International Journal of Computational Intelligence Systems (2024) 17:41
https://doi.org/10.1007/s44196-024-00432-9

RESEARCH ARTICLE

DeepMCGCN: Multi-channel Deep Graph Neural Networks

Lei Meng[1,2] · Zhonglin Ye[1,2] · Yanlin Yang[1,2] · Haixing Zhao[1,2]

Received: 19 September 2023 / Accepted: 5 February 2024
© The Author(s) 2024

Abstract
Graph neural networks (GNNs) have shown powerful capabilities in modeling and representing graph structural data across various graph learning tasks as an emerging deep learning approach. However, most existing GNNs focus on single-relational graphs and fail to fully utilize the rich and diverse relational information present in real-world graph data. In addition, deeper GNNs tend to suffer from overfitting and oversmoothing issues, leading to degraded model performance. To deeply excavate the multi-relational features in graph data and strengthen the modeling and representation abilities of GNNs, this paper proposes a multi-channel deep graph convolutional neural network method called DeepMCGCN. It constructs multiple relational subgraphs and adopts multiple GCN channels to learn the characteristics of different relational subgraphs separately. Cross-channel connections are utilized to obtain interactions between different relational subgraphs, which can learn node embeddings richer and more discriminative than single-channel GNNs. Meanwhile, it alleviates overfitting issues of deep models by optimizing convolution functions and adding residual connections between and within channels. The DeepMCGCN method is evaluated on three real-world datasets, and the experimental results show that its node classification performance outperforms that of single-channel GCN and other benchmark models, which improves the modeling and representation capabilities of the model.

Keywords Deep graph neural networks · Multi-relational graphs · Multi-channel interaction · Channel-level attention mechanism

1 Introduction

In the past few years, there has been widespread attention and research on graph data as an important type of unstructured data. Graph data express complex relationships between objects through nodes and links between nodes, and its topology contains rich connectivity, relevance, and global structural information. Numerous crucial data can be effectively represented by graphs in the real world, encompassing domains like social networks [1], protein–protein networks [2], scientific collaboration networks [3], public transport networks [4]. The study of graph data has significant theoretical significance and wide application value.

The advancement of deep learning methods has simultaneously offered fresh perspectives to the field of graph data mining. Researchers have harnessed deep learning techniques for analyzing graph data, leading to the development of GNNs. GNNs learn graph topology recursively, and can extract node structural features and global graph topology information features effectively. Based on these features, various downstream tasks like node classification [5, 6], node clustering [7, 8], link prediction [9–11], knowledge graph [12, 13] and recommendation systems [14–17] can be performed well.

However, in the current research of graph neural network, most of them focus on the design of graph convolution function and graph convolution framework, without considering the multi-relationship features between graph data and ignoring the influence of different relationship features on

✉ Haixing Zhao
haixing_zhao@163.com

Lei Meng
lei_meng@foxmail.com

Zhonglin Ye
zhonglin_ye@foxmail.com

Yanlin Yang
yanlin_yang@foxmail.com

1 College of Computer, Qinghai Normal University, Xining 810001, Qinghai, China

2 The State Key Laboratory of Tibetan Intelligent Information Processing and Application, Xining 810008, Qinghai, China

node features, and most of them only consider the vertical network deepening, while ignoring the width of the network in the model design. To address the aforementioned issues, a multi-channel deep graph convolutional network (DeepMCGCN) is proposed, which forms a wide and deep graph neural network model through the interaction of information between multiple channels and the deepening of the graph convolutional layers. By introducing multi-channel graph convolution, we aim to overcome the limitations of current methods in capturing diverse information within the graph. Firstly, multi-relational features between nodes are considered to extract more hidden information from the network by constructing multi-relational feature maps. Secondly, the graph convolutional function is optimized in single-relational feature convolution channels to reduce oversmoothing and over-fitting. Then, residual connections with information interactions within and between channels are added to deepen the network. And finally, attention mechanism is employed to aggregate different relational features. DeepMCGCN addresses limitations in existing GCNs by introducing a multi-channel graph convolution mechanism, enabling more effective capture of diverse information within the graph. This helps to improve the model's ability to represent graph structures and node features, thereby enhancing performance and overcoming some of the limitations of current GCNs.

Briefly, contributions are as follows:

1. A multi-channel deep graph convolutional neural network approach is proposed, called DeepMCGCN for short, which is used for the task of semi-supervised node classification.
2. Multiple relationship features of the network are utilized to construct multiple relationship feature subgraphs in order to uncover more hidden information within the graph.
3. The graph convolution function and residual connections within and between channels are optimized and introduced to alleviate over-smoothing and overfitting issues. Finally, a channel-level attention mechanism is applied to integrate the learned node feature information from each channel.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 introduces the basic concepts of graph and some preparatory knowledge. Section 4 introduces multi-channel deep graph convolutional neural network method. Section 5 presents the experimental results. Finally, we conclude this paper in Sect. 6.

2 Related Work

GNNs [18, 19] are specialized neural network architectures tailored for learning graph data. Their primary objective is to iteratively enhance node representations by amalgamating information from neighboring nodes as well as from the preceding layer [20–23]. Zeng et al. [24] proposed the concept of Cut subgraphs and extend the encoding paradigm of random walk to the probabilities of returning to the root node of the subgraph. This strategy is employed to capture structure information as node features, consequently bolstering the expressiveness of GNNs. Sriramulu et al. [25] introduced a novel hybrid approach that combines neural networks and statistical structure learning. This approach is designed to learn the relationships between multiple variables in data and create dynamic dependency graphs. The integration of statistical structure modeling and neural networks can effectively identify causal relationships in time series. Peng et al. [26] improved feature learning by learning graph structures in the intrinsic space of raw data points, proposing a reverse graph learning method for GNNs. Liu et al. [27] proposed an evolutionary GNN (EGNN) approach that uses evolutionary algorithms to optimize GNN parameters and performs mutation by estimating different graph structures. Zou et al. [28] designed an explicit selection strategy that aggregates only similar neighbors instead of all weighted ones. A threshold is introduced to select aggregated neighbors for mean aggregation. Zhong et al. [29] constructed a hierarchy in graphs by dividing nodes into different levels of super-nodes and performed top-down message passing and aggregation via ensembles. The paper also proposed an attention mechanism to adaptively learn contribution weights at various levels, thereby enhancing the significance of multi-level messages. A time-series-based GNN is proposed by Oskarsson et al. [30] to handle irregular time steps and partially observed graphs. This model utilizes continuous time to define latent states and can make predictions at arbitrary future time points. Islam et al. [31] introduced a pattern-based graph pooling method called MPool, which combines the advantages of selective pooling and clustering pooling, allowing for the simultaneous capture of both local and global graph structures. In the selective pooling model, a node ranking model is designed based on pattern relationships among nodes, and the top-ranked nodes are selected to create the next layer of the pooling graph. Bo et al. [32] introduced an effective method that encodes all feature values and performs self-attention in the spectral domain, thus achieving learnable set-to-set spectral filters. Additionally, Specformer designed a decoder with learnable basis functions to enable non-local graph convolution. Dudzik et al. [33]

proposed integral transforms as an abstract object capable of capturing both the message-passing/aggregation stage of GNNs and the scoring/recombination stage of dynamic programming (DP). By selecting the appropriate support set and latent space, the update rules of GNNs and dynamic programming algorithms can be described as an integral transform.

