-tuned, as shown by the blue line in Figure 2.

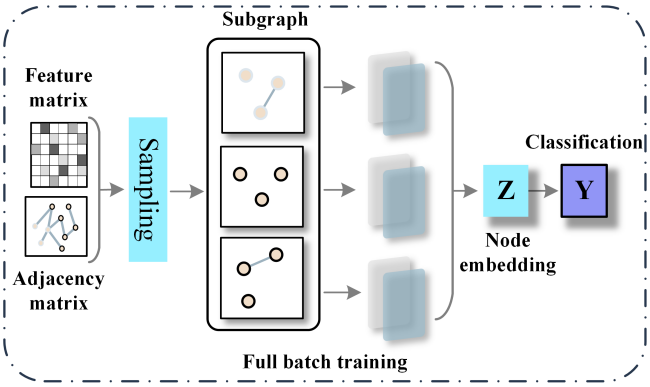


Figure 1. Traditional subgraph sampling framework.

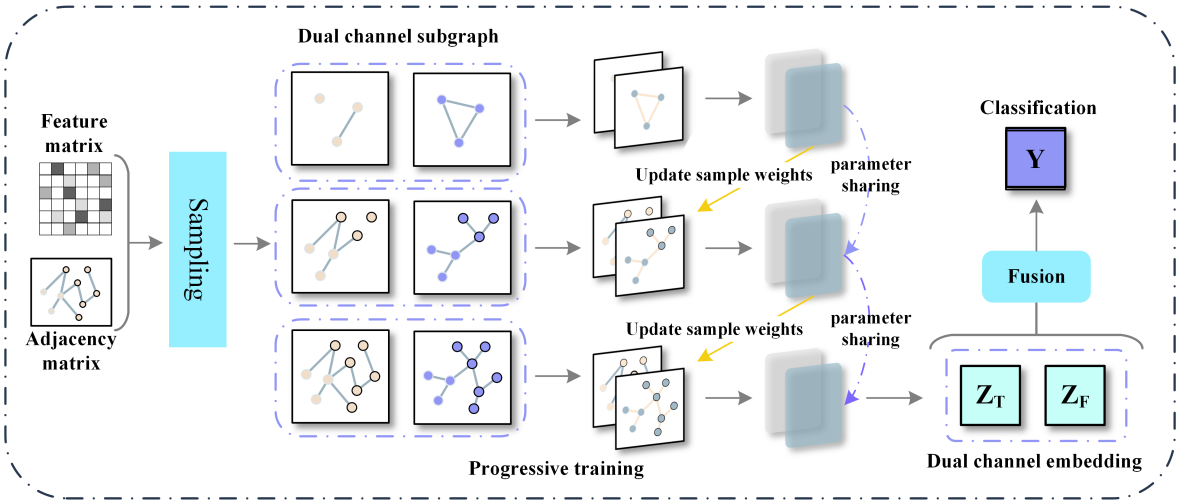


Figure 2. The framework of DPGCN.

3.3. Progressive sampling

Using clustering algorithm divides the set of nodes to maximize the similarity of the sample points inside the class and minimize the similarity of the sample points among the classes. The set of nodes V is divided into K clusters and the adjacency matrix A is divided into K^2 sub-matrices:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1K} \\ \vdots & \ddots & \vdots \\ A_{K1} & \cdots & A_{KK} \end{bmatrix}. \tag{3.1}$$

For any diagonal block, $A_{tt}, (t \leq K)$, which denotes the edge E_{tt} inside the set of nodes V_t. For the remaining matrix blocks, A_{t1} denotes the edge E_{t1} between the set of nodes V_t and V_1, thus G is divided into the following:

$$G = [G_1, ..., G_K] = [(V_1, E_{11}), ..., (V_K, E_{KK}), E_{1K}, ...E_{K1}], \tag{3.2}$$

where $(E_{1K}, ..., E_{K1})$ contains the connection information among different sets of nodes.

In order to retain the correlation among subgraphs, we adopt the following progressive subgraph sampling strategy: the strategy extracts the set of nodes V_t to construct the first subgraph G_1. Then, it randomly chooses the set (V_c, E_{cc}, A_{cc}) and the edge (E_{ct}, E_{tc}) to merge them into the previous subgraph G_2. The above steps are continued until all subgraphs have been successfully merged. We can obtain K progressive subgraphs $G = (G_1, G_2, ..., G_K)$. According to the nodes of different subgraphs, the corresponding features and labels are $X = (X_1, X_2, ..., X_K), Y = (Y_1, Y_2, ..., Y_K)$.

3.4. Evaluation and selection of feature information

While extracting different feature information, the new channel should avoid excessive information redundancy in order to effectively supplement the information of the original channel. Therefore, a reasonable selection of the feature information is a previous assurance to achieve the comprehensiveness and efficiency of the multi-channel architecture. Therefore, we attempt to utilize the joint entropy in the information theory to evaluate the amount of complementary information provided by different feature information.

