International Journal of Machine Learning and Cybernetics (2024) 15:3733–3743
https://doi.org/10.1007/s13042-024-02117-3

ORIGINAL ARTICLE

Robust graph neural networks with Dirichlet regularization and residual connection

Kaixuan Yao[1] · Zijin Du[1] · Ming Li[2] · Feilong Cao[3] · Jiye Liang[1]

Received: 8 March 2023 / Accepted: 4 February 2024 / Published online: 7 April 2024

Abstract

Graph Neural Network (GNN) has attracted considerable research interest in various graph data modeling tasks. Most GNNs require efficient and sufficient label information during training phase. However, in open environments, the performance of existing GNNs sharply decrease according to the data (structure, attribute and label) missing and noising. Several recent attempts have been made to improve the performance and robustness of GNNs, most of which are contrastive learning-based and auto encoder-based strategies. In this paper, a semi-supervised learning framework is proposed for graph modeling tasks, i.e., robust graph neural network with Dirichlet regularization and Residual connection (DRGNN). Specifically, the structure and feature of the original graph are both masked to generate the masked graph, which is sent to the graph representation learning block (encoder) to learn the latent node representation. Additionally, an initial residual connect is introduced into the graph representation learning block to directly transmit the original node feature to the last layer to retain the inherent information of the node itself. Finally, the whole network is jointly optimized by the structure reconstructed loss, feature reconstructed loss and the classification loss. Note that a Dirichlet regularization constraint is introduced into the learning objective to dominate the latent node representation into a local smoothing scenario, which is more conforms with the manifold assumption of the graph representation learning. Extensive experiments demonstrate the state-of-the-art accuracy and the robustness of the proposed DRGNN on benchmark datasets.

Keywords Graph neural networks · Dirichlet regularization · Residual connection · Self-supervised learning · Mask strategy

✉ Jiye Liang
ljy@sxu.edu.cn

Kaixuan Yao
yaokx2@gmail.com

Zijin Du
thetempest0302@163.com

Ming Li
mingli@zjnu.edu.cn

Feilong Cao
feilongcao@gmail.com

[1] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 030006, Shanxi, China

[2] Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua 321004, Zhejiang, China

[3] Department of Applied Mathematics, China Jiliang University, Hangzhou 310018, Zhejiang, China

1 Introduction

Many practical applications study graph structure data (non-Euclidean data), such as the urban traffic network [1], the user-product network in recommendation systems [2], social networks in computational social sciences [3], and so on. The irregularity and complexity of graph structure data tremendously impose significant challenges on traditional machine learning methods transferring to graph data mining tasks. Recently, graph convolutional networks (GCNs) [4, 5] have attracted much attention in light of their excellent results on various graph structure data mining tasks. Technically, GCN [6] and its variants investigate to define effective convolution computation operators with the ability of feature representation from different hops of neighbors, i.e., generalize the convolution operation to the graph domain as different neighbor aggregation strategies or message passing schemes [7]. Here we omit a comprehensive review of the

existing models. Interested readers can refer to survey papers [8, 9] for more details.

Graph neural networks and their extensions have achieved remarkable attention in various tasks, their success is largely underpinned by a sufficient number of available labels for learning the discriminating representations. However, data labeling is very time-consuming and lacking in many practical applications. Insufficient label information severely limits the expressive power of GNN and most of its extensions. On the other hand, in open environments, the structure and the attribute of the graph data may be missing, noising or adversarial attacked, which have a great influence on the robustness and security of GNNs.

Recently, several efforts have been made to improve the weakly-supervised learning capability of GNNs [10, 11]. Contrastive learning, a widespread technique for self-supervised learning in computer vision [12, 13], has been developed to graph modeling tasks under weakly-supervised scenarios [14, 15]. However, the construction of the negative samples is a very laborious and time-consuming process, which are necessary for most graph contrastive learning objectives, e.g., GraphCL [16] and GCA [17]. Recently, there have been several works focus on augmentation-free based graph contrastive learning methods, which aims to avoid data augmentations [18–21]. Actually, these augmentation-free based graph contrastive learning methods need to use cluster strategies or similarity computing methods to recalculate similarity nodes (corresponds to generating positive/negative pairs), however, which will bring extra calculation cost. Further, the effectiveness of most graph contrastive learning models have serious dependence on high-quality data augmentation, e.g., JOAO [22] and DiGCL [23]. Finally, most graph contrastive learning methods highly rely on specialized and complex training strategies, e.g., BYOL [24] and GCC [25].

