*Journal of Network Intelligence*
**Taiwan Ubiquitous Information**
 ISSN 2414-8105 (Online)
Volume **8**, Number **2**, May **2023**

# Collaborative Weighting for Graph Convolutional Networks

Yong Chen

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
2122031117@s.xmut.edu.cn

Xiao-Zhu Xie*

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
xz4xxz@gmail.com

Wei Weng

College of Computer and Information Engineering
Xiamen University of Technology
Fujian Key Laboratory of Pattern Recognition and Image Understanding
Xiamen, 361024, P. R. China
xmutwei@163.com

Shan-Dan Zhang

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
zhangshand@163.com

Tong Li

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
2022031405@stu.xmut.edu.cn

*Corresponding author: Xiao-Zhu Xie

Received September 24, 2022, revised November 17, 2022, accepted December 24, 2022.

Abstract. *Graph neural network (GNN), as a powerful method for graph representation, has attracted extensive research interest. Recently, Graph Convolutional Network (GCN) and Graph Attention Network (GAT) have shown superior performance on graph node classification and are considered as the most promising frameworks. Both GAT and GCN use a weighting mechanism in the information aggregation process. However, GCN is weighted by topological structure, while the node content is still unemployed. GAT is weighted by node content, while the topological structure remains being ignored. These weighting mechanism, which does not make full use of topology and node content, is not conducive to graph representation. In this paper, we propose a "Collaborative Weighting For Graph Convolutional Network" (CWGCN) that combines the advantages of GCN and GAT. Specifically, its weighting mechanism not only considers the topology of the graph, but also the node content. With this weighting method for representation learning, CWGCN achieves encouraging performance on a number of citation datasets in node classification.*

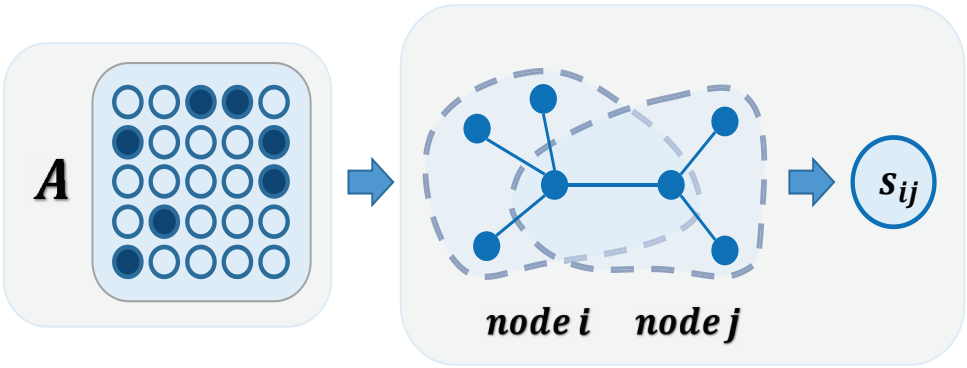
**Keywords:** Neural Network, Graph Analysis, Convolutional Network, Attention Network

1. **Introduction.** Many real-world datasets are usually represented with the graph structure, such as citation network [1, 2], social network [3, 4], and chemical molecular network [5, 6]. In social internet of vehicles, the interaction between vehicle, infrastructure and pediatrics can be represented by a graph structure [7]. In the graph structure, complex information, such as topological structure and node content, is particularly important for the learning tasks [8, 9]. Graph neural network (GNN) [10, 11], as a powerful graph representation learning tool, has excellent performance in processing graph structured data and is becoming more and more popular. GNN provides an efficient framework for processing non-Euclidean data [12], which uses information propagation iteratively to update the state of each node. Therefore, GNN can be regarded as an information transfer framework, and it can efficiently handle different types of tasks on graphs, including node embedding [13, 14] or classification [15, 16], graph classification [17, 18], etc.

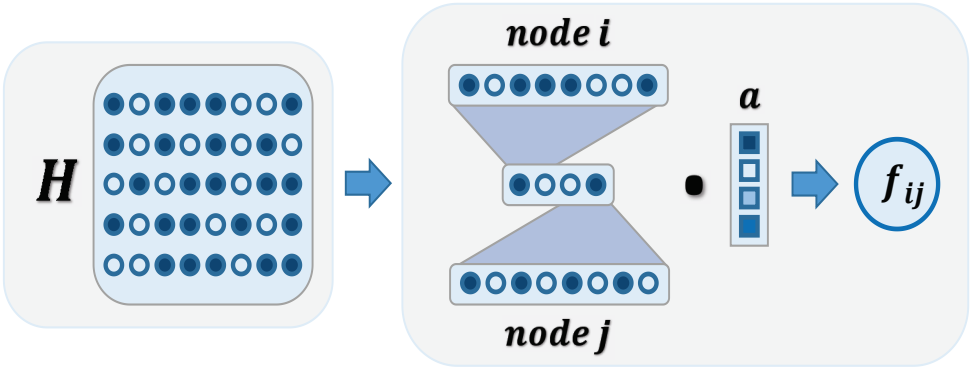
The convolutional neural networks (CNN's) architecture was originally used for image data [19, 20], so it can only handle Euclidean data and can not be adapt to graph-structured data, that is, non-Euclidean data. Wang et al. [21] used a fast Recurrent Concurrent Neural Networks to extract image features and then recognize the image. Graph Convolutional Networks are proposed by generalizing the convolution operation to graphs [22]. Graph Convolutional Networks can be categorized into two categories, namely spectral approaches and Non-spectral approaches [23, 24]. On one hand, the spectral approaches are based on the spectral representation of graphs. Through the eigendecomposition of the graph Laplacian operator [25], the convolution operation defined in the Fourier domain requires intensive matrix calculation and non-local space filtering calculation. On the other hand, non-spectral approaches perform convolution directly on the graph rather than on the graph's spectrum. Non-spectral approaches that operate on groups of spatially close neighbors can learn embeddings by aggregating features from a node's local neighborhood. One of the challenges of this approach is how to define the operation to determine the sampling field size and learn the relative weights of pairs of nodes to ensure a parameter sharing mechanism.