Joint entropy [17] is used to measure the amount of information in multiple stochastic systems under a joint probability distribution. A higher entropy means that it contains more information [18]. The definition of the joint entropy $H(P, A)$ of the adjacency matrix A and the new channel P is as follows:

$$H(P, A) = -\sum_{\mathscr{X}} \sum_{\mathscr{Y}} p(x, y) \log p(x, y). \tag{3.3}$$

This equation is used to observe the amount of information obtained by a system integrated into a new view. For a further analysis, we transform the equation as follows:

$$H(P, A) = H(A) + H(P|A), \tag{3.4}$$

where $H(A)$ represents the information entropy of the original topological view A. The conditional entropy $H(P|A)$ represents the amount of information obtained when obtaining a new view P. The amount of information added by the introduced new view can be represented by the joint entropy.

Therefore, we use the general methods of distance or similarity measures based on the feature information, such as the Euclidean distance, the Manhattan distance, the Cosine similarity, and the Pearson correlation coefficient, to calculate their joint entropy. Specifically, we calculate different similarity matrices or matrices based on four methods for a pairwise nodes' features. Then, the obtained matrices are normalized to be limited between (0, 1). The obtained results are shown in Table 1.

Table 1. Joint entropy of different methods.

	Cora	ACM	Citeseer	UAI	Texas	Chameleon
Euclidean distance	**0.6412**	**0.7243**	**0.7028**	**0.3095**	0.6923	**0.9790**
Cosine similarity	0.0170	0.0775	0.0071	0.2322	**0.7825**	0.0172
Pearson correlation	0.0179	0.0309	0.0024	0.0259	0.7807	0.0016
Manhattan distance	0.0136	0.0285	0.0570	0.1088	0.6408	0.5382

It is observed that the joint entropy of the Euclidean distance matrix is superior, (i.e., the amount of complementary information provided by the Euclidean distance matrix is higher). This means that when considering both the original adjacency matrix and the Euclidean distance matrix, the model has a greater amount of information. The smaller joint entropy of the other methods indicates that the other method's feature information may have a significant information overlap. Thus, the fusion may not bring a significant enhancement for the representation.

After evaluation, we choose the Euclidean matrix to represent the geometric structure to construct dual channels for fusion learning. In addition, we introduce the neighborhood radius r, (i.e., the node's distance beyond the neighborhood radius r is set as the maximum value). The purpose is to flexibly adjust the learning range of the model's feature channels by adjusting the neighborhood radius r. Finally, the distance matrix is added to multiple progressive subgraphs G, where D denotes the distance matrix.

$$\mathbf{G} = \begin{bmatrix} (V_1, E_1, A_1) & (V_2, E_2, A_2) \ldots & (V, E, A_K) \\ (V_1, E_1, D_1) & (V_2, E_2, D_2) \ldots & (V, E, D_K) \end{bmatrix} \tag{3.5}$$

3.5. Progressive training via GCN

For the L-layer GCN, in each layer of convolutional computation, the model is expressed as follows:

$$\mathscr{Z}^{l+1} = A^{'} X^{l} W^{l}, \quad X^{l+1} = \sigma(\mathscr{Z}^{l+1}), \tag{3.6}$$

where $A^{'}$ is the normalized and regularized adjacency matrix that represents the topological structure. X^l is the embedding of the previous layer. W^l is the feature transformation matrix of the current layer. The feature transformation matrix serves downstream tasks after learning. $\sigma(\cdot)$ is an activation function, which uses the **Relu** activation function.

The model initializes the sample weight of any node as $\alpha = 1/|V_1|$. Thus, we obtain the sample weight matrix of G_1 as $\mathbf{S}_1 = [\alpha_1^1, \alpha_2^1, \cdots, \alpha_{|V_1|}^1]$. Next, the subgraph G_1 is inputted into the dual-channel to obtain the embedding representation. Thus, the embedding representation of the dual channel can be obtained as follow:

$$\begin{bmatrix} \mathscr{Z}_t^2 \\ \mathscr{Z}_f^2 \end{bmatrix} = \begin{bmatrix} A_1'\sigma\left(A_1'X_1W_t^1\right)W_t^2 \\ D_1'\sigma\left(D_1'X_1W_f^1\right)W_f^2 \end{bmatrix}, \tag{3.7}$$

where the feature transfer matrix of the original channel is (W_t^1, W_t^2). The feature transfer matrix of the added channel is (W_f^1, W_f^2). The topology channel $\mathscr{Z}_t^2$ focuses on the local structural features of the nodes and their direct connections, while the feature channel $\mathscr{Z}_f^2$ focuses on extracting the unstructured features of the distance matrix. To achieve a faster computation, the model adopts maximum pooling to fuse the embedding representation of the two channels:

$$\mathbf{Z}_i = max(\mathscr{Z}_t^2, \mathscr{Z}_f^2). \tag{3.8}$$

Maximum pooling extracts the most significant node features on the topology channel and the feature channel to obtain the embedding representation $\mathbf{Z}$. In contrast to the attention fusion mechanism, maximum pooling ensures that the most significant feature information is fused into the embedding representation without increasing the complexity condition of the model. The resulting embedding representation is passed through the last layer to further integrate the information to get the final node classification. W_3 is the feature transformation matrix for the downstream task. This results in the prediction of the subgraph G_1 as follows:

$$output_1 = \sigma(A_1'Z_1W^3), \overline{\mathbf{Y}}_1 = softmax(output_1). \tag{3.9}$$