To handle more complex graph data, multi-channel graph neural networks have emerged. The core idea of multi-channel graph neural networks is to utilize multiple channels (or views) to capture different aspects of information within a graph. Lin et al. [34] proposed a multi-view clustering model with deep augmentation and fusion to learn more coincident graphs, which comprises three core modules, that is, the View Enhancement, Feature Fusion and Graph Embedding Modules. The View Enhancement Module utilizes a Generative Adversarial Network (GAN) to generate enhanced affinity graphs, enabling a more comprehensive exploration of feature information. A meticulously crafted deep fusion network can substantially bolster the complementarity of each view utilizing the enhanced graph as a foundation. The iterative processing of the fused graph through feature extraction and reconstruction layers yields a uniform latent representation well-suited for clustering. Zhu et al. [35] introduced a graph convolutional network method called CNIM-GCN for node classification tasks. This method integrates topological and feature graphs and maintains shared information between them through modeling a consistency graph explicitly. Zhai et al. [36] proposed a multi-channel Attention GCN and employed it to node classification tasks. The model efficiently learns the mutual information between node features and network topology by fusing them. The difference between first-order and second-order neighbors is captured by signals with different frequencies in order to reduce the occurrence of over-smoothing. Chao et al. [37] presented a residual GAT-based emotion recognition method, which relies on a residual network to extract spatial location information from electrode channels and the correlation information among adjacent brain regions. It employs a GAT to acquire knowledge about neural functional connections among various brain regions. Li et al. [38] comprehensively learned graph representation from three aspects, which are local and global topology information and feature information, and utilized attention mechanism to fuse information, and then proposed a multi-view unsupervised graph representation learning, called MVGAE for short.

With the advancement of GNNs, researchers have found that graph convolutional networks (GCNs) often achieve optimal performance using two convolutional layers, while deeper models with more convolutional layers tend to underperform. To investigate this issue, deep graph neural networks (DeepGCNs) have emerged. Numerous experiments show that as the number of graph convolutional layers increases, graph models may suffer from overfitting and oversmoothing, which are the main causes of performance degradation. Overfitting refers to good performance on training data but decreased performance on test data, harming generalization on small datasets. Over-smoothing means node features become increasingly similar after multiple graph convolution operations, making it difficult for the model to distinguish different nodes. To tackle these concerns, researchers have proposed various optimization methods for deep GNNs. A jumping knowledge (JK) network model was proposed to leverage information from different receptive field sizes, thereby improving the learned representations [39]. Li et al. [40] argued that graph convolutions in GCNs fundamentally amount to a type of Laplacian smoothing, may causing oversmoothing. They proposed joint training and self-training to address the shortcomings of shallow GCNs, showing significant improvements with little labeled data. Li et al. [41] proposed DeeperGCN for training deep GCNs, incorporating jumping connections, specialized normalization, residual connections, and improved weight initialization to address vanishing gradients. Experiments show DeeperGCN can effectively train deep GCNs and achieve better performance and faster training on many tasks. DropEdge [42] was proposed to mitigate overfitting and over-smoothing, which randomly removed a portion of edges from the input data with all training epochs, serving as both a data augmentation technique and a means to reduce message passing. They proved DropEdge reduces over-smoothing theoretically. Chen et al. [43] proposed GCNII, which incorporates initial residual and identity mapping to alleviate the issue of oversmoothing. GCNII builds jumping connections from the input layer with initial residuals and adds identity matrices in the weight matrices. Experiments show these simple techniques effectively prevent over-smoothing and consistently improve performance when increasing depth. Gao et al. [44] introduced a novel deep GNN framework called WD-GNN. It is composed by wide and deep components, which are linear graph filters and non-linear graph neural networks, respectively. During the training process, the architecture jointly learns non-linear representations from the data. During the testing process, the wide part undergoes online retraining while the deep part remains fixed. Feng et al. [45] proposed a GNN generation process for autonomously creating high-performance deep graph neural network models. In contrast to existing manually designed and neural architecture search (NAS)-based GNN models, this approach alleviates the oversmoothing problem by introducing various flexible residual connections and initial residual connections. It also applies a two-stage search strategy within the search space to explore diversity and depth in GNN architectures.

3 Preliminaries

3.1 Main Symbols

See Table 1.

3.2 Graph and Its Representation

Suppose $G = (A, X)$ is an undirected simple graph with n nodes, where $A \in \mathbb{R}^{n\times n}$ is adjacency matrix, if $A_{ij} = 1$, it indicates that node i is adjacent to node j, and if $A_{ij} = 0$, it indicates that that node i is not adjacent to node j. $X \in \mathbb{R}^{n\times d}$ is the node feature matrix, and d is the feature dimension. The degree of node i is the number of edges associated with node i, D is the diagonal degree matrix, L is the Laplacian matrix, $L = D - A$.

The Laplacian matrix of a graph has a close relationship with its underlying structure. It provides information about the graph's topology and connectivity. The eigenvalues and eigenvectors of the Laplacian matrix contain spectral information about the graph. The Laplacian matrix serves as the basis for graph convolution operations. In graph convolutional networks (GCN), the convolution operation commonly used relies on the multiplication between the Laplacian matrix and the node feature representation. The eigenvectors of the Laplacian matrix can be regarded as the representation of graph nodes in the spectral domain. By multiplying them with the node features, we can perform smoothing operations or other transformations on the node features in the frequency domain. The eigenvalue decomposition and Laplacian matrix provide mathematical tools and theoretical foundations for modeling and analyzing the structure and topology of graphs in graph neural networks.

Table 1 Main symbols

G	Graph G
$\tilde{G}$	Graph with self-looped
A	Adjacency matrix
X	Characteristic matrix
U	The matrix consisting of the eigenvectors of the normalized Laplacian matrix
L	The normalized graph Laplacian matrix
D	The diagonal degree matrix
c	The number of Channel
l	The number of graph convolution layers
A_o	Optimized adjacency matrix
$H^{(c,l)}$	Characterization of lth layer of the cth channel

3.3 Node Classification

The main research focus of this paper is the application of graph neural networks (GNNs) in the task of node classification. GNNs are deep learning models based on graph structures, capable of classifying and predicting the labels of nodes. In the node classification task, the objective is to learn the feature representations of each node using GNNs and assign them to pre-defined categories. GNNs are able to capture complex relationships and contextual information between nodes by taking into account the relationships and local neighborhood information among nodes. GNNs employ graph convolution operations to progressively aggregate and update node representations, enabling richer information to be incorporated into the node features.

Given a set of nodes $\{v_1, v_2, \ldots, v_n\} \subset V$, the corresponding label categories of them are $\{l_1, l_2, \ldots, l_d\} \subset$ Labels, and then the node classification is to learn the node representation, thereby obtaining the following mapping: $f : V \rightarrow$ Labels.

3.4 Graph Convolutional Network

The graph convolutional neural network in the spectral domain mainly uses the graph spectral theory to design the convolution operation, which defines the convolution operation by computing the eigenvalue decomposition of the Laplacian matrix and using the Fourier transform. Given a graph signal $x \in \mathbb{R}^N$, the graph Fourier transform is performed on the graph signal in the spatial domain, the obtained graph Fourier coefficients are modulated, and then the graph signal is reconstructed in the spatial domain, that is

$$g_\theta \cdot x = U g_\theta(\Lambda) U^T x, \tag{1}$$

where $g_\theta = \mathrm{diag}(\theta)$ is the convolution kernel, $\theta \in \mathbb{R}^N$ is the parameter need to learn, U is the matrix consisting of the eigenvectors of the normalized Laplacian matrix, that is

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T, \tag{2}$$

where L is the normalized graph Laplacian matrix, A is the adjacency matrix, D is the degree matrix, and Λ is the diagonal matrix of eigenvalues.

