



# Multi-view dual-channel graph convolutional networks with multi-task learning

Yuting Ling<sup>1</sup> · Yuan Li<sup>1</sup> · Xiyu Liu<sup>1</sup> · Jianhua Qu<sup>1</sup>

Received: 31 October 2022 / Accepted: 18 September 2023 / Published online: 16 October 2023  
© The Author(s) 2023

## Abstract

Network embedding has been extensively used in several practical applications and achieved great success. However, existing studies mainly focus on single task or single view and cannot obtain deeper relevant information for accomplishing tasks. In this paper, a novel approach is proposed to address the problem of insufficient information consideration in network embedding, which is termed multi-task-oriented adaptive dual-channel graph convolutional network (TAD-GCN). We firstly use kNN graph construction method to generate three views for each network dataset. Then, the proposed TAD-GCN contains dual-channel GCN which can extract the specific and shared embeddings from multiple views simultaneously, and attention mechanism is adopted to fuse them adaptively. In addition, we design similarity constraint and difference constraint to further enhance their semantic similarity and ensure that they capture the different information. Lastly, a multi-task learning module is introduced to solve multiple tasks simultaneously and optimize the model with its losses. The experimental results demonstrate that our model TAD-GCN not only completes multiple downstream tasks at the same time, but also achieves excellent performance compared with eight state-of-the-art methods.

**Keywords** Multi-view network · Network embedding · Multi-task learning · Dual-channel graph convolutional networks

## Introduction

Recently, people focus on solving these two problems: network embedding and graph convolutional network. Network embedding [1] is in an effort to learn the low-dimensional representation of network nodes. Because network embedding methods [2–7] generally are able to preserve the integrality of structure and characteristics, which of great significance of the network, they have a huge appositeness for downstream tasks. GCN, which is a semi-supervised method, combines structural and content information and performs convolution operations in the spectral domain. On this basis, a convolution operation widely applied in the spatial domain was proposed by Kipf et al. [8], which have a significant influence in a variety of downstream tasks.

Most of the known algorithms for attribute network embedding are based on two-stage or non-task-oriented

frameworks, and such learning models contain little task-related information. Some methods based on graph convolutions are used for end-to-end frameworks for semi-supervised node classification or link prediction, but only the information related to one task has been considered instead of a sufficiently general framework. Although the known methods have a great achievement, they mainly have the following shortcomings.

- (1) So far, a lot of work on attribute network embedding [9–13] has been done. However, most of them are not task-oriented models. In addition, the existing models for solving multiple tasks are also two-stage models. That is, these models need to learn network embedding first and then apply it to the downstream tasks that need to be solved. These frameworks contain almost no task-related information. Some methods based on graph convolution are used in the end-to-end framework of semi-supervised tasks [8, 14]. But they only consider content relevant to a task instead of forming a general enough framework. Therefore, most frameworks can only solve each task individually, duplicating a lot of work and ignoring the wealth of related information

✉ Jianhua Qu  
qjh@sdu.edu.cn

<sup>1</sup> School of Business, Academy of Management Science, Shandong Normal University, Jinan 250014, China

between tasks. In general, training for multiple tasks at the same time has much more advantages than training for a single task. For example, the link prediction task (one at the edge layer) and the node classification task (the other at the node layer) are both classification problems in a sense, so the model can consider these two tasks at the same time during optimization. Next, multiple tasks based on the same network are related in many cases. Multitask learning can potentially improve performance, so an effective multitask-oriented model is urgently needed in the field of network representation learning.

- (2) Typical GCN [8] and its variants [15, 16] usually follow the way of message transmission. Feature aggregation is a key step, that is, the feature information from the topological domain is aggregated through each convolution layer. The great success of GCN is partly due to the fact that GCN learning node embedding strategy can simultaneously fuse topological structure and node features, and the process is completed by the end-to-end framework. However, some recently works show that GCN actually performs Laplacian smoothing on node features, so as to gradually converge nodes of the entire network [17]. In addition, [18, 19] proved that when information is transmitted through network topology, topology can achieve the effect of low-pass filtering on node features. The above research indicates that GCN may not have the ability to optimize automatically to learn the relevant information between two important parts of network, topology structure and node features.

Aiming at limitations above, this paper proposes a multi-task-oriented adaptive dual-channel graph convolutional network model (TAD-GCN) to solve the problem of insufficient information consideration in network embedding. The core idea is that the similarity between node attributes and the similarity inferred from the topology are complementary, and the framework can adaptively fuse the embedded representations generated by dual-channel GCN to obtain deeper relevant information for accomplishing multiple tasks. And multi-task learning can share effective information among multiple tasks. It should be noted that before proceeding, multiple graph structures need to be given in advance. Here, for the network topology and attribute matrix, we choose the kNN graph construction method to generate three views for each dataset.

Specifically, in our framework, two specific convolution modules are firstly used to extract specific embedded representations from topological space and attribute space simultaneously, and a shared convolution model with shared parameters is designed to extract common embedded representations from their combinations. Accordingly, similarity

and difference constraints are designed for the two graph convolution models. Finally, the attention mechanism is used to learn the weights of embedding learned above adaptively and then fuse them and get the final representation. In this way, the model simultaneously solves multiple tasks, including node classification, link prediction, node clustering, and view reconstruction as auxiliary tasks, and these tasks participate in and supervise the process of learning, adaptively adjusting losses of each part to extract the most relevant information between nodes.

Our main contributions are as follows:

- A novel adaptive dual-channel GCN framework for attribute network representation learning is proposed, namely TAD-GCN, to complete multiple tasks simultaneously by jointly optimizing multiple loss functions.
- Specifically, the TAD-GCN model deeply embeds the multi-view information into node representations. Dual-channel GCN is applied as the basic model to effectively learn specific and shared embeddings from multi-view network. Accordingly, the consistency and difference constraints are designed for the two types of graph convolution. This overcomes the problem of graph convolution network in the fusion of node attributes and topology.
- The TAD-GCN model is based on multi-task learning, which through joint optimization of multiple loss functions to complete multiple tasks at the same time. It not only makes full use of the information that may be dependent between multiple tasks, but also effectively reduces the overall time complexity of multiple tasks.
- Experimental results on several benchmark datasets clearly show that TAD-GCN is effective and efficient over state-of-the-art baseline approaches.

The rest of this paper is organized as follows. In section "[Related work](#)", we review the related work. In section "[Preliminary](#)", we introduce the notations and preliminaries. Section "[TAD-GCN framework](#)" introduces the framework and technical details of our proposed model. Section "[Experiments](#)" presents the experimental results. Finally, we conclude our work in section "[Conclusion](#)".

## Related work

### Network representation learning

Network representation learning method (NRL) was first proposed in early 2000. After twenty years of continuous development, the types and number of methods have been increasing, so there are many different categorization systems [20, 21]. The research of network representation

learning based on deep learning model has attracted extensive attention in recent years. It mainly includes AE-based NRL, GCN-based NRL and generative adversarial network (GAN)-based NRL. Methods based on GCN can be divided into two categories: spatial method based on node features and spectral method based on node convolution. In the spatial method, convolution acts directly on the input graph data, processes the adjacent nodes in space for each node, and represents the influence exerted by its neighbors on the target node, for example the graph sampling aggregation (GraphSAGE) model presented by Hamilton et al. [22] and GAT model presented by Velickovic et al. [23]. Unlike spatial methods, spectral convolution is processed on the spectral representation of the network, usually based on eigendecomposition, where spectral analysis takes into account the locality of graph convolution. Specifically, Bruna et al. [24] proposed eigendecomposition based on graph Laplacian matrix for the first time and defined a general graph convolution framework. Later, ChebyNet et al. [8] optimized this method. In order to realize eigenvalue decomposition, Chebyshev polynomial approximation method was adopted. Finally, Kipf et al. [25] proposed graph convolution network (GCN) by using ChebyNet's first-order approximation, which achieved a more efficient filtering operation than spectral graph convolution. In terms of scalability, the space-based approach is generally superior to the spectrum-based approach.

Some improvements or variants are proposed based on these methods. For instance, GASN [6] utilizes an initial GCN or its variants to construct an autoencoder for the node features and network structure, and MAGCN [12] presents a novel model to combine multiple views of network topology with attention mechanism. However, most GCN-based methods only use a single GCN to embed the network topology and node attributes.

## Multitask learning

Multitask learning (MTL) has many successful applications in computer vision, natural language processing, bioinformatics and other fields. However, there are only a few studies that compare multitask learning with network learning, but it has become a new research trend. TANE [26] takes into account the information and multi view attributes related to downstream tasks, but does not use multiple tasks to optimize the model at the same time. MTGAE [27] designs an autoencoder with shared parameters for two tasks, node classification and link prediction, and finally trains the model by combining loss functions. JAME [28] proposes a framework for encoding the shared representation of local network structure, node features and available node labels, called the joint autoencoder framework for multi-task network embedding, and defines an adaptive loss weight layer for each task loss

function. Recently, however, there has been a growing interest in considering both multitask and multi-view learning. Zhang et al. [29] introduced a multi-task multi-view clustering framework (MTMVC), including clustering in-view, learning multi-view relation and multi-task relation. Huang et al. [30] designed a multi-view graph convolutional network MT-MVGCN, which can perform link prediction and node classification tasks simultaneously, and take view reconstruction as an auxiliary task. MT-MVGCN model can extract rich information from multi-view of the network for multi-tasks to share.