Recently, many excellent models in GNN have been proposed for graph convolution. For example, GCN [26] applies convolution operations to graphs. GCN efficiently handles the first-order neighborhood and weights them by topological structure for node embedding. GAT [27] introduces an attention mechanism mainly based on node content into GNN. GAT can assign different weights to the first-order neighbors of nodes so as to focus on

important parts of the data. The weighting process of GCN and GAT is shown in Figure 1. Although GCN and GAT have become one of the most effective models for graph learning, there is still room for improvement. Specifically, the weighting process of GCN ignores the importance of node content, and that of GAT ignores the importance of topological information. These weighting mechanisms that do not make full use of the node content and topology information in the graph are bound to have limitations. We conjecture that combining node and topological information should provide more valuable guidance for weighting. The reason is that topological structure represents the degree of connection between nodes, and the content of nodes can also calculate the relationship between them from the perspective of word vectors. For example, in the graph structure data shown in Figure 2, node $A$ is connected to node $B$ and node $C$, but nodes $A$ and $C$ belong to the same community, and node $B$ belongs to other community. It is worth noting that there are two important indicators for judging the importance of adjacent nodes to node $A$. On one hand, from the perspective of topology, node $C$ is more closely connected to node $A$ than node $B$. On the other hand, from the perspective of node content, node $A$ is more similar to node $C$ than node $B$ in terms of word vectors.Therefore, important adjacent nodes are often closely connected in topology and/or similar in node content. To achieve this, the weighting mechanism of the model needs to be improved to make it more appropriate.




(a)GCN, where $A$ is the adjacency matrix and $s_{ij}$ is the weight based on topological information




(b)GAT, where $H$ is the node feature matrix and $f_{ij}$ is the weight based on node content

FIGURE 1. The weighting process of GCN and GAT

In this paper, we propose a novel Collaborative Weighting For Graph Convolutional Network (CWGCN), which considers both topological structure and node content information. This means that CWGCN combines the advantages of GCN and GAT, and makes up for their shortcomings. The key idea is that the proximity between two nodes should be determined by their content and structure. Based on a local graph network, each node

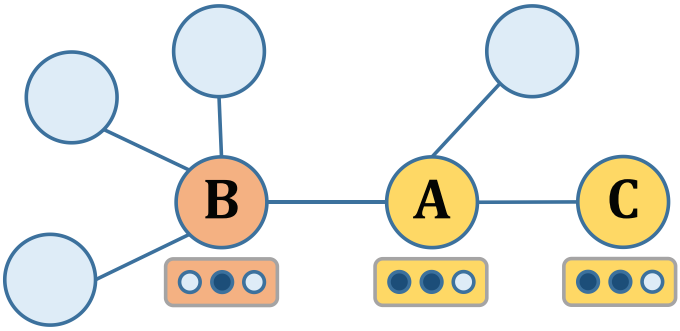



FIGURE 2. The graph structure data

in the neighborhood will be assigned a non-negative weight. In particular, a learnable cooperative trade-off mechanism is applied in the weighting process, enabling CWGCN to discover important neighbors for information aggregating. Then, the receptive field will automatically adapt to the learning task of the local graph. So, our model enables the learned node embeddings to better capture the complex topological structure and node content in graphs. Furthermore, the overall model can be optimized via backpropagation in an end-to-end manner.

The contributions of our work are summarized as follows:

- We propose a novel GNN that not only focuses on topological structure but also node content. Therefore, our work makes message delivery determined by a combination of structure and content.
- The model employs a learnable collaborative weighting mechanism, which can be adaptively optimized by learning. Benefitting from such cooperative weighting mechanism, the proposed CWGCN can better discover meaningful connections and exclude noisy connections, which will improve model performance.
- We conduct experiments on multiple citation networks to evaluate the performance of the proposed model. Compared with other state-of-the-art models, the results show that the model has superior performance. And, it has a good interpretability, which will be proved by experimental analysis.

## 2. Related Works.

**2.1. Graph neural networks.** Graph Neural Networks (GNNs) were first proposed by [28], which aims to use deep neural networks to process real-world graph-structured data. In GNNs, each node has a feature vector indicating its content, which can be updated by means of multiple information aggregation operations between neighbors. Wang et al. [29] introduced neural network into the 6G-Enabled IoT, and classify the network by using the features. DropGNN [30] proposed a new propagation model that combines gated graph neural networks with input transformations, which allows nodes and edges to have their own hidden representations. It avoided the parameter explosion problem existing in previous work. DGCN [31] separates the weight matrix from the feature propagation through a decoupling structure to improve the expressiveness and generalization performance of the model, thereby constructing a deep GNN.

**2.2. Network embedding.** The focus of GNNs is network embedding, also known as network representation learning (NRL), which aims to learn low-dimensional latent representations of nodes in the network and preserve the network structure and properties. The learned feature representations can be used for various graph-based tasks such as

classification, clustering, connection prediction, and visualization. The network embedding process can be divided into two types, one only relies on graph structure, and the other relies on graph structure and node features.

Network embedding methods depend on graph structure: Deepwalk [32] adopts a random walk method for embedding. Node2vec [33] controls the propensity of random walks. Deepwalk and Node2vec focus on the tightness of node connections, and Struc2vec [34] focuses on the similarity of regional structure. SDNE [35] uses multiple nonlinear layers for embedding.

Network embedding methods based on graph structure and node features: GraphSAGE [36] aggregates information from a fixed number of neighbor nodes to generate new features. GCNII [37] effectively alleviates the problem of over-smoothing by using Residual and Mapping, and proposes a deep neural network. AP-GCN [38] optimizes the information dissemination mechanism by independently adjusting the number of communication steps for each node. Other models include GCN and GAT.

2.3. **Weighted graph convolutional networks.** Graph Convolutional Networks (GCNs) are a powerful deep learning method for sharing information among adjacent nodes. In recent years, many different GCNs optimize information sharing by assigning different weights to different neighbors. GCN assigns different weights to nodes based on topology. GAT assigns different weights to neighbor nodes by comparing node contents with self-attention. HWGCN [39] assigns weights to high-order neighbors through node content, thereby reducing the feature loss during information aggregation. By encoding edges, SuperGAT [40] can detect noise in the graph and assign larger weights to important adjacent nodes. NeuralSparse [41] is a sparsification model that improves generalization by removing possibly task-irrelevant edges from the input graph. To sum up, the weighting process of all these algorithms above does not consider topological structure and node content information together in graph representation learning.