Additionally, the loss of the subgraph G_1 is follows:

$$\mathscr{L}_{G_1} = -\sum_{j\in V_i} \alpha_j^1 y_j \log(\bar{y}_j), \alpha_j^1 \in \mathbf{S}_1. \tag{3.10}$$

By training the subgraph G_1, we obtain the set of misclassified nodes Q_1. Thus, we get the pseudo-loss ε_1 based on the misclassified samples of the subgraph G_1:

$$\varepsilon_1 = \frac{1}{2\,|Q_1|}\sum_{t\in Q_1}(1 - output_1(x_t, \bar{y}_t) + output_1(x_t, y_t)), \tag{3.11}$$

where $output_1(x_t, \bar{y}_t)$ is the confidence level of the incorrect category, $output_1(x_t, y_t))$ is the confidence level of the correct category, and the range of pseudo-loss ε_1 is $(0, 0.5)$. The sample weight α_t^2 is as follows:

$$\alpha_t^2 = \begin{cases} \frac{1}{|V_2|}\cdot\frac{\varepsilon_1}{1-\varepsilon_1}, t \in (V_1 - \mathbf{Q}_1) \\ \frac{1}{|V_2|}, t \notin (V_1 - \mathbf{Q}_1) \end{cases}, t \in V_2, \tag{3.12}$$

where $(V_1 - Q_1)$ denotes the set of nodes correctly classified in the classification of subgraph G_1. By weight updating, the misclassified samples Q_1 are corrected in the loss calculation for subgraph G_2.

In order to clearly express the algorithmic process, the following is the pseudo-code of the DPGCN:

Algorithm 1: DPGCN

Input: The graph dataset $G = (V, E, A)$; The number of node classes K; GCN encoder $g(\cdot)$; The maximal number of iterations h; The max patience p; The neighborhood radius r;

Output: The final node classification result $\overline{\mathbf{Y}}$

Using K-means to cluster feature matrices $\mathbf{X}$ and obtain $[V_1, V_2, ..., V_K]$;

Using the node set V_i and the nearest neighbor radius r to calculate the distance matrix $[D_1, D_2, ..., D_K]$;

Integrate to obtain the sequential training sample set $\mathbf{G} = [\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_K]$

for $i = 1; i \leq K; i++$ **do**

  for $j = 1; j \leq h; j++$ **do**

    Calculate node embeddings $\mathscr{Z}_t^2$, $\mathscr{Z}_f^2$ using the encoder via Eq (3.7);

    Obtain fused node embeddings $\mathbf{Z}_i$ using global max pooling via Eq (3.8);

    Calculate the loss function of subgraph $\overline{G}_i$ via Eq (3.10) and update patience;

    if *patience* $\geq$ *max patience* **then**

      Break

    end

    Update parameters $\mathbf{W}$ by applying gradient descent to minimize $\mathscr{L}_{\overline{G}_i}$

  end

  Calculate the sample weight S_{i+1} for subgraph G_{i+1} via Eq (3.12)

  Save the optimal model $\mathbf{W}$ to the next subgraph $\overline{G}_{i+1}$ for training;

end

Calculate the classification results $\overline{\mathbf{Y}}$

Return The classification results $\overline{\mathbf{Y}}$

4. Experiments

4.1. Experiment setup

4.1.1. Datasets

Our proposed model was evaluated on 4 real-world datasets, and the details of the datasets are shown in Table 2.

Table 2. Datasets used in the experiments.

Data	Nodes	Edges	Classes	Features	Test	Label rate
Cora	2708	5429	7	1433	1000	0.05
Citeseer	3327	4732	6	3703	1000	0.05
Squirrel	5201	2089	4	217,073	1000	0.05
Acm	3025	26,256	3	1870	1000	0.03

4.1.2. Baselines

In this paper, the DPGCN is compared with 8 models to verify the effectiveness of the proposed method, and the comparison algorithms are described as follows:

GCN [3]: A classic semi-supervised graph convolutional network model that learns node representation by aggregating information from neighbors.

GAT [19]: A graph neural network model that uses an attention mechanism to aggregate the node features.

SGC [20]: A simplified linear model of graph convolution. The resulting function is collapsed into a single linear transformation by iteratively removing nonlinearities between the GCN layers.

Cluster-GCN [11]: A model that reduces computation by dividing the graph data into multiple subgraphs and independently performs graph convolution operations on each subgraph.

MOGCN [21]: The model uses an encoder based on multi-scale information fusion. MOGCN constructs a multilevel adjacency matrix and uses the GCN to obtain the output of different adjacency matrices.