To sum up, the existing network representation learning methods cannot utilize the relevant information in multiple tasks in the meanwhile and explore correlation information and difference information between multiple tasks, so as to achieve better performance than single task.

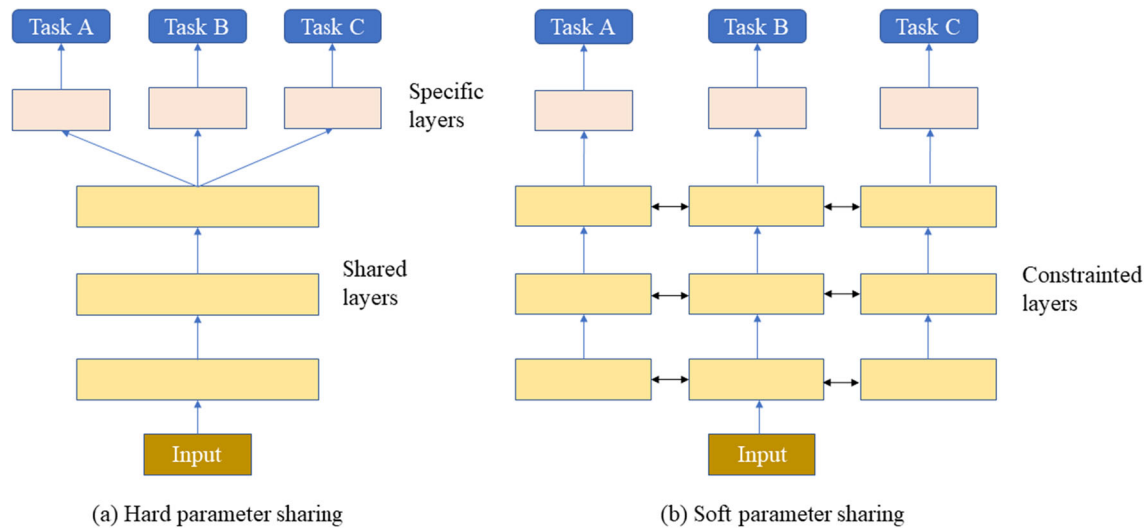
Based on the above considerations, a multitask-oriented attribute network representation learning model TAD-GCN is proposed in the paper. The core idea of this model is that the similarity between node attributes and the similarity inferred from the topology are complementary, and TAD-GCN adaptively fuses the embedding representations generated by multiple channels GCN, thereby obtaining the more in-depth information about the task. Extensive experiments on four benchmark attribute networks display that the model TAD-GCN has good performance on multiple task learning (Fig. 1).

## Preliminary

This part focuses on the general problem of network representations learning of arbitrary undirected attribute network  $G$ . Some definitions and symbols are introduced.

**Definition 1** (*Attribute Network*). Given an attribute network  $G = (V, E)$ , where  $V = \{v_1, v_2, \dots, v_N\}$  denotes the set of nodes, and  $N$  denotes the number of nodes.  $E$  denotes the set of edges, while  $e_{ij} = \langle v_i, v_j \rangle \in E$  is the edge between  $v_i$  and  $v_j$ . The topology of the  $G$  can be represented as  $A \in \mathbf{R}^{N \times N}$ , an adjacency matrix, where  $a_{ij}$  denotes the item in the  $i$ -th row and the  $j$ -th column of the matrix  $A$ . In addition, if  $e_{ij} \in E$ , then  $a_{ij}=1$ , otherwise  $a_{ij}=0$ . The node attribute matrix associated with the network is represented as  $X \in \mathbf{R}^{N \times F}$  ( $F$  is the dimension of the feature), where  $x_i \in \mathbf{R}^F$  denotes the node  $v_i$  attribute vector, which corresponds to the  $i$ -th row of  $X$ .

**Definition 2** (*Network Representation Learning*). Given a node attribute matrix  $X$  and a network topology matrix  $A$ , network representation learning (NRL) aims at learning a function  $f$  such that  $f(A, X) \rightarrow Z$ ,  $Z \in \mathbf{R}^{N \times H}$ .  $H$  is the dimension of the learned low-dimensional representation



**Fig. 1** Multi-task deep learning Hard parameter sharing is widely used in deep learning. It mainly acts on some hidden layers between all tasks to encode task information as a shared layer and at last retains a specific output layer for each task. In this way, the risk of overfitting can be reduced to the greatest extent. Soft parameter sharing is realized by

( $H \ll F$ ).  $z_i \in R^H$  denotes the low-dimensional representation of node  $v_i$ , which is the  $i$ -th row of matrix  $Z$ .

**Definition 3** (*Multitask Learning*). Given  $M$  related or partially related tasks denoted as  $\{T_m\}_{m=1}^M$ , the goal of multi-task learning (MTL) aims to utilize relevant information of  $M$  tasks and explore the association information and difference information between tasks in meanwhile, so as to achieve better performance than single task.

## TAD-GCN framework

In the section, the multi-task-oriented attribute network representation learning model TAD-GCN is constructed. The model not only considers the fusion of node attributes and network topology information, but also extracts node label-related information in these two spaces. TAD-GCN first uses two specific convolution modules to simultaneously extract specific embedding representations from topological space and attribute space and additionally designs a shared convolution module with shared parameters to extract common embedding representations from their combination. Correspondingly, two constraints of consistency and difference are designed for the two graph convolution models. Finally, the attention mechanism is used to adaptively learn the different weights of embeddings obtained earlier, so as to fuse them to obtain the final embedding representation. In this way, the model solves multiple tasks simultaneously, including node classification, clustering, link prediction and view reconstruction as auxiliary tasks, and these tasks participate

reserving a corresponding parametric model for each task, that is, each task has its own backbone network. Meanwhile, soft constraints such as sparsity, gradient similarity or L2 regularization are imposed on the distance between model parameters to ensure that the parameters of multiple models are as similar as possible [39]

in and supervise the learning process, adaptively adjusting the loss of each part to extract the most relevant information between nodes.

Figure 2 shows the overall framework of the TAD-GCN model. The TAD-GCN model details are covered here according to the GCN, the specific graph convolutional layer, the shared graph convolutional layer, the attention mechanism, and the optimization goal.

## Graph convolutional neural networks

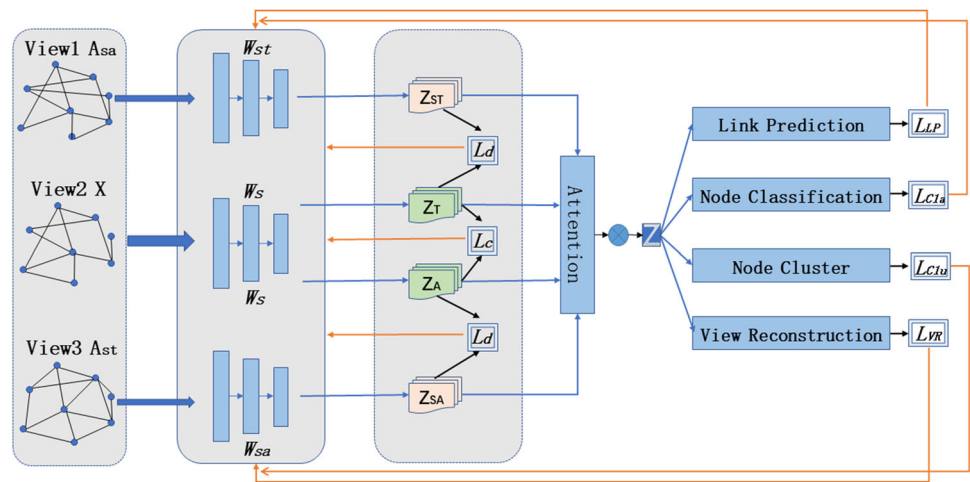
As an extension of convolutional neural network (CNN) in the field of graph structure, graph convolutional neural network (GCN) directly learns low-dimensional representations on the graph structure. Given a network  $G = (A, X)$ , a single-layer GCN is defined as follows:

$$Z^{l+1} = f(Z^l, A|W^l), \quad (1)$$

where  $Z^l$ , which is learned by the  $l$  th convolutional layer, is the latent representation matrix ( $Z^0 = X$ ) and  $W^l$  is the trainable weight matrix of the  $l$  th layer. Based on a single-layer GCN, networks of arbitrary depth can be constructed. However, after multiplying  $W^l$  by  $A$  for each layer, the vector size is unstable because  $A$  is not normalized. To solve this problem, GCN adds a self-loop to each node to normalize  $A$ , and then, the propagation rule of GCN at  $l$  th layer is

$$f(Z^l, A|W^l) = \sigma\left(\widehat{D}^{-\frac{1}{2}}\widehat{A}\widehat{D}^{-\frac{1}{2}}Z^lW^l\right), \quad (2)$$

**Fig. 2** Overall framework of TAD-GCN



Here,  $\widehat{A} = A + I$ , where  $I$  is the identity matrix,  $\widehat{D}$  is the diagonal matrix of  $\widehat{A}$ , and  $\sigma(\cdot)$  is the activation function, such as the *ReLU* function or the *tanh* function.