3. **The Proposed Algorithm.** In this section, we introduce our novel semi-supervised graph convolutional network for node classification. Our proposed model incorporates the advantages of the weighting mechanisms of GCN and GAT, while making up for the shortcomings of them. First, we obtain the weights based on the topological structure and on the node information, respectively. Then, CWGCN applies a learnable cooperative weighting mechanism to tune and obtain the optimal weight combination for the node embedding of a specific task. Figure 3 shows the entire weighting process of CWGCN. Finally, node information aggregation is performed and they are used for node prediction. To better understand the whole framework of CWGCN, we also give a brief visualization process in Figure 4. Algorithm 1 describes the pseudocode of the CWGCN algorithm.

3.1. **Notations and preliminary.** Throughout this paper, we assume a graph, denoted as $G = (V, E)$, containing $N$ nodes, $|E|$ edges , and $C$ classes to which the nodes belong. It consists of a set of nodes $V = \{v_1, v_2, ..., v_N\}$ and a set of edges $E = \{e_{ij}\}$. $E = \{e_{ij}\}$ can be represented as an adjacency matrix $A = \{a_{ij}\} \in \{0, 1\}^{N \times N}$, where $a_{ij} = 1$ if $e_{ij} \in E$, otherwise $a_{ij} = 0$. $N_{ij}$ denotes the union of node $i$ and its one-hop neighbors. $H = [h_1, h_2, ..., h_N]^T \in \mathbb{R}^{N \times D}$ represent node feature matrix, where $\{h_i^l\}_{i=1,2,...,N} \in \mathbb{R}^D$ are the features of node $v_i$ at $l$-th layer of graph convolutional network. For a given graph $G$, our goal is to learn new embeddings of nodes through a graph convolutional network, and then perform node classification. We expect nodes of same category to have the similar embeddings.

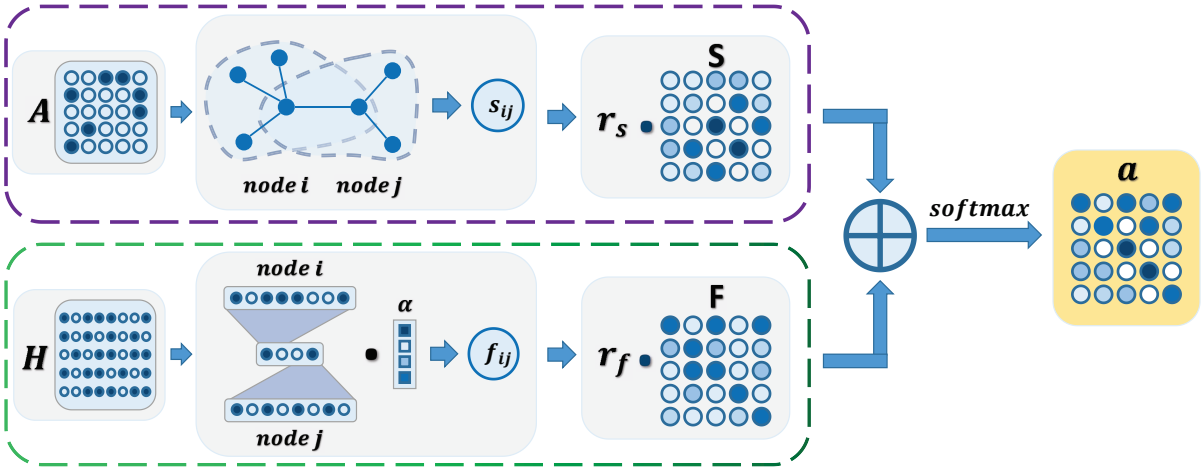



FIGURE 3. The entire weighting process of CWGCN

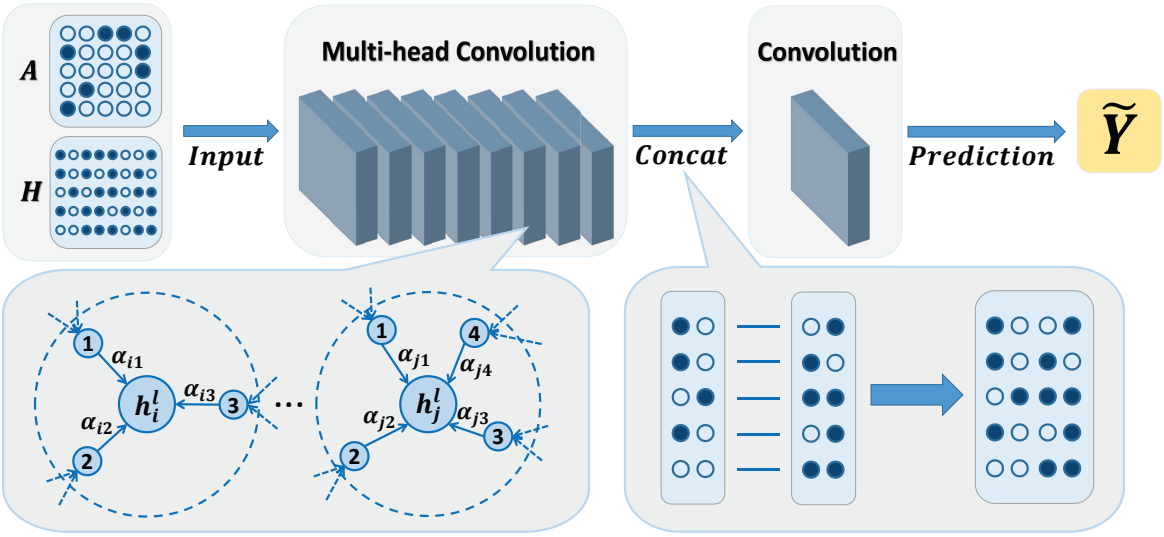



FIGURE 4. The architecture of CWGCN

3.2. **Collaborative weighting.** We should note that the neighbors of each node have different importance for its embedding. Therefore, before aggregating node information, we need to judge the importance of each neighbor node. Here, we introduce collaborative weighting, which learns the importance of different neighbors and weights them. As shown in Figure 3, the collaborative weighting mechanism of the new model consists of two parts: weighting based on topological structure and weighting based on node content.