PAGCN [16]: A framework that utilizes contrast learning and attention mechanisms to efficiently learn and fuse the node representation of perturbation-enhanced graphs and input graphs.

SGAT [22]: A Sparse Graph Attention Network (SGAT) that integrates sparse attention mechanisms into a graph attention network by regularizing the edges of the graph. A 2-head indicates the use of a two-layer SGAT model.

hLGC [23]: A simplified graph convolution operator which adds complexity and nonlinearity by linear transformations or control nonlinearities, where a series of simple graph convolution operators are proposed. An hLGC enhanced linear graph convolution is one of them.

4.1.3. Parameters setting

For a complete evaluation of the models, the above comparison models use the parameters suggested in their papers to ensure an optimal performance. For the DPGCN, each channel uses two layers of GCN encoders. The model uses the Adam optimizer with a learning rate of 0.001 to 0.005. The discard rate is 0.5. The weight decay setting is 5e-4. The neighborhood radius to constructe the feature information is 0.7. For all methods, the paper is run 20 times. The experimental configuration is shown in Table 3.

Table 3. Experimental environment configuration.

Component	Basic configuration
CPU	Intel(R) Core(TM) i5-8300H
Memory	32 GB DDR4
Graphics card	NVDIA RTX 1050
System	Windows10
CUDA	Cuda 11.6
Environment library	Python3.8; PyTorch 1.13

4.2. Experimental results and analysis

4.2.1. Accuracy analysis of node classification

We compare the DPGCN with eight different models to compare their accuracy and the experimental results are shown in Table 4, where the best results are highlighted in bold.

Table 4. ACC of node classification tasks.

Model	Datasets			
	Cora	ACM	Citeseer	Pubmed
DPGCN	**0.832**	**0.912**	0.716	**0.816**
GCN	0.817	0.878	0.704	0.792
GAT	0.828	0.874	**0.726**	0.791
SGC	0.812	0.863	0.719	0.789
PAGCN	0.831	0.909	0.704	0.793
Cluster-GCN	0.812	0.901	0.714	0.807
hLGC	0.830	0.892	0.703	0.808
SGAT-2head	0.819	0.902	0.706	0.802
MOGCN	0.824	0.901	0.724	0.792

According to the results in Table 4, the DPGCN performs well in most of the datasets, improving by 0.15–2.1%, 0.3–4.9%, and 0.83–2.64% on Cora, ACM, and Pubmed, respectively. Thus, the DPGCN demonstrates a high accuracy and an excellent performance in the semi-supervised node classification task, indicating the effectiveness of the model. Compared with the classical subgraph sampling model Cluster-GCN, the DPGCN retains the connection information among the subgraphs, which makes the model classification accuracy higher. Therefore, this progressive sampling method provides a new solution to the plague of accuracy loss in traditional sampling models.

In addition, compared with the PAGCN and the MOGCN, which are two representative multi-channel models, the DPGCN exhibits a higher accuracy rate. It verifys that the dual-channel strategy is effective and the complementary information is crucial. Reasonably fusing different feature information can help to enhance the representation.

4.2.2. Convergence speed analysis

To further explore the performance of the DPGCN, five representative models are used for comparison on the Cora dataset. The number of iterations is set to 500. The experimental results are shown in Figures 3 and 4.

Observing the result, the DPGCN demonstrates its fast convergence speed in the Cora dataset, which is a significant advantage. In particular, the DPGCN is able to achieve a high level of classification accuracy after about 60 iterations. The fast convergence is partly attributed to the parameter sharing mechanism. Parameter sharing not only reduces the number of parameters and the computational burden, but also effectively transfers the information during the multiple subgraphs.

In addition, progressive sample weights also play a key role in training the DPGCN. By dynamically adjusting the sample weights, it can ensure that the model focuses on the samples that are difficult to classify. As the iteration proceeds, the sample weights are continuously adjusted so that the model can gradually adapt to learn the whole data distribution.

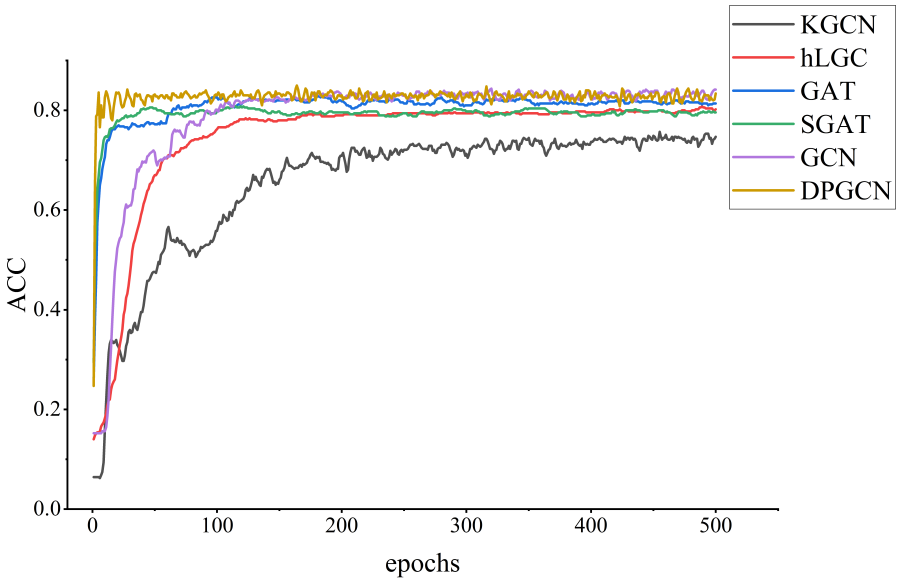


Figure 3. Convergence curve of ACC.