**Specific graph convolutional layers**

**The attributes-preserved graph convolutional layers**

To capture the underlying information of node attributes, a KNN graph  $G_{sa} = (A_{sa}, X)$  of the node attribute matrix  $X$  is constructed by the KNN method, where  $A_{sa}$  is the adjacency matrix of the KNN graph. There are two steps to this process:

1. Calculate the cosine similarity of  $N$  nodes to obtain the similarity matrix  $S \in R^{N \times N}$ . The feature vectors of node  $v_i$  and node  $v_j$  can be expressed as  $x_i$  and  $x_j$ , and then, the cosine similarity is calculated as
 
$$S_{ij} = \frac{x_i \cdot x_j}{|x_i| |x_j|}. \tag{3}$$
2. According to the similarity matrix obtained in 1), for each node, select the top  $K$  nodes with high similarity and set the edges. The adjacency matrix  $A_{sa}$  of the KNN graph can be obtained in the end.

Finally, using the KNN graph  $G_{sa} = (A_{sa}, X)$  constructed based on the node attribute matrix as the input, the output  $Z_{sa}^l$  of the  $l$  th layer GCN can be expressed as

$$Z_{sa}^l = \sigma \left( \widehat{D}_{sa}^{-\frac{1}{2}} \widehat{A}_{sa} \widehat{D}_{sa}^{-\frac{1}{2}} Z_{sa}^{l-1} W_{sa}^l \right), \tag{4}$$

where  $W_{sf}^l$  is the weight matrix of the  $l$  th layer GCN of the specific convolution module based on node attributes, and  $\sigma$  is the activation function (here is *tanh* function).  $Z_{sa}^0 = X$ ,  $\widehat{A}_{sa} = A_{sa} + I_{sa}$ , and  $\widehat{D}_{sa}$  is the diagonal matrix of  $\widehat{A}_{sa}$ . In

addition, the final output of GCN is denoted as  $Z_{SF}$ , which is denoted as a specific node embedding learned in the node attribute space of the network.

**The topology-preserved graph convolutional layers**

Similar to section "The attributes-preserved graph convolutional layers", first, to learn the information of network topological space, the KNN graph of the topology matrix  $A$  is constructed using the KNN method  $G_{st} = (A_{st}, X)$ , where the adjacency matrix of the KNN graph is denoted as  $A_{st}$ . Then,  $Z_{st}^l$  is calculated by a specific graph convolutional layer based on topology:

$$Z_{st}^l = \tanh \left( \widehat{D}_{st}^{-\frac{1}{2}} \widehat{A}_{st} \widehat{D}_{st}^{-\frac{1}{2}} Z_{st}^{l-1} W_{st}^l \right), \tag{5}$$

the output of the last layer of GCN is denoted  $Z_{ST}$ , which is denoted as the specific node embedding learned in the topological space of the network.

In the specific convolutional layer, the weight matrices of  $W_{sf}^l$  and  $W_{st}^l$  are different, which is to project the node attributes and topology into their specific semantic spaces and capture the network in both node attributes and topology. The specific information is then fused into the final low-dimensional embedding.

**Shared graph convolutional layers**

For a network, its node attribute space and topology structure space have both connections and differences. Therefore, for fully capturing the information of the network, this paper also designs a shared convolution layer, which is a convolution layer shared by node attributes and topology structure.

The shared graph convolution layer learns node low-dimensional representations  $Z_a^l$  and  $Z_t^l$  from graphs  $G_a =$



$(A_{sa}, X)$  and  $G_t = (A_{st}, X)$ , respectively:

$$\mathbf{Z}_a^l = \tanh\left(\widehat{\mathbf{D}}_a^{-\frac{1}{2}} \widehat{\mathbf{A}}_{sa} \widehat{\mathbf{D}}_a^{-\frac{1}{2}} \mathbf{Z}_a^{l-1} \mathbf{W}_s^l\right), \quad (6)$$

$$\mathbf{Z}_t^l = \tanh\left(\widehat{\mathbf{D}}_t^{-\frac{1}{2}} \widehat{\mathbf{A}}_{st} \widehat{\mathbf{D}}_t^{-\frac{1}{2}} \mathbf{Z}_t^{l-1} \mathbf{W}_s^l\right), \quad (7)$$

where  $\mathbf{Z}_a^{l-1}$  and  $\mathbf{Z}_t^{l-1}$  are the node embeddings output by the  $(l-1)$  layer GCN,  $\mathbf{Z}_a^0 = \mathbf{X}$  and  $\mathbf{Z}_t^0 = \mathbf{X}$ , and  $\mathbf{W}_s^l$  is the  $l$ th layer weight matrix of the shared graph convolutional layers. According to different input graphs, the outputs of the last layer of GCN are denoted as  $\mathbf{Z}_A$  and  $\mathbf{Z}_T$ , which represent the common node embeddings learned from the network-based node attribute graph and topology graph, respectively.

Each layer of the shared convolutional layer shares the weight matrix  $\mathbf{W}_s^l$ ; through this shared framework, the two can interact and implicitly cooperate and project into the same semantic space and make the model scalable, taking up less memory. The next subsection will discuss the aggregation function  $f$  which aggregates the node embeddings generated in the above process.

### Attention aggregation layer

The specific convolutional layer and shared convolutional layer learn to obtain two specific embeddings  $\mathbf{Z}_{SA}$  and  $\mathbf{Z}_{ST}$  and two shared embeddings  $\mathbf{Z}_A$  and  $\mathbf{Z}_T$  based on the node attribute graph and topology graph of the network, respectively. To achieve the collaboration between the two-channel graph convolutional layers, the above four node embeddings need to be fused to get the final representation. To this end, this paper uses an attention mechanism which could capture complex relationships between them instead of using aggregation functions such as concatenation, pooling or summation. The attention mechanism learns adaptive importance weights for different embeddings:

$$(\alpha_{SA}, \alpha_{ST}, \alpha_A, \alpha_T) = \text{Attention}(\mathbf{Z}_{SA}, \mathbf{Z}_{ST}, \mathbf{Z}_A, \mathbf{Z}_T), \quad (8)$$

where  $\alpha_{SA}, \alpha_{ST}, \alpha_A, \alpha_T \in \mathbf{R}^{N \times 1}$ , respectively, correspond to the attention weights of  $N$  nodes embedded in  $\mathbf{Z}_{SA}, \mathbf{Z}_{ST}, \mathbf{Z}_A, \mathbf{Z}_T$ .

For each node  $v_i$ , transform  $\mathbf{z}_{SA}^i \in \mathbf{R}^{h \times 1}$  through non-linear transformation, and then, the attention value of  $\mathbf{z}_{SA}^i$  is  $\beta_{SA}^i$ :

$$\beta_{SA}^i = \mathbf{c}^T \cdot \tanh\left(\mathbf{W} \cdot (\mathbf{z}_{SA}^i)^T + \mathbf{b}\right), \quad (9)$$

where  $\mathbf{c} \in \mathbf{R}^{h \times 1}$  denotes the learnable shared attention vector,  $\mathbf{W} \in \mathbf{R}^{h \times h}$  denotes the weight matrix, and  $\mathbf{b} \in \mathbf{R}^{h \times 1}$  denotes the partial set vector. In the same way, the attention

values  $\beta_{ST}^i, \beta_A^i$  and  $\beta_T^i$  of  $\mathbf{z}_{ST}^i, \mathbf{z}_A^i$  and  $\mathbf{z}_T^i$  can be obtained, respectively:

$$\beta_{ST}^i = \mathbf{c}^T \cdot \tanh\left(\mathbf{W} \cdot (\mathbf{z}_{ST}^i)^T + \mathbf{b}\right), \quad (10)$$

$$\beta_A^i = \mathbf{c}^T \cdot \tanh\left(\mathbf{W} \cdot (\mathbf{z}_A^i)^T + \mathbf{b}\right), \quad (11)$$

$$\beta_T^i = \mathbf{c}^T \cdot \tanh\left(\mathbf{W} \cdot (\mathbf{z}_T^i)^T + \mathbf{b}\right). \quad (12)$$

The softmax function to define the final importance weight of node  $v_i$  can be expressed as:

$$\alpha_{SA}^i = \frac{\exp(\beta_{SA}^i)}{\exp(\beta_{SA}^i) + \exp(\beta_{ST}^i) + \exp(\beta_A^i) + \exp(\beta_T^i)}. \quad (13)$$

Similar to formula (13),  $\alpha_{ST}^i, \alpha_{CA}^i$  and  $\alpha_{CT}^i$  can be calculated. The final node embedding  $\mathbf{Z}$  can be defined as:

$$\mathbf{Z} = \alpha_{SA} \cdot \mathbf{Z}_{SA} + \alpha_{ST} \cdot \mathbf{Z}_{ST} + \alpha_A \cdot \mathbf{Z}_A + \alpha_T \cdot \mathbf{Z}_T. \quad (14)$$

For different tasks such as node classification, node clustering, and so on, the final node embeddings can then be applied utilized for completing them.

### Regularization term

Through backpropagation, the learnable parameters can be automatically learned. The parameters are learned by minimizing the objective function during backpropagation,

$$\min L_{global} = \min \sum_{t_i \in T} L + \xi \cdot \Omega, \quad (15)$$

where  $T$  is the training set,  $L$  denotes the multi-task-oriented loss function, and  $\xi$  is the hyperparameter used to balance the regularization term  $\Omega$ .

The regularization term  $\Omega$  of TAD-GCN contains a total of two constraints. The similarity constraint is two embeddings  $\mathbf{Z}_A$  and  $\mathbf{Z}_T$  for the shared graph convolution output, which further enhances their semantic similarity and improves generality. Differential constraints are two embeddings  $\mathbf{Z}_{SA}$  and  $\mathbf{Z}_A$  and  $\mathbf{Z}_{ST}$  and  $\mathbf{Z}_T$  learned from the same graph for both types of graph convolutions to ensure that they capture the different information.