Using k-order truncated Chebyshev polynomials $T_k(x)$ to approximate $g_\theta(\Lambda)$, we have

$$g_\theta \cdot x \approx \sum_{k=0}^{K} \theta_k T_k\left(\tilde{L}\right) x, \tag{3}$$

where $\tilde{L} = \frac{2}{\lambda_{\max}} L - I$, $\lambda_{\max}$ is the maximum eigenvalue of the normalized graph Laplacian matrix L, and $\theta \in \mathbb{R}^k$ is the parameter vector consisting of the Chebyshev polynomial coefficients.

Let the order of the Chebyshev polynomial K equal to 1 and the maximum eigenvalue approximate to 2, we have

$$g_\theta \cdot x \approx \theta_0' x + \theta_1' (L - I_N) x = \theta_0' x - \theta_1' D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x, \tag{4}$$

where θ_0', θ_1' are the adjustable parameters. Let $\theta_0' = \theta_1' = -\theta'$, we have

$$g_\theta \cdot x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x. \tag{5}$$

The renormalization operation is performed on $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, that is $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \to \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ and $\tilde{A} = A + I_N, \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, therefore, the final graph convolution operation is

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta, \tag{6}$$

where $X \in \mathbb{R}^{N \times C}$ is the node representation before performing graph convolution, $\Theta \in \mathbb{R}^{C \times F}$ is the convolution kernel parameter matrix, $Z \in \mathbb{R}^{N \times C}$ is the node representation after performing graph convolution, C is the number of channels, and F is the number of convolution kernels.

The ith and jth term of $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is not 0 only when node i and node j are connected. And for a single node, this process can be seen as aggregating information about its 1-hop neighbors, whose nodes themselves are considered 1-hop neighbors.

In order to get better results, the number of layers of neural networks becomes more and more. However, with the network model deepening, problems such as overfitting, gradient disappearance and gradient explosion will occur. The appearance of these problems not only does not improve expressiveness of the model, but also reduces the effectiveness of the model. To solve the above problems, residual connectivity is proposed and applied to graph neural networks. Graph data possess complex structures and topological relationships, typically consisting of a large number of nodes and edges. In the stacked structure of GNNs, the propagation of information and gradients can be constrained by local propagation, leading to information loss or gradient decay. The introduction of residual connections in GNNs helps facilitate the better propagation of information and gradients within the network, thereby improving the performance of graph neural networks.

The general form of ResGCN is

$$H_{\mathrm{ReS}}^{l+1} = H^{l+1} + H^l = \mathcal{F}(H^l, W^l) + H^l, \tag{7}$$

where $\mathcal{F}$ is the graph convolution operation, H^l is the hidden state matrix of the lth layer, and W^l is the parameter matrix of the lth layer.

4 4. Proposed Methods

4.1 Definition

To more comprehensively capture features in graph data, a multi-channel graph neural network allows for greater flexibility in learning and representing various aspects of information within the graph. This enhances the model's capability to model complex relationships by introducing multiple channels to learn different facets of the data.

Due each relationship, subgraph is corresponded to a channel in this article. Suppose c is the index of the channel, and the channel input for a graph is $G^c = (X^c, A^c)$. For a given channel, it contains multiple layers and each layer contains two operations: graph convolution and node feature learning. Suppose l is the number of layers, the lth layer of the cth channel is denoted as $G^c = \left(X^{(c,l)}, A^{(c,l)}\right)$.

4.2 Overall Framework

The framework diagram of DeepMCGCN is presented in Fig. 1.

The framework primarily comprises three components, the relational network construction module, the graph convolution module and the feature aggregation module, respectively. In the relational network construction module, the main purpose is to construct a relational network for each channel. Three kinds of relational networks are constructed, where relational network 1 is built on the basis of the citation relationship between nodes, that is, the paper is regarded as a node, and if there is a citation relationship between papers, the two nodes are connected, relational network2 is constructed based on the word co-occurrence relationship of node text, that is, if the title of the paper is treated as a node and the same word appears in the title of the paper, then there is a link between the two nodes, and relationship network3 is a hybrid relationship network between relationship network1 and relationship network2, whose adjacency matrix is formed by summing the elements of the adjacency matrices from relationship network1 and relationship network2, and the non-zero elements are set to 1. By constructing different channel networks, we can capture relationship information of different levels and types, enabling a more comprehensive understanding and analysis of graph data. It helps to uncover complex associations between nodes. In the graph convolution module, a single-channel deep graph convolution module and a convolution module for inter-channel information interaction are designed. Graph convolutional modules are used to extract node features from each different relationship network. The feature aggregation

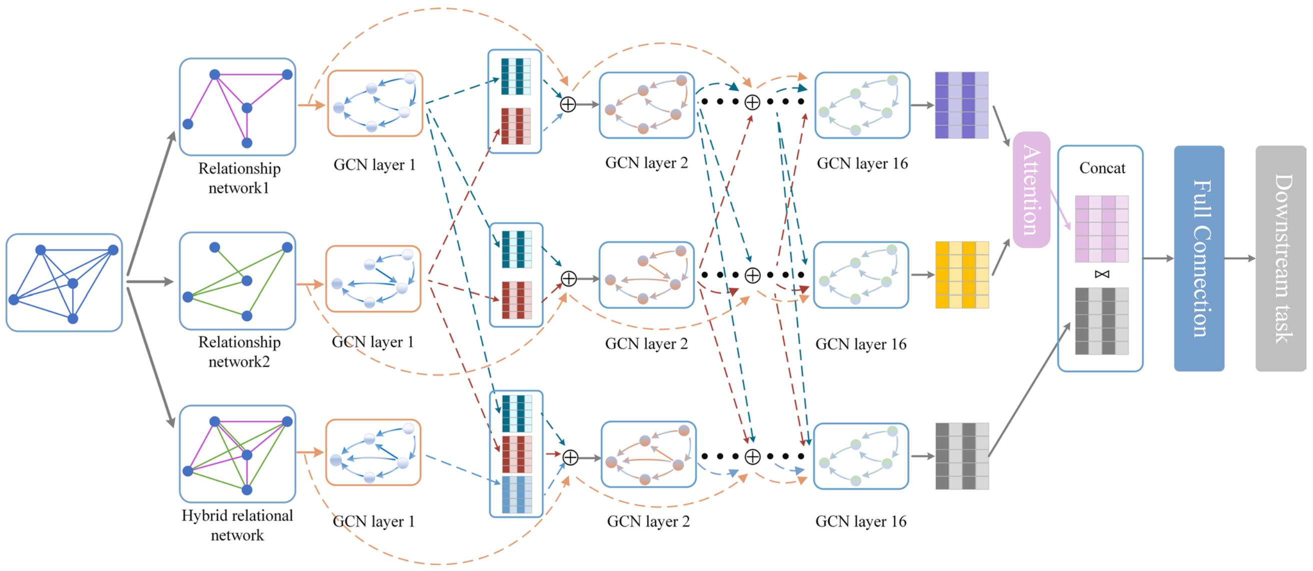


Fig. 1 Schematic description of DeepMCGCN framework

module mainly includes the attention fusion and node feature aggregation modules. Attention Fusion is used to emphasize important information through the utilization of attention mechanisms for feature fusion. Node feature aggregation, on the other hand, aims to aggregate features from different channels to obtain the final representation of node features. The introduction of multi-channel graph convolution allows DeepMCGCN to simultaneously utilize information captured by different channels, providing a more comprehensive understanding of graph structures. This contributes to improved modeling of complex relationships and node features, thereby enhancing the model's performance.