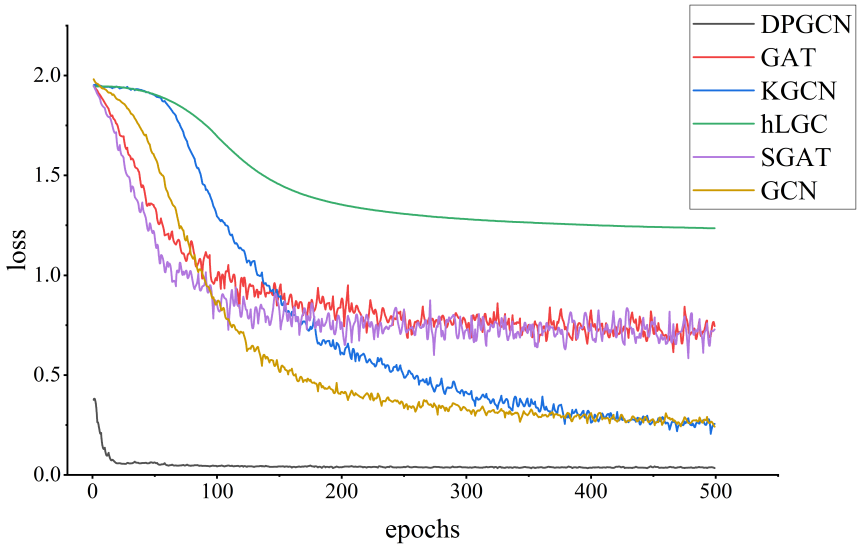


Figure 4. Convergence curve of LOSS.

4.2.3. Node classification analysis of low label rate

In real scenarios, models often face the situation of high cost or scarcity in labeling the data. Therefore, we select Cora, Citeseer, ACM, and Pubmed data for low label rate experiments. The experimental results are shown in Table 5.

Table 5. Node classification task of low label rate.

	Label number	DPGCN	GCN	GAT	SGC	PAGCN	Cluster-GCN	hLGC	SGAT-2head	MOGCN
Cora	14	**0.686**	0.438	0.445	0.409	0.476	0.436	0.453	0.457	0.419
	28	**0.729**	0.623	0.668	0.617	0.721	0.603	0.652	0.671	0.715
Citeseer	12	**0.448**	0.247	0.331	0.243	0.289	0.232	0.33	0.281	0.281
	18	0.592	0.436	0.584	0.423	**0.604**	0.498	0.574	0.493	0.589
ACM	9	**0.671**	0.518	0.563	0.501	0.567	0.502	0.552	0.562	0.566
	15	**0.853**	0.656	0.646	0.446	0.735	0.653	0.677	0.712	0.681
Pubmed	9	**0.625**	0.439	0.42	0.457	0.51	0.399	0.432	0.402	0.402
	15	**0.776**	0.605	0.566	0.558	0.635	0.622	0.625	0.624	0.685

Observing the experimental results, the DPGCN performs more outstandingly in low label rate experiments. Compared to the other methods, the result not only demonstrates the adaptability and robustness of DPGCN to different data distributions, but also shows the great potential of the model in dealing with sparse labels.

The superior performance of the DPGCN in low labeling rate experiments verifies that the model has a strong representation. Therefore, the strategy of the DPGCN effectively responds to the challenge of insufficient representation in traditional subgraph sampling models.

4.2.4. Ablation experiment

The DPGCN demonstrates an excellent performance on different data, especially with more outstanding performance in the low label rate experiments. We conduct ablation experiments to further explore the contribution of each component in the DPGCN. The model contains two main components: progressive subgraph sampling and dual-channel fusion module. Therefore, we set up only progressive subgraphs to get variant 1, DPGCN-P, and only dual-channel fusion to obtain variant 2, DPGCN-D. In this paper, we set up different label rates on Cora and Citeseer to compare the two variants with the original model. The experimental results obtained are shown in Figure 5.