Topology-based weighting exploits the structural information provided by the adjacency matrix, and the weighting mechanism is derived from GCN. The weighting $s_{ij}$ between nodes $i$ and $j$ can be expressed as:

$$s_{ij} = d_i^{-\frac{1}{2}} \cdot a_{ij} \cdot d_j^{-\frac{1}{2}}, \tag{1}$$

where $a_{ij}$ indicates whether node $i$ and node $j$ are neighbors to each other. $d_i$ represents the degree of node $i$. It is worth noting that we take into account the respective degrees of adjacent nodes in the calculation.

**Algorithm 1** The overall process of CWGCN.

**Input:** The heterogeneous graph $G = (V, E)$, node feature $\{h_i, \forall i \in V\}$, number of attention head $K$.

**Output:** Final embedding $H$, structural weight coefficient $s_{ij}$, content weight coefficient $f_{ij}$, weight parameters $g_s$ and $g_f$.

**for** $k = 1...K$ **do**
  **for** $i \in V$ **do**
    **for** $j \in N_i$ **do**
      Learn the structural weight coefficient $s_{ij}$ using Eq.2.
      Learn the structural weight coefficient $f_{ij}$ using Eq.4.
      Learn the structural weight coefficient $g_s$ and $g_f$ using Eq.5 and Eq.6.
      Calculate the weight coefficient $a_{ij}$ using Eq.7.
    **end**
  **end**
  Concatenate the learned embeddings $h_i$ from all attention head using Eq.9.
**end**
**for** $i \in V$ **do**
  **for** $j \in N_i$ **do**
    Learn the structural weight coefficient $s_{ij}$ using Eq.2.
    Learn the structural weight coefficient $f_{ij}$ using Eq.4.
    Learn the structural weight coefficient $g_s$ and $g_f$ using Eq.5 and Eq.6.
    Calculate the weight coefficient $a_{ij}$ using Eq.7.
  **end**
  Learn node embedding $H$ using Eq.8.
Calculate Cross-Entropy Loss using Eq.10.
Back propagation and update parameters in CWGCN.
**return** $H, s_{ij}, f_{ij}, g_s, g_f$.

Since the nodes in the same community have greater similarity in node features, weighting based on node content exploits this property. The weighting $f_{ij}$ between node $i$ and node $j$ follows the attention mechanism introduced in the original GAT, and its process can be shown as follows:

$$f_{ij} = \frac{\exp(\sigma(\alpha^T \cdot [Wh_i || Wh_j]))}{\sum_{k \in N_i} \exp(\sigma(\alpha^T \cdot [Wh_i || Wh_k]))}, \tag{2}$$

where $\{h_i\}_{i=1...N}$ is the feature vector of the node, and $h_i \in \mathbb{R}^D$. $\alpha \in \mathbb{R}^{2D}$ is the mapping vector, $||$ is the connection function, $W$ is the node feature map parameter, $\sigma(\cdot)$ denotes the activation function LeakyReLU. And $N_i$ represents the set of neighbor nodes (including itself) of node $i$. Note that the computational process can be learned. In addition, the formula uses the softmax function for normalization. $f_{ij}$ evaluates the importance of node $j$ to node $i$ in terms of node information. Please note that, $f_{ij}$ is asymmetric, i.e., the weight coefficient of node $i$ to node $j$ and the weight coefficient of node $j$ to node $i$ can be quite difference.

For the obtained $s_{ij}$ and $f_{ij}$, we introduce two learnable parameters, $g_s$ and $g_f$, to determine the relative importance of $s_{ij}$ and $f_{ij}$. We normalize them via softmax function:

$$r_s = \frac{\exp(g_s)}{\exp(g_s) + \exp(g_f)}, \tag{3}$$

$$r_f = \frac{\exp(g_f)}{\exp(g_s) + \exp(g_f)}. \tag{4}$$

Note that $r_s$ and $r_f$ can explain the contribution of different weights. Then with the obtained $r_s$ and $r_f$ we calculate the final weights by the following method:

$$a_{ij} = \frac{\exp(r_s \cdot s_{ij} + r_f \cdot f_{ij})}{\sum_{k \in N_i} \exp(r_s \cdot s_{ik} + r_f \cdot f_{ik})}, \tag{5}$$

where $r_s$ and $r_f$ are used to act as a trade-off parameter between $s_{ij}$ and $f_{ij}$. Obviously, the higher the parameter, the more important the corresponding weight. $a_{ij}$ means the final weight of node $j$ to node $i$. We only compute the $a_{ij}$ of $j \in N_i$. And we still use the softmax function for normalization. Also note that the weights $a_{ij}$ are asymmetric, which means that the connected nodes have different importance to each other. This is not only because they have different neighbors from each other, but also because they have different topological networks.

3.3. **Node information aggregation.** After obtaining the corresponding weight coefficients of the neighbors of node $i$, we perform the information aggregation operation on neighbors to form an edge-based embedding for it. The corresponding calculation process is as follows:

$$h_i^{l+1} = \sigma(\sum\nolimits_{j \in N_i} a_{ij} W h_i^l), \tag{6}$$

among them, $N_i$ represents the node's neighbor node set (including node $i$ itself). $W$ is the feature map matrix, which is learnable during task training. $\sigma(\cdot)$ is a non-linear activation function. $h_i^l$ is the input feature of the node in $l$th layer neural network. $h_i^{l+1}$ is the updating result, that is, the embedding obtained by node $i$ through learning.

The above aggregation process is a single-layer convolution operation. In order to enable the network to find more useful information, we use multi-head convolution operations, that is, we use multiple graph convolutions in a single layer of neural network. Then splicing the learned embedding, and use the obtained result as the input of the next layer:

$$h_i^{l+1} = \mathop{\Big\Vert}_{k=1}^{K} \sigma(\sum\nolimits_{j \in N_i} a_{ij} W h_i^l), \tag{7}$$

where $||$ is the concatenation operation, which is used to combine different convolution outputs, $K$ is the number of multi-head convolution.

First, in the first layer, we use Eq.(7) and set $K = 8$. After that, in the second layer, we used a single graph convolution to learn the final node embedding i.e., Eq.(6). After completing the above operations, the final node embedding will perform softmax operations.