4.3 Single-Channel Feature Learning

In the single-channel node learning process, the classical GCN graph convolution process is optimized, that is, optimizing the matrix $\tilde{A}$ so that the information of its own nodes can be aggregated several times for different nodes and graph structures during the process of updating its own node information by convolution operation. Therefore, let

$$\tilde{A}_o = \tilde{A} + \theta D, \tag{8}$$

where D is the degree matrix, $D_{ii} = \sum_j A_{ij}$, A is adjacency matrix, and θ is the adjustable parameter. Replacing $\tilde{A}$ in the above equation with A_o, we have

$$Z = \tilde{D}_{\mathrm{o}}^{-\frac{1}{2}} \tilde{A}_{\mathrm{o}} \tilde{D}_{\mathrm{o}}^{-\frac{1}{2}} X\Theta, \tag{9}$$

where $\tilde{D}_{\mathrm{o}ii} = \sum_j \tilde{A}_{\mathrm{o}ii}$.

For the cth-channel and lth-layer input $G^c = \left(X^{(c,l)}, A^{(c,l)}\right)$, the feature learning process is

$$H^{(c,l+1)} = \sigma\left(\tilde{D}_{\mathrm{o}}^{(c,l)-\frac{1}{2}} \tilde{A}_{\mathrm{o}}^{(c,l)} \tilde{D}_{\mathrm{o}}^{(c,l)-\frac{1}{2}} W^{(c,l)}\right). \tag{10}$$

where $H^{(c,l+1)}$ is feature representation of the $l+1$th layer of the cth channel.

4.4 Multi-channel Interactive Learning

In the feature learning of multiple channels, the information feature interactions between channels are designed by considering different node features in different relational networks. The node features learned at layer l in the first channel are added to the node features learned at layer l in the second channel as part of the input at layer $l+1$ of the second channel, and then the residual connection is added to form a residual connection with information interaction between channels. The second channel is the same as above. Specially, the third channel is a mixed information channel, where the node features of the first two channels in the l layer are added to the node features of the third channel as the input of the $l+1$ layer of the third channel during the learning process.

For the cth-channel and lth-layer input $G^c = \left(X^{(c,l)}, A^{(c,l)}\right)$, the feature learning process between channels is as follows:

$$H^{(1,l+1)} = \mathcal{F}\left(\left[H^{(1,l)} + H^{(2,l)}\right] \cdot W^l\right) + H^{(1,l)}, \tag{11}$$

$$H^{(2,l+1)} = \mathcal{F}\left(\left[H^{(1,l)} + H^{(2,l)}\right] \cdot W^l\right) + H^{(2,l)}, \tag{12}$$

$$H^{(3,l+1)} = \mathcal{F}'\left(\left[H^{(1,l)} + H^{(2,l)} + H^{(3,l)}\right] \cdot W^l\right) + H^{(3,l)}, \tag{13}$$

where $\mathcal{F}'$ is the graph convolution operation defined in this article, and W^l is the parameter matrix can be learned.

4.5 Feature Aggregation

Considering that different features in different relational feature networks have different effects on nodes, a channel-level attention mechanism is employed to combine different relational features to account for the varying importance of distinct node features, and the aggregation function is defined by this attention mechanism. Specifically, the node feature input $X_v \in \mathbb{R}^d$ performs linear transformation, and the importance coefficients of the node features are converted into the correlation between the attention vector and the node feature matrix after linear transformation:

$$t_v = a^T \cdot \mathrm{ReLU}(W \cdot X_v + b), \tag{14}$$

where $W \in \mathbb{R}^{d_a \times d}$ is the weighting parameter, $b \in \mathbb{R}^{d_a}$ is the bias, and ReLU is a activation function. The normalization process of t_v is performed to obtain the final attention coefficients, that is

$$\mu_v = \frac{\exp(t_v)}{\sum_{v \in V} \exp(t_v)}, \tag{15}$$

where μ_v is the attention coefficient. Therefore, the different node features are aggregated as:

$$H^{(l)}_{\mathrm{ATT}} = \mu_v \cdot H^{(1,l)} + \mu_v \cdot H^{(2,l)}, \tag{16}$$

where $H^{(l)}_{\mathrm{ATT}}$ is the network representation after aggregating different relational features.

In order to get the final node representations of the three channels, $H^{(l)}_{\mathrm{ATT}}$ and the feature $H^{(3,l)}$ which is learned after the third channel are fused by the concatenation operation, that is

$$H^l_{\mathrm{final}} = \mathrm{concat}\left(H^l_{\mathrm{ATT}}, H^{(3,l)}\right). \tag{17}$$

where H^l_{final} is the network representation after concatenating $H^{(l)}_{ATT}$ and $H^{(3,l)}$, and it is the input of the final fully connected layer.

4.6 Model Training

When the model is trained, the obtained final node embedding is used as inputs for the node classification task, starting with a fully connected layer and a softmax activation function:

$$P = \mathrm{softmax}\left(H^l_{\mathrm{final}}\Theta'\right), \tag{18}$$

where node labels are assumed to have class C, $\Theta' \in \mathbb{R}^{d^m \times C}$ is the dimension reduction transformation matrix, $P \in \mathbb{R}^{N \times C}$ is the final probability matrix.

And then, the model is trained by computing the cross-entropy between the actual minimum value and the predicted value, that is

$$\mathrm{LOSS} = -\sum_{v \in \mathcal{V}_L} \sum_{c=1}^{C} Y^v[c] \cdot \ln(P^v[c]), \tag{19}$$

where $\mathcal{V}_L$ is the set of labeled nodes, Y^v is the one-hot vector indicates the ground-truth labels of nodes, P^v is the true value of the node.

5 Experiment and Result Analysis

5.1 Datasets

All experiments were carried out on three publicly available citation network datasets, that is, Citeseer (M10), DBLP (V4), and SDBLP. In each experiment, the effectiveness of the proposed method was demonstrated by comparing it with baseline methods. Detailed description of these three datasets is provided in Table 2. These datasets were chosen because they are widely used for evaluating graph classification and node classification tasks. Specifically, Citeseer involves academic paper citation relationships, DBLP includes relationships among academic papers and authors, and SDBLP contains more granular information. The selection aims to provide diversity and challenges for a more comprehensive evaluation of DeepMCGCN's performance.

Each dataset is partitioned into semantic relationship networks and structural relationship networks based on the type of relationship in Table 1. In the semantic network, the relationships between nodes are constructed based on word co-occurrence, meaning that if the same word appears in the

Table 2 Description of dataset attributes

Dataset	CiteSeer		DBLP		SDBLP	
	Structure	Semantics	Structure	Semantics	Structure	Semantics
Node	4610	4610	17,725	17,725	3119	3119
Edge	5923	819,346	105,781	1,253,600	39,516	439,182

titles of two papers, there is a connection between the corresponding nodes. In the structural network, the connections between nodes are determined based on the citation relationships between different papers. To validate the feasibility in a dense network, a high average degree network called SDBLP is constructed based on DBLP (V4). In the SDBLP network, nodes with less than 3 citations are deleted, meaning nodes with a degree less than 3 are removed.