Another important self-supervised learning methods to graph modeling tasks are the autoencoders (AEs) [26, 27]. In contrast to graph contrast learning methods, most graph autoencoders (GAEs) do not require complex negative sampling and data augmentation, which are easily integrated to most GNNs without complex pre-works. However, the performance of most GAEs is weaker than most graph contrast learning models in node-level and graph-level classification tasks. A major reason is that most GAEs leverage graph structure reconstruction as the auxiliary optimization objectives, while the over-emphasized structure information is not consistent with the classification principle in some extent, e.g., VGAE [28], NWR-GAE [29] and DGI [30]. On the other hand, the loss function of most GAEs chooses the mean square error (MSE) loss, while the MSE loss may lead to global over-smoothing issue. Finally, existing GAEs directly take the input structure or attributes as the self-supervised information, however, once the input features are attacked or existing deficiencies, the GAEs can not be trained effectively to learn robust representations.

To sum up, there are three specific problems to be dealt with:

- Q1), how to improve the robustness of the GAEs?
- Q2), how to alleviate the over-emphasized structure information in GAEs?
- Q3), how to alleviate the global over-smoothing problem in GAEs?

Motivated by the above issues, in this paper, we develop a robust graph neural network with Dirichlet regularization and Residual connection (DRGNN). Specifically, for Q1), it has been shown that traditional deep neural networks are vulnerable to adversarial examples that are generated by adding perturbations and noise to original inputs. Essentially, GAE is a classical neural network-based model, as with the traditional deep neural networks, GAEs are also vulnerable to adversarial examples (generated by adding perturbations and noise). There has been several works focus on how to enhance the robustness of GAEs, such as [31, 32]. To address the Q1), we introduce the mask strategy into our model, i.e., the topology structure and attributes of the original graph are masked to generate the masked graph, which is sent to the graph representation learning block (encoder) to learn the latent node representation. For Q2), most GAEs leverage the graph structure reconstruction as the optimization objective to encourage the topological connections between neighbour nodes (such as [28, 33]), which has influenced and weakened the nature of the node itself to some extent. To alleviate this issue, we add an initial residual connect in the graph representation learning block, which can directly transmit the original node feature to the last layer to retain its inherent information. Then, the latent node representation is sent to the structure decoder and feature decoder to reconstruct the original graph structure and attribute. For Q3), most neural network based models adopt the MSE loss function and stochastic gradient descent (SGD) to as the training strategy, which may cause the global over-smoothing issue [34, 35]. To alleviate this issue, we introduce a Dirichlet regularization constraint to learn a latent node representation with local smoothing, which is more conformed to the classification principle. The learned latent node representation is utilized to classification tasks with a multilayer perceptron. Finally, the whole model is jointly optimized by the structure reconstruction loss, feature reconstruction loss and the classification loss. Our main contributions are summarized as follows:

1) We propose a robust residual graph neural network to improve the weakly-supervised learning capability and robustness of graph neural networks.

2) We introduce an initial residual connect to alleviate the over-emphasized structure information in GAEs, which can directly transmit the original node feature to the last layer to retain its inherent information.
3) We introduce a Dirichlet regularization constraint to learn a latent node representation with local smoothing, which is more conformed to the classification principle.

2 Related works

In this section, we review the related works, which include graph convolutional networks, graph contrastive self-supervised learning models and graph autoencoders.

2.1 Graph convolutional networks

As an important tool of graph representation learning, graph convolutional networks have achieved much attention after its proposition. Technically, graph convolutional network is defined by the spectral graph theory [36] and convolution theorem in graph domain (non-Euclidean domain). For a graph signal $x \in \mathbb{R}^N$, in graph signal processing filed, the graph Fourier transform (GFT) is defined as $\hat{x} = U^{\mathrm{T}}x$, and the inverse form is $x = U\hat{x}$ [37], where U represents the matrix of eigenvectors of the graph Laplacian matrix $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, D is the degree matrix, A is the adjacency matrix. Then, the graph convolution operation $*_{\mathcal{G}}$ can be defined in the graph Fourier domain [38], which is formulated as:

$$x \star_{\mathcal{G}} y = U((U^{\mathrm{T}}x) \odot (U^{\mathrm{T}}y)), \tag{1}$$

where $\odot$ represents the Hadamard product.

Furthermore, the graph convolution operation can be further expressed as a graph filter g_θ multiplying the graph signal x, i.e.,

$$g_\theta \star x = g_\theta(L)x = g_\theta(U\Lambda U^{\mathrm{T}})x = Ug_\theta(\Lambda)U^{\mathrm{T}}x, \tag{2}$$

where $\star$ denotes the convolution operation. Most of the earlier spectral-based graph convolutional networks are based on the Eqs. (1) and (2).

Graph convolutional neural network is a new graph representation learning tool that designed for graph modeling tasks, which is the combination of deep learning and graph computing method. There are two main types graph convolutional networks, i.e. spectral-based and spatial-based models. For the spatial-based graph convolutional networks, representative models like inductive representation learning on large graphs (Graph-SAGE) [39], diffusion-convolution operation-based graph neural networks (DCNN), geometric deep learning on graphs and manifolds using mixture model cnns (MoNet) [40], neural message passing-based graph neural networks (MPNN) [7], graph isomorphism networks (GIN) [41]. Representative spectral-based graph models such as Chebyshev polynomial-based fast localized spectral filtering model (ChebNet) [42], widely used graph convolutional network (GCN) [6], graph attention network (GAT) [43], shortest path graph attention network (SPAGAN) [44], variance reduction-based graph convolutional network [45], fast learning-based graph convolutional networks via importance sampling (FastGCN) [46]. And other spectral graph convolution operation-based GCN models such as Fast Haar transform-based graph neural network (HANet) [47], wavelet transform-based graph neural network (GWNN) [48].