### Similarity constraint

Although the shared weight parameter strategy of shared graph convolution can project the outputs into the same semantic space and extract consistent information, a similarity constraint is designed to encourage the embedding  $\mathbf{Z}_A$  to be similar to  $\mathbf{Z}_T$ . Using the inner product operation, the similarity of  $N$  nodes in the network can be

obtained  $S_A$  and  $S_T$ , and then, a geometric relationship similarity score can be calculated as  $Sim(\mathbf{Z}_A, \mathbf{Z}_T)$ , where  $Sim(\cdot)$ , measured by Euclidean distance, cosine similarity, and so on, is a similarity function. Here, the simplest difference similarity function is taken for  $Sim(\cdot)$ . Then, the similarity constraint  $L_{sim}$  can be expressed as

$$L_{sim} = \|\mathbf{S}_A - \mathbf{S}_T\|_F^2. \tag{16}$$

**Difference constraint**

To preserve unique information, TAD-GCN also introduces differential loss to enhance the difference between shared embeddings and specific embeddings. There is an essential difference between consistent information and specific information, and information redundancy should be avoided. That is, shared embeddings and specific embeddings should be guaranteed to capture the information of the network from different perspectives. Therefore, the difference constraint  $L_{dif}$  is defined by the orthogonality constraint between the shared embedding and the specific embedding in the node attribute graph and the network topology graph:

$$L_{dif}^A = \|\mathbf{Z}_A \odot \mathbf{Z}_{SA}\|_F^2, \tag{17}$$

$$L_{dif}^T = \|\mathbf{Z}_T \odot \mathbf{Z}_{ST}\|_F^2, \tag{18}$$

$$L_{dif} = L_{dif}^A + L_{dif}^T. \tag{19}$$

**Multi-task learning module**

The purpose of the multi-task-oriented module is to gather task-related information and integrate it into the representation learning process and then directly output the results of the specific task. In this paper, three important tasks are studied: node classification, link prediction and node clustering. Since the loss function of the clustering task basically has no adjustment effect on the optimization of TAD-GCN model, node clustering loss part is not considered.

For the classifier of the node classification task, this paper selects multi-layer perceptron (MLP) which can predict the labels of nodes through the final node embedding  $\mathbf{Z}$ . The loss function  $L_{class}$  of the node classification task is as follows:

$$L_{class} = - \sum_{i: v_i \in V_L} \log P(y_i | \mathbf{z}_i), \tag{20}$$

where  $V_L$  is the label node and  $y_i$  denotes the label of node  $v_i$ .  $P(y_i | \mathbf{z}_i)$  is the probability of the node label output by the

softmax layer, which can be expressed as:

$$P(y_i | \mathbf{z}_i) = \frac{\exp(\mathbf{z}_i \cdot \mathbf{W}_i^c + \mathbf{b}_i^c)}{\sum_{y_j \in \mathcal{Y}} \exp(\mathbf{z}_j \cdot \mathbf{W}_j^c + \mathbf{b}_j^c)}, \tag{21}$$

where  $\mathcal{Y}$  represents the set of possible true labels and  $\mathbf{W}^c$  and  $\mathbf{b}^c$ , in turn, are the weight matrix and bias in MLP.

It is a link prediction task to predict whether there is a link between two nodes in a network. In the model, the link prediction layer is trained based on the node embedding:

$$p(\widehat{\mathbf{A}} | \mathbf{Z}) = \prod_{i=1}^n \prod_{j=1}^n p(\widehat{A}_{ij} | \mathbf{z}_i, \mathbf{z}_j), \tag{22}$$

$$p(\widehat{A}_{ij} = 1 | \mathbf{z}_i, \mathbf{z}_j) = \text{sigmoid}(\mathbf{z}_i^T, \mathbf{z}_j). \tag{23}$$

For a link prediction layer, to form the positive set  $E_+$ , we select the existing links of network as positive instances. Meanwhile, for the negative set  $E_-$ , we randomly sample an equal number of inexistent node pairs as negative instances. Then, the loss function of the link prediction task is as follows:

$$L_{linkp} = - \sum_{i, j: (v_i, v_j) \in E_+} y_{ij} \log \widehat{y}_{ij} - \sum_{k, l: (v_k, v_l) \in E_-} (1 - y_{kl}) \log (1 - \widehat{y}_{kl}), \tag{24}$$

where  $y_{ij}$  and  $y_{kl}$  indicate that the link between the node pair  $(v_i, v_j)$  and  $(v_k, v_l)$  actually exists. If there is a link between the node pair, the value is set to 1; otherwise, it is 0.  $\widehat{y}_{ij}$  is the cosine similarity of the final representations between node  $v_i$  and  $v_j$ , and the same as  $\widehat{y}_{kl}$  between node  $v_k$  and  $v_l$ . Formula (24) is a cross-entropy loss that enforces higher and lower link probabilities for positive and negative instances, respectively.

In view reconstruction, the attribute matrix  $\mathbf{X}$  and the adjacency matrix  $\mathbf{A}$  can be reconstructed, but only the adjacency matrix is reconstructed here, because such a model is also suitable for networks without attribute information. In the model, the adjacency matrix  $\widehat{\mathbf{A}}$  is reconstructed using the inner product against the node embeddings obtained:

$$\widehat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T). \tag{26}$$

Then, the network is trained with the reconstruction loss  $L_{rec}$ ,

$$L_{rec} = E_{q(\mathbf{Z} | \mathbf{X}, \mathbf{A})} [\log P(\widehat{\mathbf{A}}, \mathbf{Z})]. \tag{27}$$

## Optimization objective

As mentioned above, the model TAD-GCN includes two types of losses, namely constraints and loss functions for multiple tasks. To jointly train these losses, we get the final overall objective function of TAD-GCN:

$$L = \alpha L_{sim} + \beta L_{dif} + L_{class} + L_{linkp} + L_{rec}, \quad (27)$$

$$L = \alpha L_{sim} + \beta L_{dif} + L_{linkp} + L_{rec}, \quad (28)$$

where  $\alpha$  and  $\beta$  are hyperparameters that control the importance of similarity loss and difference loss, respectively. Formula (27) indicates that the TAD-GCN model simultaneously completes three tasks: node classification, link prediction and view reconstruction. Formula (28) indicates that the TAD-GCN model simultaneously completes three tasks: node clustering, link prediction and view reconstruction. This paper uses gradient descent and backpropagation algorithms to automatically optimize the parameters of the model. Pseudo-code of TAD-GCN model is shown in Algorithm 1.

---

### Algorithm 1: The pseudo-code of TAD-GCN Model

---

**Input:** Network  $G=(V, X, A)$ , a set of tasks  $\{1, 2, \dots, T\}$   
**Output:** The network representation  $Z$  for the network  $G$   
 1: **for** each iteration  $i$  do  
 2:   Compute  $Z_{sf}^i, Z_{st}^i, Z_{cf}^i, Z_{ct}^i$  by Equation (4) - (7)  
 3:   Compute  $\alpha_{sf}, \alpha_{st}, \alpha_{cf}, \alpha_{ct}$  by Equation (8)  
 4:   Compute  $Z$  by Equation (14);  
 5:   Compute regularization loss  $\Omega = L_{sim} + L_{dif}$  by Equation (16), (19);  
 6:   Compute each task loss  $L_{class}, L_{linkp}, L_{rec} \in \mathbb{R}^1$  by Equation (20), (24), (26);  
 7:   Obtain the global loss  $L$  by Equation (27) or (28);  
 8:   Updating model parameters by gradient descent;  
    **end for**  
 9: **return** the hidden representation  $Z$ ;

---

## Complexity analysis

The time complexity of the TAD-GCN model is has two parts: (a) learning the latent representations of nodes and (b) multi-task learning on the latent representations of nodes. For the first part, learning the latent representation of nodes, its complexity in one iteration is  $O(|N|FD + |E|D)$ , where  $|N|$  denotes the number of nodes,  $|E|$  denotes the number of edges,  $F$  is the number of node features, and  $D$  is the maximum dimension of hidden layer. Regarding the second part, since  $|N| \ll |E|$ , the time complexity of node classification task, link prediction task, and node clustering task in one iteration is  $O(|N| + |E| + |N|)$ . In fact,  $|E|, F, D$  are independent of  $|N|$ , so the complexity of TAD-GCN is  $O(|N|FD + |E|D)$ , which is linear with  $|N|$ .

## Experiments

This section is validated on four public real datasets to prove the effectiveness of the multi-task-oriented attribute network representation learning method TAD-GCN proposed in this paper. In these four datasets, we only treat the relationships between nodes as undirected edges. This section introduces the datasets, benchmark methods and evaluation metrics used in the experiments, as well as the settings of the experimental parameters.

During the experiment, the Python 3.7 version was used, based on PyTorch 1.7.0, and it was run on the NVIDIA2080TI graphics processing unit (GPU) platform with a memory of 64 GB.

## Datasets

Four standard network datasets—Cora, ACM, CiteSeer and PubMed—are utilized in our experiments (shown in Table 1). They are two types of networks: citation network and social network.

### Cora

Cora [31] is a famous published article citation network, including 2708 nodes, 5429 edges and 7 tags. In the network, nodes represent published articles, edges represent citation relationships between articles, and all articles are divided into 7 different topics. The label is the topic of the paper, and the node attribute consists of a word vector, which is used to indicate whether the corresponding word exists.