5.2 Baseline Methods

We categorize the baseline methods into two groups. The first group consists of traditional network representation learning methods like DeepWalk [46], LINE [47], Node2Vec [48] and GraRep [49]. These methods described above are traditional network representation learning methods that have been widely adopted and studied in the literature. These methods have shown competitive performance and have been benchmarked on various network analysis tasks. The second group consists of graph neural network methods, including GCN [20], GAT [21] and GCNII [43]. Comparison with these methods allows for a better understanding of the advantages of different methods in the context of specific network analysis tasks.

5.3 Experimental Settings

We evaluate our proposed algorithm against baseline methods on the node classification task. The DeepMCGCN framework is implemented in Python and TensorFlow, and experiments are conducted on an NVIDIA 3060 platform (GPU:12G, CPU:16G). The GNN parameters are initialized using the Xavier initialization method and optimized with the Adam optimizer. Xavier initialization is a technique for initializing the weights of neural network layers. It aims to keep the variance of the activations and gradients relatively constant across layers during forward and backward propagation. The idea behind this initialization is to prevent the gradients from vanishing or exploding, which can hinder the training process. Adam (Adaptive Moment Estimation) is an optimization algorithm that combines the ideas of both momentum optimization and adaptive learning rates. It is one of the most widely used optimization algorithms for training deep neural networks. The logistic regression classifier is trained for 200 iterations to ensure convergence. To validate the generalization ability, we set the training set proportions to 0.2, 0.4, 0.6, and 0.8, using the remaining nodes as the test set. Additionally, the hyper-parameters are set as: Hidden size is 128, Learning rate is 0.005, Dropout rate is 0.1, Degree Matrix coefficient is 0.5.

5.4 Experimental Results Analysis

The paper utilizes three real citation network datasets, namely Citeseer, DBLP, and SDBLP, as evaluation datasets. Different training set proportions, specifically 0.2, 0.4, 0.6, and 0.8, were extracted from each dataset as training sets, with the remaining data serving as the test set. The node classification accuracy of both baseline methods and the proposed algorithm on these three datasets at various training set proportions is provided in Table 3. Accuracy (ACC) is employed as the metric to assess the effectiveness of the models in classifying nodes, and the experimental results are the averages of 10 repetitions.

As shown in Table 3, DeepMCGCN achieves average ACC values of 0.89865, 0.8415, and 0.8802 across four different training proportions on the Citeseer, DBLP, and SDBLP datasets, respectively. At an 80% training proportion, compared to the best performing baseline methods on the Cora, DBLP and SDBLP datasets, DeepMCGCN exhibits improvements of 6.64%, 2.09% and 3.34%, respectively. Therefore, it can be observed that the proposed

Table 3 Accuracy of node classification on Citeseer, DBLP, and SDBLP

Dataset	Training rate (%)	Deepwalk	LINE	Node2Vec	GraRep	GCN	GAT	GCNII	Deep MCGCN
Citeseer	20	0.5930	0.4706	0.6561	0.5309	0.7738	0.8148	0.8109	0.8867
	40	0.6148	0.4957	0.6707	0.5975	0.7875	0.8299	0.8213	0.8924
	60	0.6230	0.5102	0.6715	0.6105	0.8026	0.8429	0.8315	0.9019
	80	0.6233	0.5307	0.6807	0.6209	0.8069	0.8472	0.8457	0.9136
DBLP	20	0.6434	0.6653	0.7398	0.6590	0.8010	0.8237	0.8152	0.8366
	40	0.6598	0.6787	0.7561	0.6792	0.8018	0.8240	0.8118	0.8412
	60	0.6618	0.6830	0.7585	0.6888	0.8051	0.8229	0.8138	0.8438
	80	0.6703	0.6889	0.7573	0.6956	0.8029	0.8235	0.8087	0.8444
SDBLP	20	0.8065	0.7701	0.8287	0.8252	0.8146	0.8194	0.8272	0.8722
	40	0.8149	0.7828	0.8451	0.8378	0.8211	0.8202	0.8326	0.8806
	60	0.8235	0.7897	0.8401	0.8417	0.8156	0.8154	0.8243	0.8819
	80	0.8271	0.7882	0.8473	0.8527	0.8217	0.8228	0.8313	0.8861

DeepMCGCN method outperforms the seven baseline methods mentioned above across the four training proportions on all three datasets. It not only achieves favorable results on sparse datasets but also surpasses other methods on dense datasets, thus demonstrating the feasibility of the proposed DeepMCGCN. Comparisons with other graph neural networks demonstrate that DeepMCGCN exhibits good scalability in handling large-scale graph data. Its multi-channel design enables more effective processing of complex graph structures.

To visually demonstrate the effectiveness of the DeepMCGCN method, this paper presents a graph illustrating the node classification performance of DeepMCGCN with respect to increasing network depth and training ratio. The specific results are shown in Fig. 2.

As shown in Fig. 2, the heatmaps depict the changes in ACC values for the Citeseer, DBLP, and SDBLP datasets with respect to network depth and training rate. Lighter colors indicate better model performance. From the figure, it is evident that the ACC of the DeepMCGCN method increases as the network depth and training ratio increase in all three datasets. The highest ACC is achieved when the network depth is 16 layers and the training ratio is 80%.

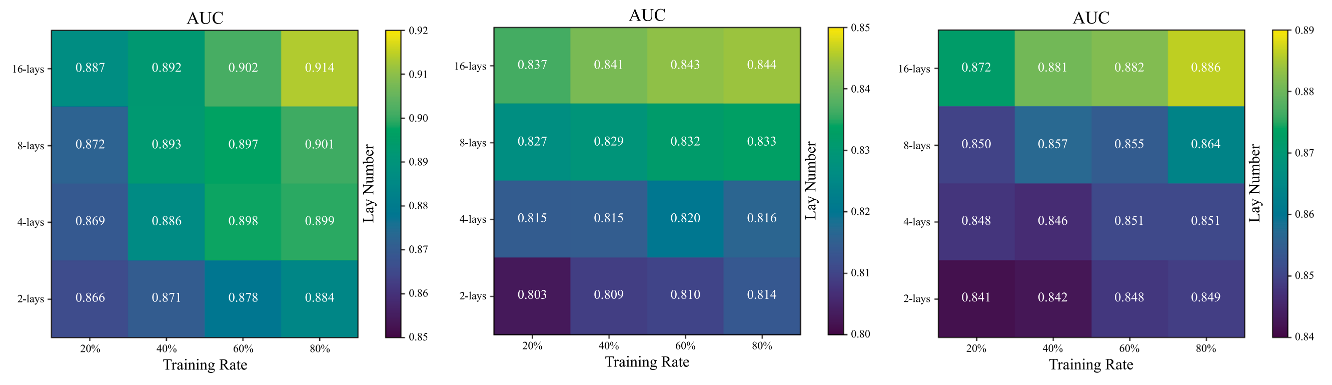
5.5 Ablation Experiment

Ablation experiments are conducted to evaluate its performance on different datasets in order to validate the effectiveness of the DeepMCGCN method. These ablation experiments were designed to systematically remove or modify components of the DeepMCGCN method, allowing for an understanding of their contributions to the overall performance. By conducting these ablation experiments on these datasets, a more comprehensive evaluation of the DeepMCGCN method's effectiveness and its adaptability in different scenarios can be obtained.