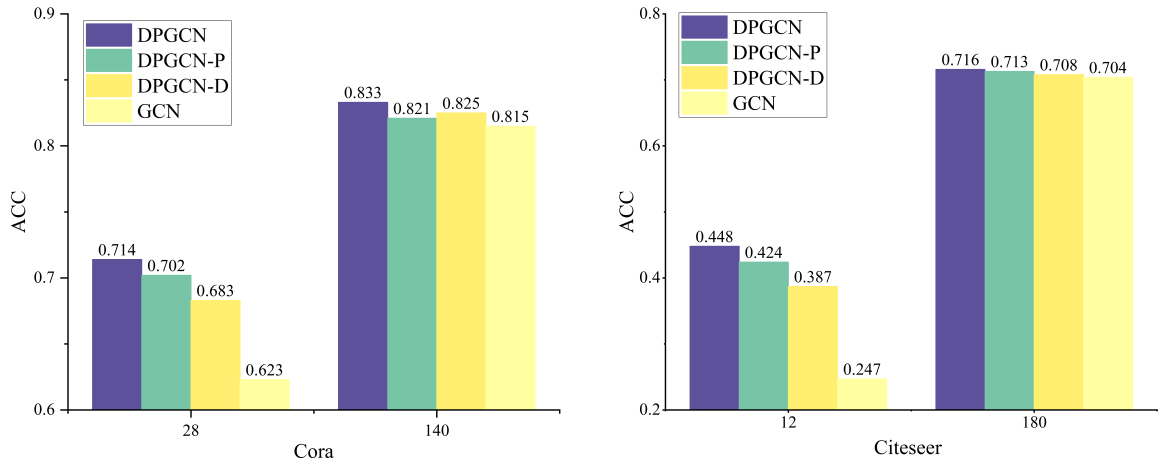


Figure 5. The ablation experiment.

The results show that the progressive subgraph sampling (DPGCN-P) component outperforms the dual-channel fusion module with low labeling rates. The reason for the phenomenon is that when

the training samples are small, adding feature channels provides less complementary information. the DPGCN-P is able to play a major role by capturing the features of the limited labeled data and fully utilizing the neighbor node features. When the number of training samples increases, the feature channel is able to provide richer information. In this case, the dual-channel information fusion plays an important role. The model obtains a superior embedding representation by capturing the feature information of nodes on different channels.

Overall, two main components are utilized to the advantage in different situations. The DPGCN effectively combines the two components to obtain an excellent representation.

4.2.5. Parameter sensitivity analysis

The DPGCN model involves a hyperparameter, namely the neighborhood radius r. In order to explore the sensitivity of the parameter, the Cora and ACM datasets are selected for the experiment. The number of training nodes are 14 and 9. The interval of the neighborhood radius is set to [0.1, 0.9]. The step size is 0.1. The results of the experiment are shown in Figure 6. The blue points in the box-and-line plots represent the average value of the accuracy under this neighborhood radius.