### ACM

The network comes from the ACM database [32], which includes electronic versions of journals and conference proceedings published by ACM. It contains a total of 3025 papers, and the papers were divided into 3 categories: database, wireless communication and data mining.

### CiteSeer

This dataset [33] is a citation network on computer science publications, containing of 3312 papers which can be grouped into 6 different research categories: Agents, AI, DB, IR, ML and HCI.

### PubMed

This dataset [34] is a citation relationship network for articles in the medical and biological fields. There are 19,717 articles in the biomedical field, and the entire network has 44,338 citation relationships. According to the type of disease studied of the article, it can be divided into one of three categories. Each article, i.e., each node in the network, has a feature vector with dimension 500.



**Table 1** Statistics for datasets

Dataset	Nodes	Edges	Features	Class	Training	Validation	Test	Hidden layer	Output layer
Cora	2708	5429	1433	7	20/40/60	500	1000	512	32
ACM	3025	13,128	1870	3	20/40/60	500	1000	512	256
CiteSeer	3327	4732	3703	6	20/40/60	500	1000	512	128
PubMed	19,717	44,338	500	3	20/40/60	500	1000	512	128

## Baseline

To fully analyze the experiments, we compare TAD-GCN with eight state-of-the-art methods. There are three types of benchmark algorithms. One is algorithms based only on network topology, such as DeepWalk, LINE. The other is algorithms that fuse node attribute information and network topology information, such as ChebyNet, GCN, GAT, DAEGC and MAGCN. The third is algorithms based on multi-task learning, such as JAME and MTGAE. A detailed comparison of the baseline algorithms is shown below.

### DeepWalk [35]

This algorithm is a classical network embedding method that learns embeddings using only network structure information. It uses before and after random walk and skip-gram model.

### LINE [2]

This algorithm is suitable for learning node representation in large-scale networks. It is based on the neighborhood assumption and preserves the first-order and second-order network topology information by optimizing two objective functions, respectively.

### ChebyNet [8]

ChebyNet is a graph convolution model using higher-order Chebyshev filters, which utilizes a K-order convolution algorithm to aggregate the feature information of node neighbors.

### GCN [25]

This algorithm is a semi-supervised network embedding model, which can use attribute information and structural information simultaneously. GCN learns the node representation by aggregating the information of neighbor nodes.

### GAT [23]

This algorithm combines the attention mechanism with GCN, pays attention to its neighbors, assigns different weights to different nodes in the neighborhood, and calculates the latent representation of nodes in the network. It does not require prior knowledge of network structure information.

### DAEGC [36]

DAEGC is a goal-oriented graph attention auto-encoding clustering framework, which is trained in a pre-training-fine-tuning approach. The framework mainly has graph attention auto-encoder and self-training clustering module.

### MAGCN [37]

This algorithm designs two path encoders (one is for multi-view, and the other is for consistency), which is a network clustering model.

### JAME [28]

JAME is a joint auto-encoder framework that can be used to combine local network structure, node attributes and other information to learn network embeddings for multiple tasks.

### MTGAE [27]

A multi-task graph autoencoder structure is constructed, capable of learning representations of node embeddings from local topology and task (link prediction and node classification) related node features.

## Evaluation metrics

### Node classification task

In this task, MLP is used as the classifier, and the trained model is used for performance evaluation on the test set. Three evaluation metrics are used: accuracy (ACC), macro-F1 and micro-F1.

For a category  $x$  in the dataset, use  $TP(x)$ ,  $FP(x)$ ,  $FN(x)$  to represent the number of times that the data of type  $x$  are predicted to be the data of type  $x$  (the number of true positives), and the data of type  $x$  are predicted to be non-positive. The number of times data of type  $x$  (number of false positives) and the number of times  $non - x$  type of data were predicted to be of type  $x$  (number of true negatives). For a dataset with  $C$  categories, the calculation process of micro-F1 and macro-F1 can be seen in the following formulas:

$$precision = \frac{TP(x)}{TP(x) + FP(x)}, recall = \frac{TP(x)}{TP(x) + FN(x)},$$

$$F_1(x) = \frac{2 \cdot precision \cdot recall}{precision + recall}, Macro - F_1 = \frac{\sum_{x \in C} F_1(x)}{|C|},$$

$$P = \frac{\sum_{x \in C} TP(x)}{\sum_{x \in C} (TP(x) + FP(x))}, R = \frac{\sum_{x \in C} TP(x)}{\sum_{x \in C} (TP(x) + FN(x))},$$

$$Micro - F_1 = \frac{2PR}{P + R}$$

.

Micro-F1 assigns the same weight to each sample in the dataset, and macro-F1 assigns the same weight to each of the C classes. That is, macro-F1 is the average of C categories micro-F1. Therefore, macro-F1 will score lower than micro-F1 when the sample size distribution of C categories is uneven.

### Link prediction task

In this task, widely used metrics are AUC and precision. Among them, AUC refers to the area under the ROC curve, which is the measurement index of the most commonly used of link prediction performance.

Assuming that  $M_+$  denotes the number of positive samples in the test set,  $M_-$  denotes the number of negative samples, and  $M_+ \times M_-$  denotes the number of sample pairs. The calculation method of AUC is as follows:

$$AUC = \frac{\sum I(P_+, P_-)}{M_+ \times M_-}, I(P_+, P_-) = \begin{cases} 1, & P_+ > P_- \\ 0.5, & P_+ = P_- \\ 0, & P_+ < P_- \end{cases}$$

### Node cluster task

For node clustering, we use the k-means function as a classifier and use three evaluation metrics: clustering accuracy (ACC), normalized mutual information (NMI), and Adj\_rand\_score (ARI).

Unsupervised clustering accuracy (ACC) is defined as follows:

$$ACC = \frac{\sum_{i=1}^n \delta(real_i, map(predict_i))}{n},$$

when  $real_i == map(predict_i)$ ,  $\delta = 1$ , otherwise  $\delta = 0$ . Among them,  $real_i$  represents the real label of node  $v_i$ , and  $predict_i$  represents the predicted label. The map represents the best match of the predicted label in the real label, and the Munkres algorithm is generally used.

NMI originated from confidence theory and is a similarity measure that can be used to measure the similarity between the real division result  $A$  and the division result  $B$  obtained by the algorithm. If the two division results are more similar, the NMI value is closer to 1. Let  $N$  be the confusion matrix, and the element  $N_{ij}$  represents the same number between  $A$  and  $B$ . The network has  $n$  nodes, and the NMI calculation

process is.

$$NMI(A, B) = \frac{-2 \sum_{i,j} N_{ij} \ln\left(\frac{N_{ij}n}{N_i N_j}\right)}{\sum_i N_i \ln\left(\frac{N_i}{n}\right) + \sum_j N_j \ln\left(\frac{N_j}{n}\right)}$$

### Parameter settings

In order to make a fair comparison, all the benchmark algorithms are based on the open source codebase (OpenNE) of the author of the relevant paper, and the relevant parameter settings are based on the requirements of the original paper. For DeepWalk algorithm, the random walk times  $s$  and length  $t$  are set to 10 and 80, respectively, and the window size  $w$  of skip-gram model is set to 10. The jump order of LINE algorithm is set to 2, the learning rate is set to 0.025, and the number of negative samples is set to 5.

For TAD-GCN, the final node embedding dimension  $d$  is set as 128, initial learning rate values for 0.0001–0.005, maximum iteration epoch is equal to 200, and  $k \in \{2, \dots, 10\}$ . For regularization coefficients  $\alpha$  and  $\beta$ , they can be searched from  $\{0.01, 0.001, 0.0001\}$  and  $\{1e - 10, 5e - 9, 1e - 9, 5e - 8, 1e - 8\}$ , respectively. In particular, the model simultaneously trains three GCNs with two layers, which have the same dimension of hidden layer and output.

### Experimental results and analysis

This section adopts the tasks commonly used in most algorithms: node classification, link prediction and node clustering to evaluate the node embedding quality obtained by the algorithm model. This is done to evaluate the ability of the node embedding obtained by the NRL algorithm to maintain characteristics of the original network topology and node attributes. The goal of link prediction task is to assess the ability of node embedding to reconstruct network structure, while the purpose of node classification task is to assess whether the node embedding contains enough trainable downstream task information. In addition, clustering task is able to assess the discriminant performance of node embedding learned by unsupervised algorithms to cluster structure. Based on the above tasks, this section selects the existing classical and latest network representation learning algorithm and the TAD-GCN model designed in this paper to conduct a comparative experiment on algorithm performance.

Generally, most algorithms evaluate the quality of node embedding through common network analysis tasks such as node classification and link prediction, while some unsupervised learning algorithms evaluate node embedding through clustering tasks. The algorithm model in this paper is semi-supervised, so the three evaluation tasks can be used to analyze node embedding.

As the variance of graph data is large, here follows the method of [30], repeats 10 times for each method, and reports average index.

### Node classification

In network analysis tasks, node classification, one of the most common tasks, can be generalized into a variety of practical applications. To accomplish the task, the network representation learning method is generally applied first to learn embedding of various network, and then, node representation learned before is used to train the classifier. In this paper, the labeled nodes with preset proportions are randomly selected as training set and other as test set. The support vector machine model is trained by node information (node representation and label) in the training set, and then, classifier predicts the category of the node according to the embedding. The cross-validation is repeated for ten times, and we take the mean value of the evaluation indexes as the final result. The evaluation indicators are described in Sect. [Evaluation metrics](#).