As shown in Table 4, SingleCGCN represents a single-channel graph neural network model, while MultiCGCN denotes a multi-channel graph neural network model with residual connections within channels but no inter-channel interactions. Observing Table 3, it can be noted that on the Citeseer, DBLP, and SDBLP datasets, the SingleCGCN model's performance rapidly deteriorates with an increase in the number of network layers. For instance, at 16 layers, the ACC values are 0.18467, 0.3293, and 0.26475, respectively. On the other hand, the MultiCGCN model exhibits improved performance as the number of network layers increases up to 4 layers. Subsequently, its performance begins to decline with further layer increments, but it still achieves ACC values of 0.87329, 0.79646, and 0.82807 on the Citeseer, DBLP, and SDBLP datasets, respectively, even at 16 layers. This demonstrates that the multi-channel graph neural network model with residual connections effectively alleviates overfitting and oversmoothing phenomena. The DeepMCGCN model consistently shows an upward trend in performance with an increase in the number of network layers, providing evidence of the effectiveness of the approach presented in this paper. Although DeepMCGCN achieves

Table 4 Accuracy of node classification for different network layers

Dataset	Method	Lays			
		2	4	8	16
Citeseer	SingleCGCN	0.879	0.64866	0.32299	0.18467
	MultiCGCN	0.88158	0.89068	0.88137	0.87329
	DeepMCGCN	0.88425	0.89941	0.9006	0.9136
DBLP	SingleCGCN	0.79546	0.62969	0.4393	0.3293
	MultiCGCN	0.80449	0.81086	0.80133	0.79646
	DeepMCGCN	0.8135	0.81632	0.83175	0.8438
SDBLP	SingleCGCN	0.84926	0.68476	0.42614	0.26475
	MultiCGCN	0.84548	0.84657	0.83460	0.82807
	DeepMCGCN	0.8492	0.8506	0.864	0.8819



(a) Citeseer (b) DBLP (c) SDBLP

Fig. 2 Heat map of ACC values with network depth and training rate on three datasets

better performance, it has higher complexity compared to single-channel GNNs methods and requires high-performance equipment and environment during training.

5.6 Parameter Analysis

An analysis of the experiment's parameters is conducted to assess the impact of parameter changes on the model's node classification performance during the experiments. The primary parameters in the experiments include hidden layer embedding dimensions, learning rate, dropout rate, and the coefficient of the degree matrix in graph convolution calculations. The dimensionality of hidden layer embeddings determines the model's complexity and expressive power. Increasing the number of layers in the hidden layers enhances the model's depth, allowing it to better capture the intricate features of graph data. However, the choice of the number of hidden layers requires a delicate balance. Too many layers may lead to overfitting, where the model performs well on the training set but poorly on the test set. Therefore, selecting the number of hidden layers should take into account the dataset's scale, complexity, and available computational resources. The learning rate governs the magnitude of each parameter update, playing a crucial role in model training. Optimal learning rate selection is vital; if too high, it may cause parameters to diverge during training, preventing model convergence. Conversely, if too low, the training speed slows down, requiring more iterations to achieve satisfactory performance. The Dropout rate, a widely used regularization technique, reduces overfitting by randomly dropping a portion of neuron outputs during training. The dropout rate determines the probability of retaining each neuron's output. Selecting an appropriate dropout rate helps balance model complexity and generalization. Higher dropout rates may enhance model robustness but could potentially compromise fitting capabilities. The coefficient of the degree matrix is a parameter introduced in this paper for optimizing graph convolution. The selection of these hyperparameters is an empirical process, typically involving multiple experiments and cross-validation to determine the optimal values.

The chart in Fig. 3 illustrates how these four parameters impact the experimental results on the Cora dataset.

Observing Fig. 3, it is evident that, concerning three parameters: Hidden layer embedding dimensions, Learning rate and the Coefficient of the degree matrix in graph convolution during computation—increasing these parameters has a relatively minimal impact on the number of layers. The model performance consistently reaches its peak at 16 layers for all three parameters. However, in the case of the Dropout rate, the influence of parameter variations on the number of layers is less stable, yet the highest performance is still attained at 16 layers.

From Fig. 3, it can be deduced that when the hidden layer embedding dimension is equal to 128, the Dropout rate is 0.5, the learning rate is 0.05, and the coefficient of the degree matrix during graph convolution is 0.5, the model achieves its optimal performance.

5.7 Visual Analysis

To better assess the effectiveness of the proposed method, the t-SNE algorithm is utilized in this study for dimensionality reduction and clustering comparison. The primary purpose of visualization is to observe whether the final node representations exhibit clustering phenomena. The more prominent the clustering phenomena, the better the performance of the model. In this study, the clustering of the results of the DeepMCGCN model on the Citeseer dataset with 2, 4, 8, and 16 layers is shown in Fig. 4.

Upon observing Fig. 4, it becomes evident that as the number of convolutional layers increases, the clustering effect of the DeepMCGCN model becomes more pronounced. At 16 layers, it is distinctly observable that nodes of the same color cluster closely together, and the clustering of different colors becomes more pronounced with well-defined boundaries. This visualization experiment also provides evidence that the method proposed in this paper improves model performance.

6 Conclusion

The paper proposes a multi-channel deep graph convolutional neural network model DeepMCGCN to deeply excavate the multi-relational features in graph data by deepening and widening the network architecture for more comprehensive and multi-perspective graph representation learning. Specifically, it constructs multiple relational subgraphs with each channel focusing on learning one type of relational subgraph features to capture different relationships. Cross-channel connections are introduced to learn the mutual information between different relational subgraphs. Convolution functions are optimized and residual connections are added within and between channels to alleviate overfitting and over-smoothing issues. Finally, a channel-wise attention mechanism is employed to aggregate node representation from each channel. Experiments on three actual datasets illustrate that DeepMCGCN adeptly captures intricate node relationships and bolsters the model's representation prowess. The research in this paper focuses on DeepMCGCN, a supervised graph neural network model that requires a large amount of data annotation, which hinders its practical application. Additionally, the model only handles initial graph data from a single modality. In future research, more attention will be devoted to self-supervised multi-channel