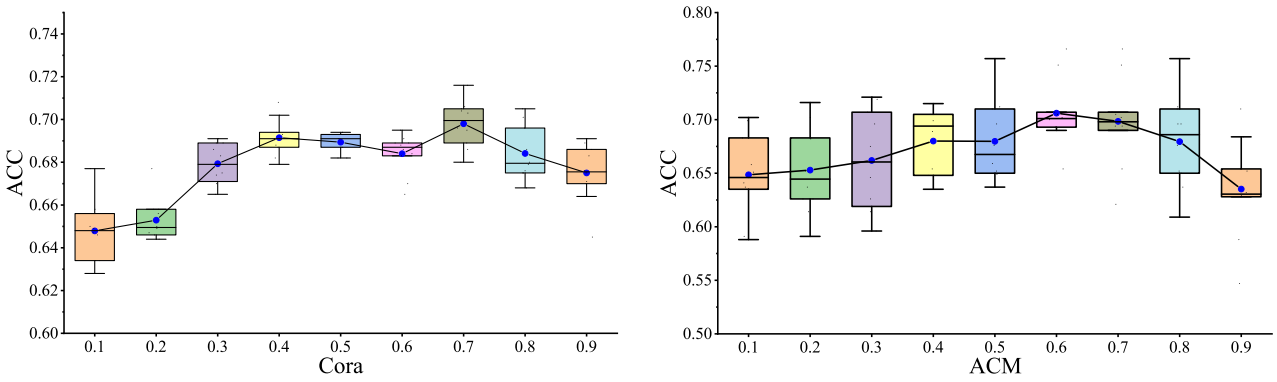


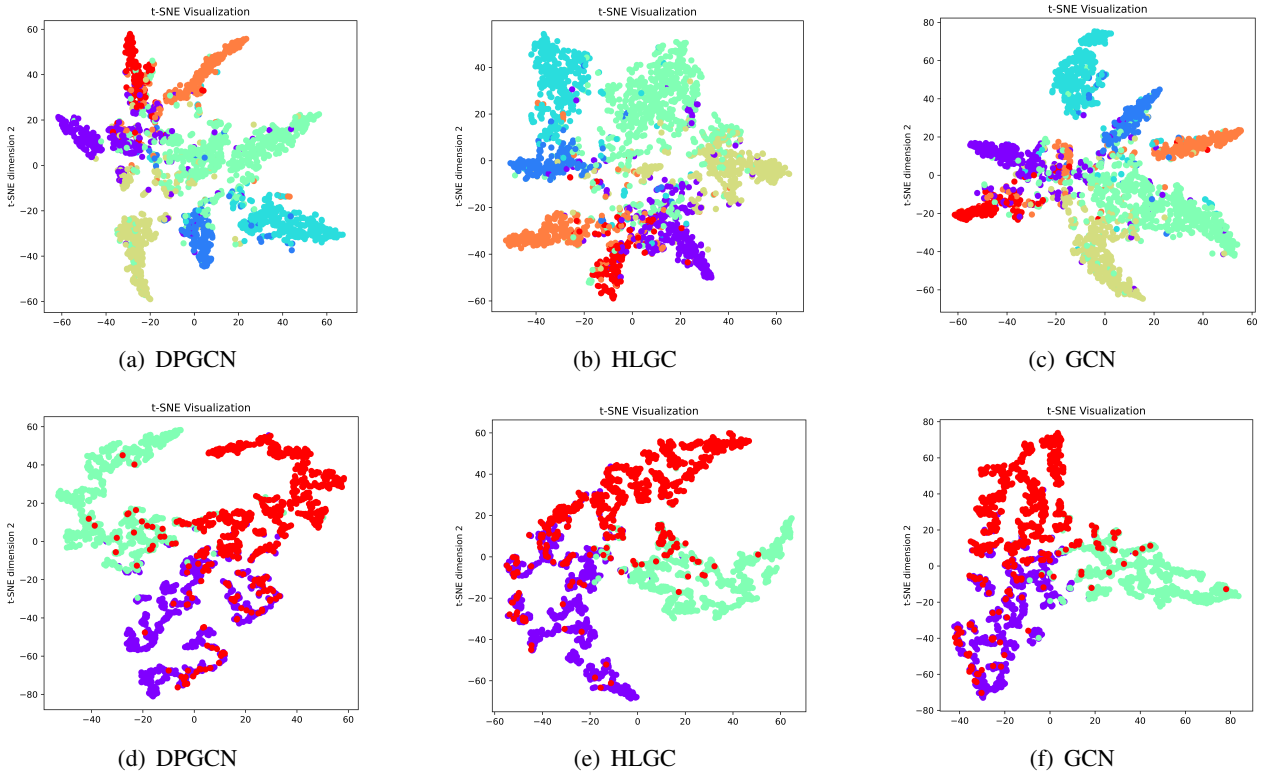
Figure 6. Parameter sensitivity analysis.

To summarize there are two main reasons that make the parameter insensitive: 1) the neighborhood radius only affects the information in the feature map channel that does not affect the model globally; and 2) even though the neighborhood radius is a fixed value at the setup, the parameter applies to all subgraphs. This means that as the subgraphs incrementally increase, the neighborhood radius actually increases as well. This gradual increase makes the neighborhood radius compatible with the subgraph size so that the appropriate local information can be adaptively captured on subgraphs of different sizes, making the whole model insensitive to changes in the neighborhood radius.

4.2.6. Visualization analysis

The Cora and ACM datasets are chosen for the visualized using T-SNE (t-distributed stochastic neighbor embedding) dimensionality reduction. The classification results are obtained and shown in Figure 7. The final node classification results of the different algorithms on the Cora and ACM datasets can be clearly observed in this figure, where the first row is visualized for the Cora dataset and the second row is the ACM dataset.

Observation of the visualization results shows that the DPGCN is able to provide clearer category boundaries and exhibits more obvious segmentation effects in the visualization. This further validates that the DPGCN successfully integrates the information from the dual channels and achieves a higher accuracy in the node classification task.



(a) DPGCN (b) HLGC (c) GCN

(d) DPGCN (e) HLGC (f) GCN

Figure 7. Visualization analysis.

5. Conclusions and future works

We designed a DPGCN via subgraph sampling. The model utilizes clustering sampling and geometric information in the feature space for effective learning. Through parameter sharing and updating sample weights, DPGCN obtained better node classification results. In subgraph sampling, we sequentially merged clusters to ensure maximum embedding utilization. In the feature space fusion, the joint entropy in information theory was utilized to evaluate the feature information. The experimental results compared the DPGCN with current state-of-the-art graph neural networks. Its superiority on several benchmark datasets was verified for the semi-supervised node classification task. It was noticed that DPGCN is particularly outstanding in label-poor environments, which shows a strong stability and an excellent accuracy performance.