3.4. **Optimization.** For semi-supervised nodes classification tasks, we build the Cross-Entropy as a loss function. We hope that the gap between the ground-truth label and the prediction value is minimized, the specific formula is as follows:

$$L = -\sum_{i \in T} \sum_{e \in C} y_{ie} \log \widetilde{y}_{ie}, \tag{8}$$

where $T$ represents the collection of training nodes, $C$ is the set of labels, $y_{ie}$ is the true label, and $\widetilde{y}_{ie}$ is its prediction. Guided by the ground-truth labels, we optimize our proposed model by backpropagation and learn the embedding of nodes.

4. **Experiments.** In this section, we compare and evaluate the proposed model with state-of-the-art methods, confirming that our model has the best performance over multiple datasets. The following content will include experimental setup, experimental results, and model analysis.

4.1. **Datasets.** We use three widely used citation network datasets in our experiments, i.e., Cora , Cite and Pubmed. They are standard citation network benchmark datasets [42]. In those datasets, each node represents a document. When there is a reference relationship between the two documents, an edge is established between them. Each document has its own feature vector and associated with a label. We divide the dataset into 3 parts: training set, validation set, and test set. The semi-supervised node classification test is performed. The information of those datasets are shown in Table 1.

Table 1. Summary of the datasets used in our experiments

| *Datasets* | *Nodes* | *Edges* | *Features* | *Classes* | *Training* | *Validation* | *Testing* |
|---|---|---|---|---|---|---|---|
| Cora | 2708 | 5429 | 1433 | 7 | 140 | 500 | 1000 |
| Cite | 3327 | 4732 | 3703 | 6 | 120 | 500 | 1000 |
| Pubmed | 3356 | 4278 | 500 | 3 | 60 | 500 | 1000 |

4.2. **Baselines.** In the experiments, we choose multiple state-of-the-art baselines to compare with our model to prove the effectiveness of CWGCN.

**MLP** is a two-layer artificial neural network. We set the number of hidden units to 64.

**TADW** [43] [1] combines the text features of nodes into network representation learning, based on matrix decomposition (MF).

**DGI** [44] [2] is a graph isomorphism network. In order to obtain node representations, it trains an encoder model to maximize the mutual information for capturing the global information content of the entire graph.

**SGC** [45] [3] is a simplified graph convolutional networks, which removes the nonlinear transition function between each layer and therefore constructs a linear model with repeated feature propagation.

**GCN** [26] [4] is a semi-supervised graph convolutional network. It encodes a localized approximation model for spectral graph convolutions by terms of layer-wise feature propagation, linear transformation, and pointwise nonlinear activation.

**GAT** [27] [5] is a semi-supervised neural network which considers the attention mechanism. To compute the hidden representations of a center node, it employs a self-attention mechanism for assigning weights to the neighbors.

[1] https://github.com/dedekinds/Graph-Embedding
[2] https://github.com/PetarV-/DGI
[3] https://github.com/Tiiiger/SGC
[4] https://github.com/tkipf/pygcn
[5] https://github.com/Diego999/pyGAT

4.3. **Parameter settings.** CWGCN includes a two-layer graph convolution. The first layer is a multi-head convolution, which consists of eight graph convolutions. Before entering the second layer, we concatenate the output of the first layer. The second layer consists of a single graph convolution, followed by a Softmax activation. Its outputs are used for classification, and the label corresponding to the maximum is the prediction. To prevent overfitting, regularization is used. During training, we use $\lambda = 0.0005$ for Cora and Cite datasets, and $\lambda = 0.001$ for Pubmed. Also, dropout [46] $p = 0.6$ is applied to the input of each layer. All the above measures can effectively prevent overfitting and improve the generalization performance of the model.

The training objective is to minimize the cross-entropy on the training set, and the learning rate is 0.005 for all datasets except Pubmed which is 0.01. During the training process, we use an early stop strategy for the cross-entropy loss of the validation nodes, with a patience of 100 epochs. That is, if the validation loss does not decrease within 100 epochs, we stop training.

4.4. **Results.** Our test method is semi-supervised node classification, and the node classification results are evaluated by multiple F1-score metrics. F1-score metrics include Micro-F1, Macro-F1 and Weighted-F1. Precision $P$ and recall $R$ will be used in the calculation process, and the calculation formula is as follows:

$$P = \frac{TP}{TP + FP}, \tag{9}$$

$$R = \frac{TP}{TP + FN}, \tag{10}$$

among them, $TP$ is True Positive, that is, the target sample is predicted correctly. $FP$ is False Positive, i.e. the prediction is positive, but it is wrong. $FN$ is False Negative, that is, the target sample that is wrongly predicted as to be negative.

Micro-F1 directly uses the precision $P$ and recall $R$ of the overall sample to calculate the F1-score, which means that it does not need to distinguish between categories. The calculation method:

$$Micro - F1 = \frac{2PR}{P + R} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}. \tag{11}$$

Macro-f1 calculates the arithmetic mean of F1-score for each class, this method treats all classes equally, regardless of the importance of different classes. Its calculation method:

$$Macro - F1 = \frac{\sum\limits_{e \in C} \frac{2P_e R_e}{P_e + R_e}}{C}. \tag{12}$$

Weighted-F1 considers the importance of different categories, that is, the ratio of the number of samples in each category to the total number of samples is used as the weight to calculate the weighted F1. The calculation method:

$$Weighted - F1 = \sum_{e \in C} a_e \frac{2P_e R_e}{P_e + R_e}. \tag{13}$$

The experimental results on the baseline datasets are summarized in Table 2, where the boldface values indicate the best performance. For each set of testing data, all approaches are run ten times to obtain the statistically steady performance. We can see that our approach outperforms all baselines in evaluating indicators.