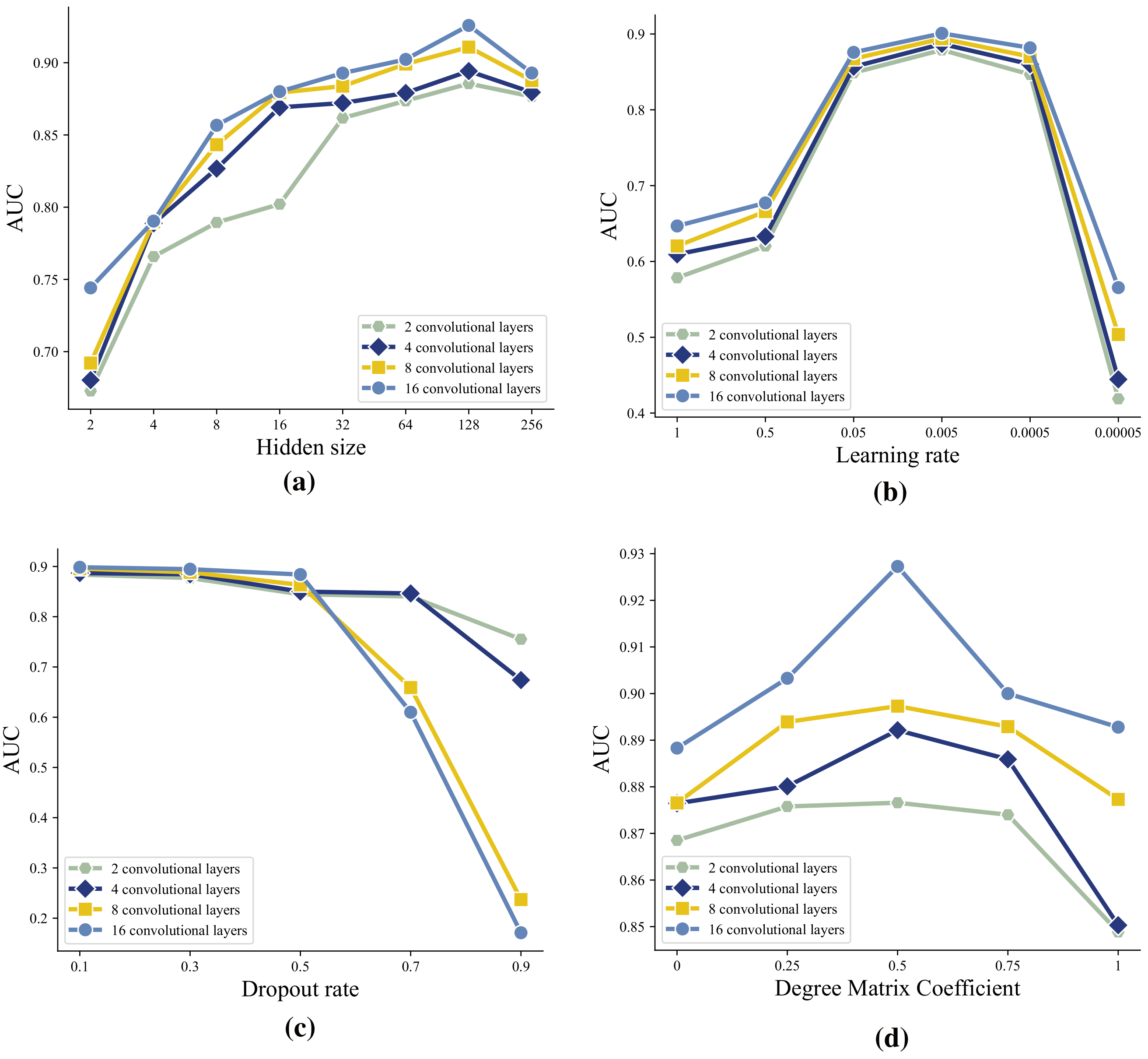


Fig. 3 Effect of parameters on experimental results on the Cora dataset

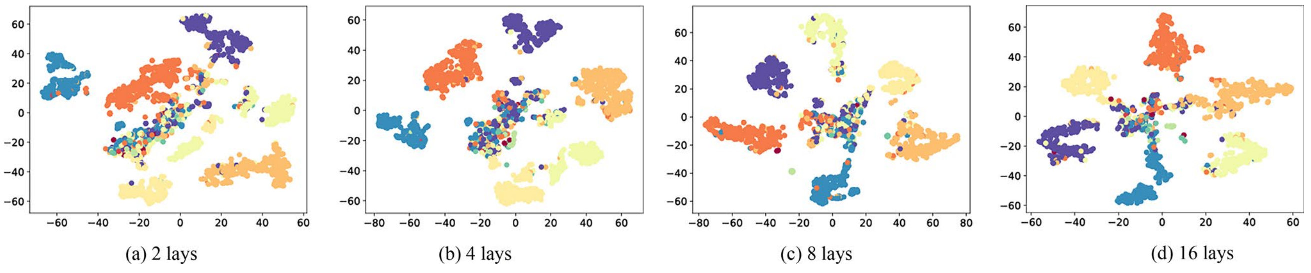


Fig. 4 Visualization of layers 2, 4, 8 and 16 on Citeseer dataset

graph neural network methods and their integration with multimodal studies to enhance the applicability of graph neural network models in real-world scenarios.

Author Contributions LM proposed an overall framework, and was a major contributor in writing the manuscript. ZY collected the dataset, proofread the manuscript. YY edited the formula, proofread the experimental data, and drew figures. HZ provided ideas for the paper and provided funding support. All authors read and approved the final manuscript.

Funding This work is partially supported by the National Key Research and Development Program of China under Grant no. 2020YFC1523300, the Construction of Innovation Platform Program of Qinghai Province of China under Grant no. 2022-ZJ-T02.

Data Availability Data will be made available from the corresponding author on reasonable request.

Declarations

Conflict of Interest The authors declare no conflict of interest.

References

1. Scott, J., Carrington, P.J.: The SAGE handbook of social network analysis. SAGE publications, London (2011)
2. Gligorijević, V., Barot, M., Bonneau, R.: deepNF: deep network fusion for protein function prediction. Bioinformatics **34**(22), 3873–3881 (2018). https://doi.org/10.1093/bioinformatics/bty440
3. Newman, M.E.J.: Scientific collaboration networks. I. Network construction and fundamental results. Phys. Rev. E **64**(1), 016131 (2001). https://doi.org/10.1103/PhysRevE.64.016131
4. Farahani, R.Z., Miandoabchi, E., Szeto, W.Y., Rashidi, H.: A review of urban transportation network design problems. Eur. J. Oper. Res.Oper. Res. **229**(2), 281–302 (2013). https://doi.org/10.1016/j.ejor.2013.01.001
5. Xiao, S., Wang, S., Dai, Y., Guo, W.: Graph neural networks in node classification: survey and evaluation. Mach. Vis. Appl. **33**, 1–19 (2022). https://doi.org/10.1007/s00138-021-01251-0
6. Luan, S., Hua, C., Xu, M., Lu, Q. C., Zhu, J. Q., Chang, X. W., Fu, J., Leskovec, J., Precup, D.: When do graph neural networks help with node classification: Investigating the homophily principle on node distinguishability (2023). arXiv preprint arXiv:2304.14274. https://doi.org/10.48550/arXiv.2304.14274
7. Wang, C., Pan, S., Yu, P.C., Hu, R.Q., Long, G.D., Zhang, C.Q.: Deep neighbor-aware embedding for node clustering in attributed graphs. Pattern Recognit. **122**, 108230 (2022). https://doi.org/10.1016/j.patcog.2021.108230
8. Khan, M.F., Bibi, M., Aadil, F., Lee, J.W.: Adaptive node clustering for underwater sensor networks. Sensors. **21**(13), 4514 (2021). https://doi.org/10.3390/s21134514
9. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: NeurIPS 2018. MIT Press, Oxford (2018)
10. Guo, Z., Shiao, W., Zhang, S., Liu, Y.Z., Chawla, N.V., Shah, N., Zhao, T.: Linkless link prediction via relational distillation. In: PMLR 2023, vol. 202, pp. 12012–12033 (2023).
11. Yang, Y.L., Ye, Z.L., Zhao, H.X., Meng, L.: A graph representation learning framework predicting potential multivariate interactions. Int. J. Comput. Intell. Syst. **16**(1), 1–16 (2023). https://doi.org/10.1007/s44196-023-00329-z
12. Chen, X., Jia, S., Xiang, Y.: A review: knowledge reasoning over knowledge graph. Expert Syst. Appl. **141**, 112948 (2020). https://doi.org/10.1016/j.eswa.2019.112948
13. Liu, S., Qin, Y.F., Xu, M., Kolmanič, S.: Knowledge graph completion with triple structure and text representation. Int. J. Comput. Intell. Syst. **16**(1), 95 (2023). https://doi.org/10.1007/s44196-023-00271-0
14. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: ACM SIGKDD'24, pp. 974–983 (2018). https://doi.org/10.1145/3219819.3219890
15. Wu, C., Liu, S., Zeng, Z., Chen, M., Alhudhaif, A., Tang, X.Y., Alenezi, F., Alnaim, N., Peng, X.C.: Knowledge graph-based multi-context-aware recommendation algorithm. Inf. Sci. **595**, 179–194 (2022). https://doi.org/10.1016/j.ins.2022.02.054
16. Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., Li, Y.: A survey of graph neural networks for recommender systems: challenges, methods, and directions. ACM Trans. Web **1**(1), 1–51 (2023). https://doi.org/10.1145/3568022
17. Zhang, Y., Li, C., Cai, J., Liu, Y., Wang, H.: BKGNN-TI: a bilinear knowledge-aware graph neural network fusing text information for recommendation. Int. J. Comput. Intell. Syst. **15**(1), 95 (2022). https://doi.org/10.1007/s44196-022-00154-w
18. Wu, L., Cui, P., Pei, J., Zhao, L.: Graph neural networks. Springer, Singapore (2022)
19. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2020). https://doi.org/10.1109/TNNLS.2020.2978386
20. Kipf, T. N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). arXiv preprint arXiv:1609.02907. https://doi.org/10.48550/arXiv.1609.02907
21. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. S.: Graph attention networks (2017). arXiv preprint arXiv:1710.10903. https://doi.org/10.48550/arXiv.1710.10903
22. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS 2017. MIT Press, Oxford (2017)
23. Wu, F., Souza, A., Zhang, T. Y, Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: PMLR 2019, vol. 97, pp. 6861–6871 (2019).
24. Zeng, D.Y., Liu, W. L., Chen, W.Y, Zhou, L., Zhang, M.L., Qu, H.: Substructure aware graph neural networks. In: AAAI'2023. AAAI Press. vol. 37(9), pp. 11129–11137 (2023). https://doi.org/10.1609/aaai.v37i9.26318
25. Sriramulu, A., Fourrier, N., Bergmeir, C.: Adaptive dependency learning graph neural networks. Inf. Sci. **625**, 700–714 (2023). https://doi.org/10.1016/j.ins.2022.12.086
26. Peng, L., Hu, R., Kong, F., Gan, J.Z., Mo, Y.J., Shi, X.S., Zhu, X.F.: Reverse graph learning for graph neural network. IEEE Trans. Neural Netw. Learn. Syst. (2022). https://doi.org/10.1109/TNNLS.2022.3161030