In future studies, we will continue to explore the role of geometric information in feature space and try to explain this principle in the field of flow learning. Additionally, there are also some improvements to the model, such as how to improve the adaptive selection of the optimal complementary feature information to further enhance the expressive power of the model.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by National Natural Science Foundation of China (62076111), Jiangsu Province Graduate Research and Practice Innovation Program Project (KYCX_233883).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. L. F. R. Ribeiro, P. H. P. Saverese, D. R. Figueiredo, Struc2vec: Learning node representation from structural identity, in *KDD '17 KDD '17: The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2017), 385–394. https://doi.org/10.1145/3097983.3098061
2. Y. Zhang, S. Pal, M. Coates, D. Ustebay, Bayesian graph convolutional neural networks for semi-supervised classification, in *Proceedings of the AAAI conference on artificial intelligence*, AAAI, **33** (2019), 5829–5836. https://doi.org/10.1609/aaai.v33i01.33015829
3. T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, preprint, arXiv:1609.02907.
4. N. Yadati, V. Nitin, M. Nimishakavi, P. Yadav, A. Louis, P. Talukdar, et al., NHP: Neural hypergraph link prediction, in *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*, ACM, (2020), 1705–1714. https://doi.org/10.1145/3340531.3411870
5. D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2016), 1225–1234. https://doi.org/10.1145/2939672.2939753
6. S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: A comprehensive review, *Comput. Soc. Netw.*, **6** (2019), 11. https://doi.org/10.1186/s40649-019-0069-y
7. X. X. Wang, X. B. Yang, P. X. Wang, H. L. Yu, T. H. Xu, SSGCN: A sampling sequential guided graph convolutional network, *Int. J. Mach. Learn. Cyb.*, **15** (2024), 2023–2038. https://doi.org/10.1007/s13042-023-02013-2
8. H. Q. Zeng, H. K. Zhou, A. Srivastava, R. Kannan, V. Prasanna, GraphSAINT: Graph sampling based inductive learning method, preprint, arXiv:1907.04931.
9. Q. Zhang, Y. F. Sun, Y. L. Hu, S. F. Wang, B. C. Yin, A subgraph sampling method for training large-scale graph convolutional network, *Inf. Sci.*, **649** (2023), 119661. https://doi.org/10.1016/j.ins.2023.119661

10. S. T. Zhong, L. Huang, C. D. Wang, J. H. Lai, P. S. Yu, An autoencoder framework with attention mechanism for cross-domain recommendation, *IEEE Trans. Cybern.*, **52** (2020), 5229–5241. https://doi.org/10.1109/TCYB.2020.3029002

11. W. L. Chiang, X. Q. Liu, S. Si, Y. Li, S. Bengio, C. J. Hsieh, Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks, in *KDD'19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, (2019), 257–266. https://doi.org/10.1145/3292500.3330925

12. X. Wang, M. Q. Zhu, D. Y. Bo, P. Cui, C. Shi, J. Pei, AM-GCN: Adaptive multi-channel graph convolutional networks, in *KDD'20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, (2020), 1243–1253. https://doi.org/10.1145/3394486.3403177

13. L. Zhao, Y. J. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, et al., T-GCN: A temporal graph convolutional network for traffic prediction, *IEEE Trans. Intell. Trans. Syst.*, **21** (2019), 3848–3858. https://doi.org/10.1109/TITS.2019.2935152

14. X. B. Hong, T. Zhang, Z. Cui, J. Yang, Variational gridded graph convolution network for node classification, *IEEE/CAA J. Autom. Sin.*, **8** (2021), 1697–1708. https://doi.org/10.1109/JAS.2021.1004201

15. K. X. Yao, J. Y. Liang, J. Q. Liang, M. Li, F. L. Cao, Multi-view graph convolutional networks with attention mechanism, *Artif. Intell.*, **307** (2022), 103708. https://doi.org/10.1016/j.artint.2022.103708

16. Q. H. Guo, X. B. Yang, F. J. Zhang, T. H. Xu, Perturbation-augmented graph convolutional networks: A graph contrastive learning architecture for effective node classification tasks, *Eng. Appl. Artif. Intell.*, **129** (2024), 107616. https://doi.org/10.1016/j.engappai.2023.107616

17. T. H. Chan, R. W. Yeung, On a relation between information inequalities and group theory, *IEEE Trans. Inf. Theory*, **48** (2002), 1992–1995. https://doi.org/10.1109/TIT.2002.1013138

18. M. Madiman, P. Tetali, Information inequalities for joint distributions, with interpretations and applications, *IEEE Trans. Inf. Theory*, **56** (2010), 2699–2713. https://doi.org/10.1109/TIT.2010.2046253

19. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, preprint, arXiv:1710.10903.

20. F. L. Wu, A. Souza, T. Y. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, (2019), 6861–6871.

21. J. Wang, J. Q. Liang, J. B. Cui, J. Y. Liang, Semi-supervised learning with mixed-order graph convolutional networks, *Inf. Sci.*, **573** (2021), 171–181. https://doi.org/10.1016/j.ins.2021.05.057

22. Y. Ye, S. H. Ji, Sparse graph attention networks, *IEEE Trans. Knowl. Data Eng.*, **35** (2021), 905–916. https://doi.org/10.1109/TKDE.2021.3072345

23. L. Pasa, N. Navarin, W. Erb, A. Sperduti, Empowering simple graph convolutional networks, *IEEE Trans. Neural Networks Learn. Syst.*, **35** (2023), 4385–4399. https://doi.org/10.1109/TNNLS.2022.3232291

AIMS Press