Tables 2, 3 and 4 respectively show the node classification effects of different algorithms in four datasets, and the best results are represented by bold numbers. From these tables, we can observe the following:

By comparing experimental results of TAD-GCN and seven baseline models, we find that attribute information of nodes can greatly affect classification effect of model, and the topology information of network plays a supplementary role. If the two are used together, the quality of node representation can be significantly improved, which also demonstrates the effectiveness of TAD-GCN model proposed in this paper.

In the four datasets of different sizes, the experimental results of TAD-GCN model are basically superior to the other seven comparison models, showing good classification performance. It shows that the model using GCN achieves better classification effect, because it can effectively integrate and utilize network structure information and node attribute information.

The improvement of TAD-GCN model's classification effect is slightly different in different datasets, which is due to the different number of categories and data size in different datasets. For Cora datasets with a large number of categories, TAD-GCN model has better generalization ability and can significantly improve the classification effect. For example, ACC and macro-F1 were 4.9% and 5.5% higher than GCN model, respectively. However, for the ACM dataset with only 3 categories, the classification performance is only 3.2% and 3.06% higher than that of the best benchmark framework.

Based on the experimental results in Tables 2, 3 and 4 and the above analysis, it is shown that TAD-GCN has high performance in node classification task and can accurately predict the nodes with unknown labels in the network, which

further proves that TAD-GCN can extract and utilize more effective information from the original network.

### Link prediction

To evaluate the ability of node embedding representation to reconstruct network original topology, which is mainly recommended in practical applications, we conduct link prediction task. In this section, 10% edges are randomly removed from the network link set as positive examples, and the same number of non-existent edges as positive examples is randomly generated as negative examples. These two kinds of edges constitute the test set together, and the rest of the links serve as the training set. In this section, AUC value, a commonly used evaluation index, is used to evaluate the performance of the algorithm in link prediction task. In Section ["Evaluation metrics"](#), the evaluation indicators are displayed. Table 5 demonstrates the comparison of TAD-GCN results with other benchmark algorithms in link prediction tasks. Among them, TAD-GCN-1 and TAD-GCN-2 represent the link prediction results obtained by using formulas (27) and (28) for the total objective function, respectively.

The link prediction results of TAD-GCN model on three datasets are all higher than those of other reference methods. Among them, TAD-GCN is significantly better than DeepWalk which only considers network topology. Second, TAD-GCN is a significant improvement over GAE, which considers both topology and node attributes. Finally, compared with JAME, the AUC and AP ratios of TAD-GCN are slightly 1–2% higher. An important reason is that the TAD-GCN model not only takes advantage of the advantages of multi-task learning, but also optimizes the model by considering two regularization terms, consistency constraint and difference constraint, which further enhances the extraction of structural information of the network by the model and finally obtains higher quality node representation.

Based on Table 5 and the above analysis, TAD-GCN can be used to predict missing or potential links in the network more accurately. This shows that TAD-GCN node embedding can more effectively retain the topological information of the original network and has an advantage in link prediction task.

### Node clustering

Node clustering aims to infer the cluster distribution of nodes in the network based on the topology and node attribute information. For example, community detection problem can be converted into node clustering problem in network analysis. The main steps are to first learn the low-dimensional node embedding representation through the network representation algorithm and then use k-means to achieve the node clustering task. This section uses three clustering evaluation

**Table 2** Node classification results of different methods (20)

Dataset	Cora		CiteSeer		ACM		PubMed	
	ACC	Ma-F1	ACC	Ma-F1	ACC	Ma-F1	ACC	Ma-F1
DeepWalk	0.293	0.281	0.435	0.381	0.627	0.621	0.452	0.446
LINE	0.178	0.182	0.327	0.318	0.413	0.401	0.409	0.385
ChebyNet	0.534	0.476	0.698	0.659	0.752	0.749	0.693	0.662
GCN	0.567	0.525	0.703	0.675	0.878	0.878	0.739	0.701
GAT	0.584	0.544	0.725	0.681	0.874	0.874	0.745	0.719
JAME	0.589	0.548	0.722	0.671	0.869	0.862	0.767	0.743
MTGAE	0.552	0.536	0.713	<b>0.699</b>	0.895	0.894	0.790	0.783
TAD-GCN	<b>0.599</b>	<b>0.563</b>	<b>0.735</b>	0.695	<b>0.910</b>	<b>0.908</b>	<b>0.836</b>	<b>0.828</b>

The best performance is in bold

**Table 3** Node classification results of different methods (40)

Dataset	Cora		CiteSeer		ACM		PubMed	
	ACC	Ma-F1	ACC	Ma-F1	ACC	Ma-F1	ACC	Ma-F1
DeepWalk	0.362	0.333	0.452	0.432	0.630	0.618	0.468	0.457
LINE	0.251	0.254	0.333	0.324	0.458	0.458	0.43	0.411
ChebyNet	0.582	0.535	0.716	0.683	0.816	0.813	0.755	0.724
GCN	0.606	0.556	0.731	0.697	0.891	0.890	0.796	0.758
GAT	0.629	0.583	0.730	0.696	0.886	0.886	0.792	0.746
JAME	0.631	<b>0.614</b>	0.747	<b>0.715</b>	0.89	0.889	0.832	0.818
MTGAE	0.590	0.572	0.739	0.725	0.904	0.903	0.821	0.8
TAD-GCN	<b>0.649</b>	0.605	<b>0.762</b>	0.708	<b>0.917</b>	<b>0.916</b>	<b>0.847</b>	<b>0.839</b>

The best performance is in bold

**Table 4** Node classification results of different methods (60)

Dataset	Cora		CiteSeer		ACM		PubMed	
	ACC	Ma-F1	ACC	Ma-F1	ACC	Ma-F1	ACC	Ma-F1
DeepWalk	0.406	0.379	0.489	0.480	0.670	0.669	0.485	0.473
LINE	0.297	0.309	0.354	0.344	0.504	0.499	0.434	0.421
ChebyNet	0.598	0.542	0.733	0.703	0.854	0.853	0.803	0.774
GCN	0.620	0.562	0.745	0.712	0.905	0.905	0.811	0.787
GAT	0.644	0.596	0.748	0.716	0.904	0.903	0.835	0.812
JAME	0.663	<b>0.630</b>	0.755	0.724	0.913	0.912	0.838	0.826
MTGAE	0.607	0.586	0.734	0.703	0.911	0.910	0.829	0.807
TAD-GCN	<b>0.669</b>	0.627	<b>0.768</b>	<b>0.735</b>	<b>0.924</b>	<b>0.924</b>	<b>0.858</b>	<b>0.842</b>

The best performance is in bold

indexes, which have been explained in Section "Evaluation metrics". The average value of evaluation indexes obtained from ten experiments is taken as the final clustering effect. Table 6 shows the experimental results of node clustering of all algorithms on three datasets respectively.

TAD-GCN achieves the best performance which can be seen from Table 6. In addition, methods (DAEGC and NAGCN) considering both result information and attribute

information. TAD-GCN is better than DeepWalk, which is based only on network structure information. It is remarkably that the multi-task-oriented learning method (TAD-GCN), which considers both result and attribute information simultaneously, performs better in the clustering experiment on three datasets, which further verifies the validity of the model proposed in this paper.

**Table 5** Link prediction results of different methods

Dataset	Cora		CiteSeer		ACM		PubMed	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DeepWalk	0.805	0.828	0.732	0.762	0.553	0.554	0.930	0.945
LINE	0.869	0.900	0.857	0.899	0.636	0.673	0.934	0.947
GAE	0.843	0.881	0.787	0.841	0.617	0.675	0.961	0.961
JAME	0.987	0.972	0.985	0.979	0.924	0.919	0.983	0.980
MTGAE	0.940	0.952	0.938	0.955	0.895	0.899	0.940	0.949
TAD-GCN-1	<b>0.993</b>	<b>0.993</b>	<b>0.993</b>	<b>0.992</b>	<b>0.937</b>	<b>0.927</b>	<b>0.990</b>	<b>0.988</b>
TAD-GCN-2	<b>0.990</b>	<b>0.991</b>	<b>0.990</b>	<b>0.992</b>	<b>0.937</b>	<b>0.926</b>	<b>0.991</b>	<b>0.996</b>

The best performance is in bold

**Table 6** Node clustering results of different methods

Dataset	Cora			CiteSeer			ACM			PubMed		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
DeepWalk	0.529	0.384	0.291	0.390	0.131	0.137	0.756	0.564	0.575	0.647	0.238	0.255
RWR-GAE	0.669	0.481	0.417	0.616	0.354	0.343	0.846	0.697	0.612	<b>0.726</b>	0.355	0.370
DAEGC	0.704	0.528	0.496	0.672	0.397	0.410	0.891	0.725	0.714	0.671	0.266	0.278
MAGCN	0.715	0.598	0.532	0.711	0.458	0.462	0.913	0.726	0.749	0.691	0.331	0.321
TAD-GCN	<b>0.746</b>	<b>0.612</b>	<b>0.584</b>	<b>0.759</b>	<b>0.479</b>	<b>0.483</b>	<b>0.928</b>	<b>0.743</b>	<b>0.798</b>	0.717	<b>0.362</b>	<b>0.381</b>

The best performance is in bold

## Visual analysis

In order to intuitively evaluate the node embedding learned by model TAD-GCN in this paper more accurately, t-SNE [38] visualization algorithm is used to reduce the dimension of node embedding representation to two-dimensional space. Taking CiteSeer and ACM datasets as examples, representative network embedding methods DeepWalk and GCN and TAD-GCN model are visualized to represent dimensionality reduction of learned nodes. Each point in the two-dimensional plane corresponds to a node in the original network, and different categories are distinguished by different colors. The two-dimensional vector representation of these points can retain the structure of the original network, that is, the more concentrated the points with the same color are, the more separated the points with different colors are, which means the visualization effect is better. Figure 3 shows the node visualization results of CiteSeer and ACM datasets.