Based on Table 2, we can see that GAT outperforms GCN on three datasets, but GCN has better performance on Cite Micro-F1. This shows that the weighting mechanism based

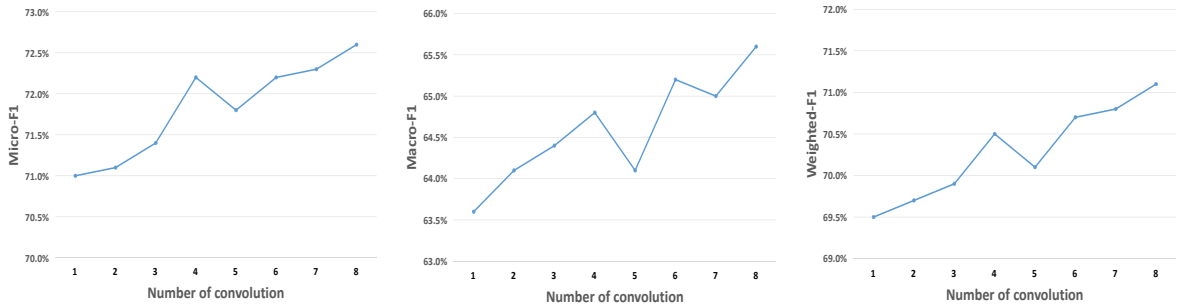
TABLE 2. Classification F1-score for different methods on the benchmark data sets

| | **Cora** | | |
|---|---|---|---|
| *Method* | $Micro-F1$ | $Macro-F1$ | $Weighted-F1$ |
| MLP | 56.0±0.1 | 50.5±0.1 | 55.8±0.1 |
| TADW | 46.2±0.7 | 56.1±0.8 | 50.1±0.7 |
| DGI | 75.3±0.4 | 69.8±0.7 | 74.2±0.5 |
| SGC | 80.0±0.1 | 78.1±0.1 | 80.2±0.1 |
| GCN | 82.1±0.6 | 79.5±0.7 | 82.1±0.6 |
| GAT | 83.8±0.2 | 82.4±0.2 | 83.8±0.1 |
| CWGCN | **84.6±0.1** | **83.1±0.1** | **84.7±0.1** |
| | **Cite** | | |
| *Method* | $Micro-F1$ | $Macro-F1$ | $Weighted-F1$ |
| MLP | 62.8±0.1 | 56.4±0.1 | 61.0±0.1 |
| TADW | 50.1±0.9 | 58.3±0.8 | 55.1±0.6 |
| DGI | 71.7±0.4 | 61.6±0.5 | 68.8±0.5 |
| SGC | 69.7±0.1 | 61.9±0.1 | 67.9±0.1 |
| GCN | 71.8±0.5 | 63.2±0.4 | 69.6±0.6 |
| GAT | 71.5±0.2 | 64.4±0.2 | 70.0±0.2 |
| CWGCN | **72.7±0.1** | **65.9±0.3** | **71.3±0.2** |
| | **Pubmed** | | |
| *Method* | $Micro-F1$ | $Macro-F1$ | $Weighted-F1$ |
| MLP | 66.5±0.1 | 64.0±0.1 | 65.5±0.1 |
| TADW | 66.1±0.7 | 68.7±0.7 | 65.8±0.5 |
| DGI | 71.1±0.2 | 67.7±0.3 | 69.7±0.2 |
| SGC | 73.9±0.1 | 71.3±0.1 | 73.3±0.1 |
| GCN | 74.2±0.7 | 71.6±0.8 | 73.4±0.8 |
| GAT | 74.8±0.1 | 73.6±0.1 | 74.5±0.1 |
| CWGCN | **76.8±0.1** | **75.9±0.4** | **76.6±0.1** |

on node content is not necessarily more effective than the weighting mechanism based on topology structure. In summary, using both node content-based and topology-based weighting mechanisms in graph convolutional networks is the key to improve performance. The weighting mechanism of CWGCN is based on the node content and topology, which makes full use of the information provided by the graph, and then can allocate weights more reasonably. This means that in the process of node information aggregation, more meaningful information can be aggregated and noise information can be eliminated. As can be seen, CWGCN consistently improves performance on all benchmark datasets.

4.5. **Parameter sensitivity.** In this section, we analyze the parameter sensitivity through the Cite data set and report the performance of CWGCN with different number of convolution heads. The results are shown in Figure 5. Note that when the number of convolution heads is set to 1, the multi-head convolution will be removed. It can be seen that with the increase of the number of convolution heads, the overall performance of CWGCN shows a trend of improvement. Therefore, to obtain the best performance on a certain dataset, we recommend searching for this parameter value first.

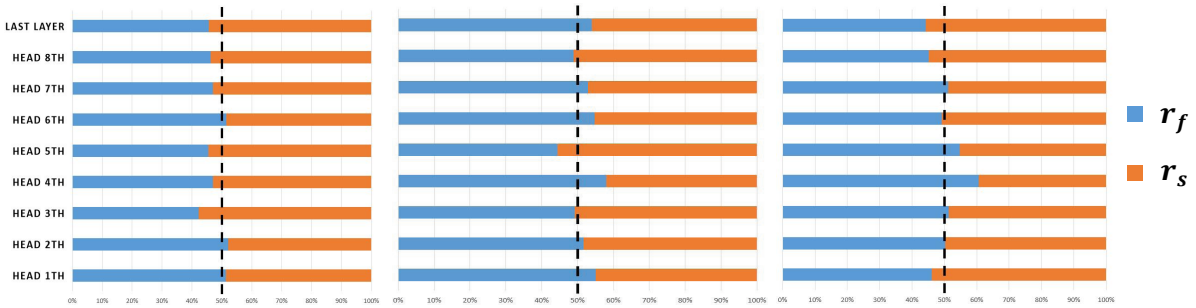
4.6. **Analysis of collaborative weighting mechanism.** Our model introduces two convolutional weights, one based on topology $s_{ij}$ and one based on node content $f_{ij}$.




(a)Micro-F1 (b)Macro-F1 (c)Weighted-F1

FIGURE 5. Parameter analysis of different number of convolutional head $K$

To explore the importance of the two weights in the model, we visualized the trade-off coefficients of the three datasets with $r_f$ and $r_f$. As shown in Figure 6, among the three we can find that both weights in the dataset play a role in CWGCN. Summarizing the results, collaborative weighting is effective.