27. Liu, Z., Yang, D., Wang, Y.J., Lu, M.J., Li, R.R.: EGNN: graph structure learning based on evolutionary computation helps more in graph neural networks. Appl. Soft Comput.Comput. **135**, 110040 (2023). https://doi.org/10.1016/j.asoc.2023.110040
28. Zou, M.H., Gan, Z.X., Cao, R.Z., Guan, C., Leng, S.Y.: Similarity-navigated graph neural networks for node classification. Inf. Sci. **633**, 41–69 (2023). https://doi.org/10.1016/j.ins.2023.03.057
29. Zhong, Z., Li, C.T., Pang, J.: Hierarchical message-passing graph neural networks. Data. Min. Knowl. Discov. **37**(1), 381–408 (2023). https://doi.org/10.1007/s10618-022-00890-9
30. Oskarsson, J., Sidén, P., Lindsten, F.: Temporal graph neural networks for irregular data. In: PMLR 2023, pp. 4515–4531 (2023).
31. Islam, M.I.K., Khanov, M., Akbas, E.: MPool: motif-based graph pooling. In: PAKDD 2023. Springer Nature, Switzerland, pp. 105–117 (2023). https://doi.org/10.1007/978-3-031-33377-4_9
32. Bo, D.Y., Shi, C., Wang, L.L., Liao, R.J.: Specformer: Spectral graph neural networks meet transformers. In: ICLR 2023. (2023). https://openreview.net/forum?id=0pdSt3oyJa1
33. Dudzik, A.J., Veličković, P.: Graph neural networks are dynamic programmers. In: NeurIPS 2022, vol. 35, pp. 20635–20647 (2022).
34. Lin, R.J., Du, S.D., Wang, S.P., Guo, W.Z.: Multi-channel augmented graph embedding convolutional network for multi-view clustering. IEEE Trans. Netw. Sci. Eng. **10**(4), 2239–2249 (2023)
35. Zhu, X.F., Li, C.H., Guo, J.F., Dietze, S.: CNIM-GCN: consensus neighbor interaction-based multi-channel graph convolutional networks. Expert Syst. Appl. **226**, 120178 (2023). https://doi.org/10.1016/j.eswa.2023.120178
36. Zhai, R., Zhang, L.B., Wang, Y.Q., Song, Y.L.: A multi-channel attention graph convolutional neural network for node classification. J. Supercomput.Supercomput. **79**(4), 3561–3579 (2023). https://doi.org/10.1007/s11227-022-04778-9
37. Chao, H., Cao, Y.M., Liu, Y.L.: Multi-channel EEG emotion recognition using Residual Graph Attention Neural Network. Front. Neurosci.Neurosci. **17**, 1135850 (2023). https://doi.org/10.3389/fnins.2023.1135850
38. Li, J.C., Lu, G.G., Wu, Z.T., Ling, F.Q.: Multi-view representation model based on graph autoencoder. Inf. Sci. **632**, 439–453 (2023). https://doi.org/10.1016/j.ins.2023.02.092
39. Xu, K., Li, C., Tian, Y., et al.: Representation learning on graphs with jumping knowledge networks. In: PMLR 2018, vol. 80, pp. 5453–5462 (2018).
40. Li, Q., Han, Z., Wu, X. M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI'2018, pp. 32(1) (2018). https://doi.org/10.1609/aaai.v32i1.11604.
41. Li, G. H., Xiong, C. X., Thabet, A., Ghanem, B.: Deepergcn: all you need to train deeper GCNS (2020). arXiv preprint arXiv:2006.07739. https://doi.org/10.48550/arXiv.2006.07739
42. Rong, Y., Huang, W., Xu, T., Huang, J. Z.: Dropedge: towards deep graph convolutional networks on node classification (2019). arXiv preprint arXiv:1907.10903. https://doi.org/10.48550/arXiv.1907.10903
43. Chen, M., Wei, Z.W., Huang, Z.F., Ding, B.L., Li, Y.L.: Simple and deep graph convolutional networks. In: PMLR 2020, pp. 1725–1735 (2020).
44. Gao, Z., Gama, F., Ribeiro, A.: Wide and deep graph neural network with distributed online learning. IEEE Trans. Signal Process. **70**, 3862–3877 (2022). https://doi.org/10.1109/TSP.2022.3192606
45. Feng, G.S., Wang, H.Z., Wang, C.N.: Search for deep graph neural networks. Inf. Sci. (2023). https://doi.org/10.1016/j.ins.2023.119617
46. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: ACM SIGKDD'14, pp. 701–710 (2014). https://doi.org/10.1145/2623330.2623732
47. Tang J, Qu M, Wang M, et al. Line: large-scale information network embedding. In: WWW'15, pp. 1067–1077 (2015). https://doi.org/10.1145/2736277.2741093
48. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: ACM SIGKDD'16, pp. 855–864 (2016). https://doi.org/10.1145/2939672.2939754
49. Cao, S., Lu, W., Xu, Q.: Grarep: learning graph representations with global structural information. In: CIKM'15, pp. 891–900 (2015). https://doi.org/10.1145/2806416.2806512