Figures 3 (a1) and (b1), respectively, show the visualization results of node representation obtained by DeepWalk on CiteSeer and ACM datasets. It can be seen that the nodes of six labels in CiteSeer datasets are mixed together, and it is difficult to distinguish and there is no obvious cluster boundary. Most of the nodes of the three labels in the ACM dataset

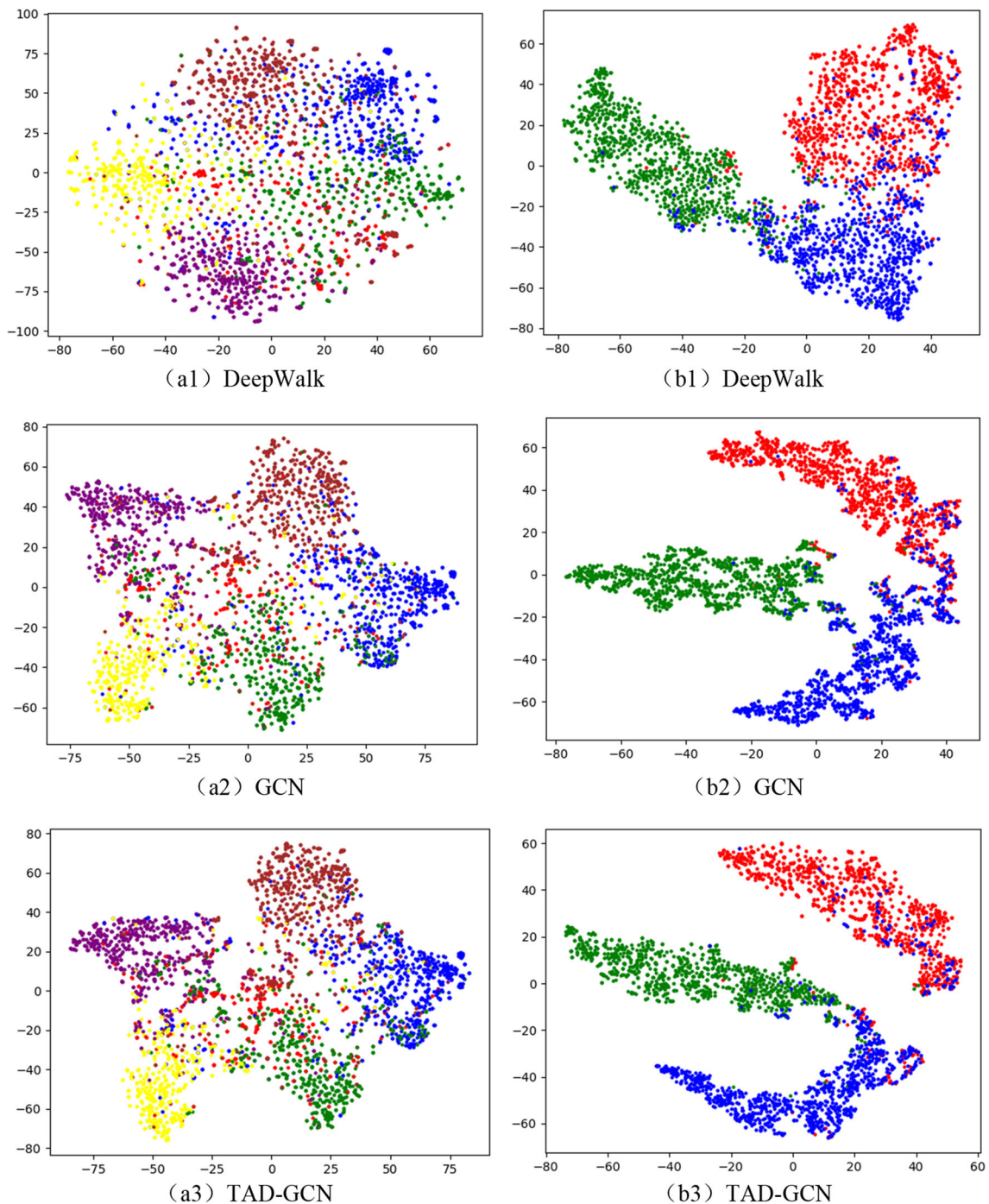
are separated into three clusters, but the boundaries of nodes of different labels are not clear.

Figures 3 (a2) and (b2), respectively, show the visualization results of node representation obtained by GCN on CiteSeer and ACM datasets. It can be seen that the visualization results of node representation obtained by GCN are better than DeepWalk, which can be slightly obvious that it is divided into six clusters (CiteSeer) and three clusters (ACM). However, there is no clearer boundary between clusters, and the nodes in the cluster are more discrete.

Finally, Fig. 3 (a3) and (b3), respectively, show the visualization results of TAD-GCN node representation on CiteSeer and ACM datasets, from which it can be seen that TAD-GCN has the best performance compared with the other two models. Although a small number of nodes with different labels are not completely separated, the boundary between clusters is very clear, and the nodes within clusters are more compact. Six clusters (CiteSeer) and three clusters (ACM) can be clearly observed.

Therefore, the TAD-GCN model has strong competitiveness in visualization tasks, which can obtain better node embedding representation and effectively capture the topology of the original network. This further proves that the node representation obtained by TAD-GCN model is more conducive to the completion of downstream tasks.



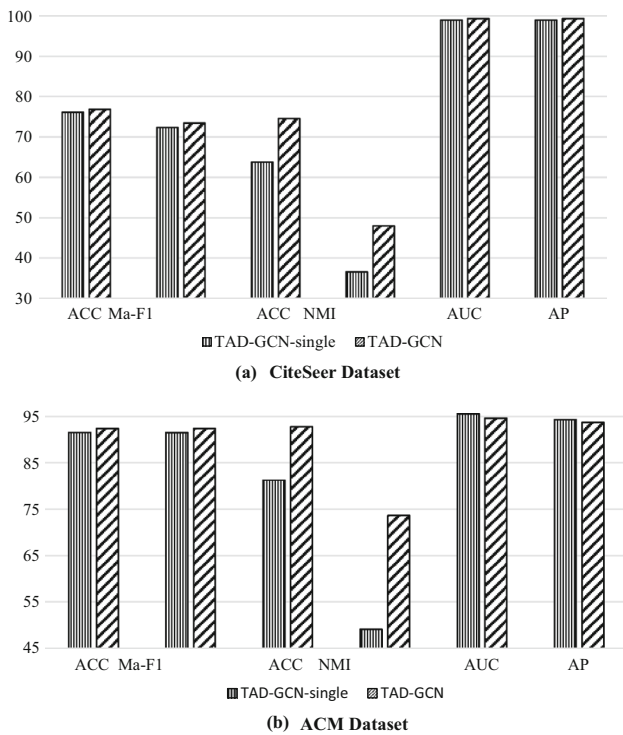


**Fig. 3** Visualization results of CiteSeer (left) and ACM (right) datasets

### Ablative experiment

First, for reason of verifying the effectiveness of multitask, this section compared the TAD-GCN model with its three variants (TAD-GCN-cla, TAD-GCN-lp and TAD-GCN-clu) on CiteSeer and ACM datasets. TAD-GCN-cla, TAD-GCN-lp and TAD-GCN-clu indicate that the node classification

task, link prediction task and node clustering task are completed separately. The results which are shown in Fig. 4 (a) and (b) represent the comparison of the results of TAD-GCN model and its three variants on CiteSeer and ACM datasets, respectively. ACC and macro-F1 were the evaluation indexes of classification task results, ACC and NMI were the evaluation indexes of clustering task results, and AUC and AP were the evaluation indexes of link prediction task results.