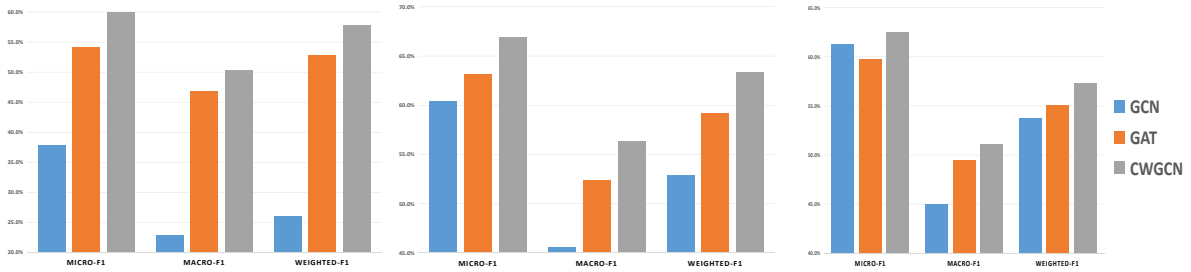



(a)Cora (b)Cite (c)Pubmed

FIGURE 6. Balancing parameters of the collaborative weighting mechanism

4.7. **Ablation analysis.** In order to further analyze the proposed method, we design a single-layer GCN, GAT, and CWGCN, while removing the influence of the multi-head convolution mechanism of GAT and CWGCN, and the randomness of dropout. In this way, other influencing factors, such as the multi-head convolution mechanism and the influence of parameter settings on results are excluded. We set the input feature dimension as the word vector dimension of the dataset samples, and the output dimension is the total number of class labels. The model is then analyzed for ablation.

4.7.1. *Results.* As shown in Figure 7, GCN, GAT, and CWGCN are all 1-layer neural networks, which not only unify parameter settings, but also remove the influence of multi-head convolution mechanism and dropout on model performance. The F1-score of all models decreased, especially on the Cora dataset. The results show that the 2-layer model is more beneficial to the representation learning for graph convolution. GCN is the worst model in the three datasets, indicating that GAT's convolution mechanism based on node information is better than GCN's topology-based convolution mechanism. The CWGCN using the collaborative attention mechanism achieves the best performance. The results show that it is beneficial to pay attention to topology information and node information when conducting convolution.

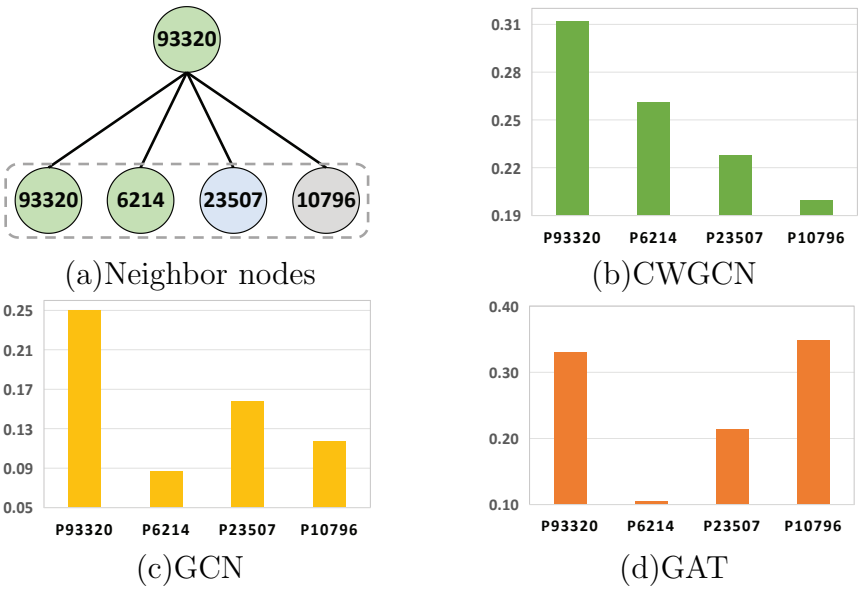



(a)Cora (b)Cite (c)Pubmed

FIGURE 7. Results of ablation experiments

4.7.2. *Analysis Of Collaborative Weighting Mechanism.* Some important neighbors that are useful for specific tasks often have greater attention value. So, the focus of improving the performance of graph convolutional networks on specific tasks is how to discover important neighbor nodes and assign appropriate weights to nodes. As mentioned earlier, take node P93320 in the Cora dataset as an example. We enumerate the neighbors of node P93320 and their weights under GCN, GAT, CWGCN. As shown in Figure 8($a$), P93320 and P6214 belong to the same label "Reinforcement Learning", P23507 belongs to the label "Neural Networks", and P10796 belongs to the label "Case Based".

As shown in Figure 8, GCN assigns a larger weight to P23507 and P10796 than P6214 based on topology, while GAT based on node content assigns the largest weight to P10796, which is obviously unreasonable. GCN and GAT predict P93320 nodes as "Neural Networks" due to unreasonable weight distribution. It is worth noting that CWGCN gives the most reasonable weight combination after neutralizing the topology and node content. CWGCN assigns higher weights to two meaningful neighbor nodes, P93320 and P6214, and finally makes correct predictions. Based on the above analysis, we can see that CWGCN has an excellent performance in discriminating the differences between neighbor nodes.




(a)Neighbor nodes (b)CWGCN

(c)GCN (d)GAT

FIGURE 8. Neighbor nodes and weights of node P93320

5. **Conclusion.** In this paper, we propose a GNN that combines the advantages of the weighting mechanism of GCN and GAT. The advantage of GCN is that its weighting

process considers topology information but ignores node content. GAT is weighted based on node content, but does not utilize topology information. It can be seen that GCN and GAT can form complementary advantages. Different from traditional models, CWGCN weighting mechanism considers not only topology structure but also node content. This means that it can measure the importance of neighbor nodes to the center node from multiple perspectives, so as to carry out more reasonable weight distribution. This weighting mechanism can greatly improve the effectiveness of node information aggregation, thereby enhancing the graph representation learning ability of CWGCN. It can be seen from the experimental results that this improvement can effectively improve the representation learning ability of graph neural networks. In future work, we will further improve the model to achieve more advanced performance.

**Acknowledgment.** This work is partially supported by Natural Science Foundation of Fujian Province (Nos.2021J011187 and 2021J011182),Scientific Research Foundation for the Introduction of Talent at Xiamen University of Technology (No.YKJ21007R). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. Yong Chen proposed the methodology, conducted experiments and wrote the manuscript; Xiao-zhu Xie and Wei Weng revised the manuscript and made constructive comments; Shandan Zhang and Tong Li participated in the discussion.

## REFERENCES

[1] H. Liu, H. Kou, C. Yan, L. Qi, "Link prediction in paper citation network to construct paper correlation graph," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1-12, 2019.

[2] I. Blank, L. Rokach, G. Shani, "Leveraging the citation graph to recommend keywords," in *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 2013, pp. 359-362.

[3] L. Tang, H. Liu, "Graph mining applications to social network analysis," *Managing and Mining Graph Data*, Springer, Boston, MA, 2010, pp. 487-513.

[4] S.-A. Myers, A. Sharma, P. Gupta, and J. Lin, "Information network or social network? The structure of the Twitter follow graph," in *Proceedings of the 23rd International Conference on World Wide Web*. WWW, 2014, pp. 493-498.

[5] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *Advances in Neural Information Processing Systems*, vol. 31, pp. 6412-6422, 2018.

[6] Y. Kim, J.-W. Kim, Z. Kim, and W.-Y. Kim, "Efficient prediction of reaction paths through molecular graph and reaction network analysis," *Chemical Science*, vol. 9, no. 4, pp. 825-835, 2018.

[7] T.-Y. Wu, X. Guo, L. Yang, Q. Meng, and C.-M. Chen, "A Lightweight Authenticated Key Agreement Protocol Using Fog Nodes in Social Internet of Vehicles," *Mobile Information Systems*, vol. 2021, 3277113, 2021.

[8] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, pp. 309-320, 2000.

[9] J. You, J. Leskovec, K. He, and S. Xie, "Graph structure of neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10881-10891.

[10] M. Gori, G. Monfardini, F. Scarselli, "A new model for learning in graph domains," in *Proceedings 2005 IEEE International Joint Conference on Neural Networks*. IEEE, 2005, vol. 2, pp. 729-734.

[11] F. Scarselli, M. Gori, A.-C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61-80, 2008.

[12] M.-M Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18-42, 2017.

[13] H. Cai, V.-W. Zheng, K.-C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616-1637, 2018.

[14] S. Cavallari, V.-W. Zheng, H. Cai, K.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 377-386.

[15] H. Xu, Y. Yang, L. Wang, and W. Liu, "Node classification in social network via a factor graph model," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, Berlin, Heidelberg, 2013, pp. 213-224.

[16] S. Xiao, S. Wang, Y. Dai, and W. Guo, "Graph neural networks in node classification: survey and evaluation," *Machine Vision and Applications*, vol. 33, no. 1, pp. 1-19, 2022.

[17] T. Kudo, E. Maeda, Y. Matsumoto, "An application of boosting to graph classification," *Advances in Neural Information Processing Systems*, vol. 17, pp. 729-736, 2004.

[18] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *7nd International Conference on Learning Representations*. ICLR, 2019, pp. 1-16.

[19] C. Farabet, C. Poulet, Y. LeCun, "An fpga-based stream processor for embedded real-time vision with convolutional networks," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2009, pp. 878-885.

[20] M. Browne, S.-S. Ghidary, "Convolutional neural networks for image processing: an application in robot vision," in *Australasian Joint Conference on Artificial Intelligence*. Springer, Berlin, Heidelberg, 2003, pp. 641-652.

[21] E.-K. Wang, X. Zhang, F. Wang, T.-Y. Wu and C.-M. Chen, "Multilayer dense attention model for image caption," *IEEE Access*, vol. 7, pp. 66358-66368, 2019.

[22] M. Edwards, X. Xie, "Graph based convolutional neural network," in *27th British Machine Vision Conference*. BMVC, 2016, pp. 1-11.

[23] D.-A. Spielman, "Spectral graph theory and its applications," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007, pp. 29-38.

[24] T. Danel, P. Spurek, J. Tabor, M. Smieja, L. Struski, A. Slowik, and L. Maziarka, "Spatial graph convolutional networks," in *International Conference on Neural Information Processing*. Springer, Cham, 2020, pp. 668-675.

[25] J.-B. Estrach, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd International Conference on Learning Representations*. ICLR, 2014, pp. 1-14.

[26] T.-N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*. ICLR, 2017, pp. 1-14.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*. ICLR, 2018, pp. 1-12

[28] A. Sperduti, A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714-735, 1997.

[29] K. Wang, P. Xu, C.-M. Chen, S. Kumari, M. Shojafar, and M. Alazab, "Neural architecture search for robust networks in 6G-enabled massive IoT domain," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5332-5339, 2020.

[30] D. Beck, G. Haffari, T. Cohn, "Graph-to-sequence learning using gated graph neural networks," *Association for Computational Linguistics*, vol. 1, pp. 273-283, 2018.

[31] W. Cong, M. Ramezani, M. Mahdavi, "On provable benefits of depth in training graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9936-9949, 2021.

[32] B. Perozzi, R. Al-Rfou, S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 701-710.

[33] A. Grover, J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855-864.

[34] L.-F.-R. Ribeiro, P.-H.-H. Saverese, D.-R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 385-394.

[35] D. Wang, P. Cui, W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1225-1234.

[36] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation Learning on Graphs with Jumping Knowledge Networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453-5462.
[37] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1725-1735.
[38] I. Spinelli, S. Scardapane, A. Uncini, "Adaptive propagation graph convolutional network," *Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4755-4760, 2020.
[39] S Liu, L. Chen, H. Dong, Z. Wang, D. Wu, and Z. Huang, "Higher-order weighted graph convolutional networks," in *International Conference on Machine Learning*. ICML, 2019, pp. 1-15.
[40] D. Kim, A. Oh, "How to find your friendly neighborhood: Graph attention design with self-supervision," in *International Conference on Learning Representations*. ICLR, 2022, pp. 1-25.
[41] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, "Robust graph representation learning via neural sparsification," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11458-11468.
[42] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T Eliassi-Rad, "Collective Classification in Network Data," *AI Magazine*, vol. 29, no. 3, pp. 93-93, 2008.
[43] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Twenty-fourth International Joint Conference on Artificial Intelligence*. IJCAI, 2015, pp. 2111-2117.
[44] P. Veličković, W. Fedus, W.-L. Hamilton, P. Lio, Y. Bengio, and R.-D. Hjelm, "Deep Graph Infomax," in *International Conference on Machine Learning*. ICML, 2019, pp. 1-17.
[45] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6861-6871.
[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.