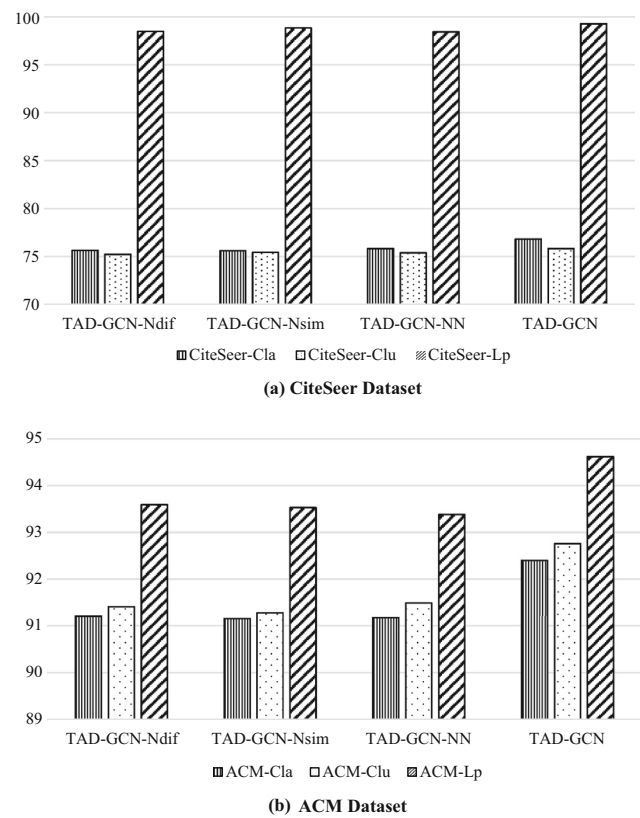


**Fig. 4** Ablative experimental results of multi-task learning

As can be seen from Fig. 4, in ACM and CiteSeer datasets, TAD-GCN model performed better than individual tasks in classification and link prediction tasks. In addition, TAD-GCN model performed better than TAD-GCN-cla in clustering tasks. TAD-GCN-lp and TAD-GCN-clu completed corresponding individual tasks, respectively. This shows that the multi-task learning module used in the TD-GCN model proposed in this paper is very effective, which cannot only complete multiple tasks such as node classification, clustering and link prediction at the same time, but also achieve good performance in the completion effect.

Finally, in order to verify the validity of regularization terms, this section compares the TD-GCN model with its three variants (TD-GCN-Ndif, TD-GCN-Nsim and TD-GCN-NN) on CiteSeer and ACM datasets (see Fig. 5). TAD-GCN-Ndif, TAD-GCN-Nsim and TAD-GCN-NN represent the removal of difference constraint, similarity constraint and regularization term in TAD-GCN model, respectively. ACC index results are used in classification task and clustering task, and ROC index results are used in link prediction task.

Figure 5 shows that the completion effect of TAD-GCN on all tasks is better than that of the other three regularization item variants in the above four datasets, indicating that considering the two constraints in the regularization item at the same time is to the benefit of improving the model capacity. In addition, in the clustering task, the variation without regularization term is basically better than the variation with



**Fig. 5** Ablative experimental results of regularization terms

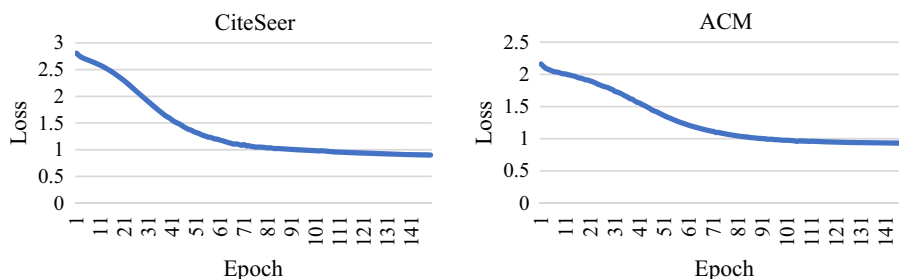
only one regularization term, which means that only when both regularization terms are considered in the model can they play an important role, and the two regularization terms constrain and promote each other.

### Convergence analysis

For purpose of verifying the convergence of TAD-GCN model, taking the model training on CiteSeer and ACM datasets as an example, the loss value of TDA-GCN model in the training process is shown in Fig. 6. From Fig. 6, the loss value curves of the model are monotonically decreasing and gradually converging.

### Conclusion

In this paper, we propose an original multi-task network embedding approach with dual-channel GCN, named TAD-GCN. More specifically, TAD-GCN considers utilizing relevant information of multi-view and multitasks through dual-channel GCN and attention mechanism. Besides, we enhance our model by introducing regularization term and multitask-oriented module which can make the multi-view and task-related information better captured. Therefore, our

**Fig. 6** Loss curve of the model

model is able to effectively learning the multi-view networks and simultaneously solve multiple downtown tasks. We manipulate four datasets to conduct extensive experiments, and our model present significantly superior to the excellent baselines by the empirical results. In the future, the model will be considered for extension to more complicated networks.

**Funding** This work was supported by National Natural Science Foundation of China [grant numbers 61876101 and 61802234].

**Data availability** The data is publicly available.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Cui P, Wang X, Pei J, Zhu W (2019) A survey on network embedding. *IEEE Trans Knowl Data Eng* 31(5):833–852
- Tang J, Qu M, Wang M, et al (2015) Line: large-scale information network embedding. *Proceedings of the 24th international conference on world wide web* 1067–1077
- Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp 855–864
- Wang D, Cui P, Zhu W (2016) Structural deep network embedding. *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp 1225–1234
- Ribeiro L, Saverese P, Figueiredo D (2017) struc2vec: learning node representations from structural identity[C]. *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp 385–394
- Wang J, Liang J, Yao K et al (2021) Graph convolutional autoencoders with co-learning of graph structure and node attributes. *Pattern Recognition* 121:108215
- Sc A, Wei S, Mz A (2022) Structure information learning for neutral links in signed network embedding. *Inform Process Manag* 59(3):102917
- Defferrard M, Bresson X et al (2016) Convolutional neural networks on graphs with fast localized spectral filtering. *Adv Neural Inform Proces Syst* 29:3844–3852
- Gao H, Huang H (2018) Deep attributed network embedding. *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp 3364–3370
- Chen J, Zhong M, Li J et al (2021) Effective deep attributed network representation learning with topology adapted smoothing. *IEEE Transact Cybern* 52(7):5935–5946
- Huang J, Li Z, Zheng VW et al (2018) Unsupervised multi-view nonlinear graph embedding. In: *UAI*, pp 319–328
- Yao K, Liang J, Liang J et al (2022) Multi-view graph convolutional networks with attention mechanism. *Artif Intell* 307:103708
- Zhang Z, Yang H, Bu J et al (2018) ANRL: attributed network representation learning via deep neural networks. *Proc IJCAI* 18:3155–3161
- Kipf TN, Welling M (2016) Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*
- Pedronette D, Latecki LJ (2021) Rank-based self-training for graph convolutional networks. *Inform Process Manag* 58(2):102443
- Zhu W, Liu S, Liu C (2021) Incorporating syntactic and phonetic information into multimodal word embeddings using graph convolutional networks. *Inform Proces Manag* 58(6):102709
- Li Q, Han Z, Wu X M (2018) Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the 32nd AAAI conference on artificial intelligence*, 538–545
- Nt H, Maehara T (2019) Revisiting graph neural networks: all we have is low-pass filters. *arXiv preprint arXiv. 1905.09550*
- Wu F, Souza A, Zhang T, et al (2019) Simplifying graph convolutional networks. *Int Conference on Machine Learning*. PMLR, pp 6861–6871
- Zhou J, Cui G, Hu S et al (2020) Graph neural networks: a review of methods and applications. *AI Open* 1:57–81
- Ding Y, Wei H, Pan Z et al (2020) Survey of network representation learning. *Comput Sci* 47(09):52–59
- Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: *Advances in neural information processing systems*, p 30
- Veličković P, Cucurull G, Casanova A, et al (2017) Graph attention networks. *arXiv preprint arXiv.: 1710.10903*
- Bruna J, Zaremba W, Szlam A, et al (2014) Spectral networks and locally connected networks on graphs. *Proceedings of the 2nd international conference Learning Representation*, pp 1–5
- Thomas N. Kipf, Max Welling (2017) Semi-supervised classification with graph convolutional networks. *Proceedings of ICLR*
- Lai D, Wang S, Chong Z et al (2021) Task-oriented attributed network embedding by multi-view features. *Knowledge-Based Syst* 232:107448
- Tran PV (2018) Multi-task graph autoencoders. *Adv Neural Inform Process Syst* 1811.02798

28. Rizi FS, Granitzer M (2020) Multi-task network embedding with adaptive loss weighting. Proceedings of 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 1–5
29. Zhang X, Zhang X, Liu H et al (2016) Multi-task multi-view clustering. *IEEE Trans Knowl Data Eng* 28(12):3324–3338
30. Huang H, Song Y, Wu Y et al (2020) Multitask representation learning with multiview graph convolutional networks. *IEEE Transact Neural Netw Learn Syst* 33(3):983–995
31. Meng Z, Liang S, Bao H, et al (2019) Co-embedding attributed networks. Proceedings of the twelfth ACM international conference on web search and data mining, pp 393–401
32. Wang X, Ji H, Shi C, et al (2019) Heterogeneous graph attention network. Proceedings of the WWW' 19: The World Wide Web Conference, pp 2022–2032
33. Yang H, Pan S, Zhang P, et al (2018) Binarized attributed network embedding. 2018 IEEE International conference on data mining (ICDM). IEEE, pp 1476–1481
34. Bojchevski A, Günnemann S (2018) Deep gaussian embedding of graphs: unsupervised inductive learning via ranking. Proceedings of International Conference on Learning Representations
35. Perozzi B, Rfou RA, Skiena S (2014) DeepWalk: online learning of social representations. Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining, pp 701–710
36. Wang C, Pan S, Hu R, et al (2019) Attributed graph clustering: a deep Attentional embedding approach. Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI' 19). AAAI Press, pp 3670–3676
37. Cheng J, Wang Q, Tao Z et al (2021) Multi-view attribute graph convolution networks for clustering. Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence 2021:2973–2979
38. Maaten L, Hinton G (2008) Visualizing data using t-SNE. *J Mach Learn Res* 9(2605):2579–2605
39. Ruder S (2017) An overview of multi-task learning in deep neural networks. <http://arxiv.org/abs/1706.05098>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.