2.2 Graph contrastive self-supervised learning

Contrastive learning is an important form of self-supervised learning, usually leverages positive and negative pairs of input samples to formulate the task of finding similar and dissimilar things for machine learning models. Recently, graph contrastive learning has become a popular strategy to benefit graph modeling tasks in a self-supervised way [49, 50]. Negative sampling is an essential process for most graph contrastive learning models. ProGCL [51] uses mixed distribution estimation to guide the generation of negative samples. Zhao et al. [52] use pseudo labels to alleviate the invalid negative problem. However, the negative sampling is a very labrious and time-consuming work. Additionally, the effectiveness of most graph contrastive learning models rely on high-quality data augmentation, e.g., feature perturbation-based [16, 53] and graph sampling-based [25, 54] data augmentation. Finally, most graph contrastive learning models require specialized and complex training strategies, e.g., BYOL [24] and GCC [25].

2.3 Graph autoencoders

Another important self-supervised learning model to graph modeling tasks is the graph autoencoder (GAE), a form of generative models, which is designed to encode the input graph to a low-dimensional embedding, and then decode the low-dimensional embedding to reconstruct the input graph. The first GAE-based model i.e., VGAE [28] take the graph convolutional network [6] as the encoder and the dot-product as the decoder to reconstruct the graph structure. Then most of its extensions also follow this pattern, i.e., take the graph reconstruction as objective, however, the over-emphasized structure information is not consistent with the classification principle in some extent, e.g., MSVGAE [55], NWR-GAE [29] and DGI [30]. The over-emphasized structure information may limit the performance of GAEs in classification tasks, how to alleviate the over-emphasized structure information is one purpose of our work. Recently, many feature reconstruction-based GAEs are proposed, e.g., GALA [56]

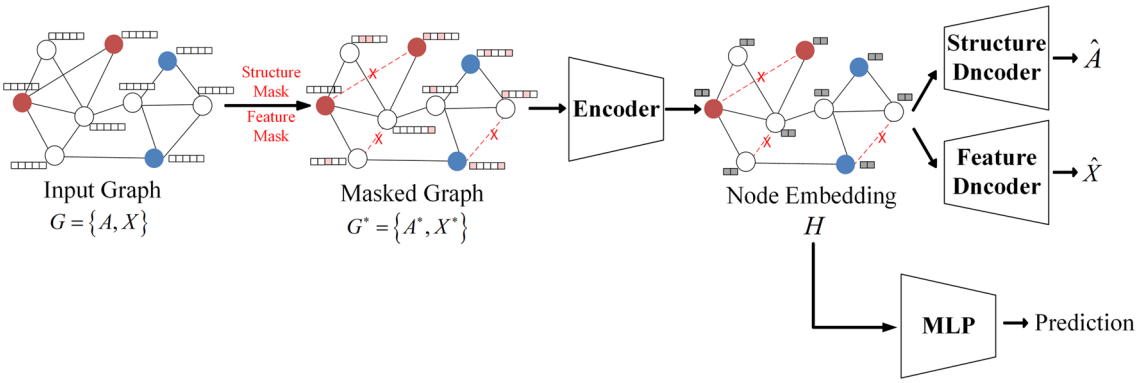

Fig. 1 The architecture of the proposed DRGNN, which includes the graph representation learning module (encoder), graph information reconstruction module (structure decoder and feature decoder), and the downstream task modeling module (Prediction)

and MGAE [57]. Most feature reconstruction-based GAEs adopt the MSE loss as the loss function, however, the MSE loss may cause the global over-smoothing problem. Actually, the graph representation learning follows the manifold hypothesis (i.e., local smoothing assumption). Finally, most existing GAEs directly take the input graph structure and node feature as the self-supervised information to train the model. However, in open environment, the providing input graph data maybe nosing and missing or attacked by artificial disturbance, which have a great influence on the performance and robustness of GAEs.

3 The proposed approach

In this section, we introduce the proposed robust graph neural network with Dirichlet regularization and Residual connection (DRGNN) in detail. In essence, the proposed DRGNN is a form of GAEs, the architecture of the proposed model is shown in Fig. 1.

3.1 Input graph and masked graph

Given a graph $G = \{V, A, X\}$, the set V comprises of N nodes, and each node includes a feature vector $x \in \mathbb{R}^M$, $X \in \mathbb{R}^{N\times M}$ denotes the feature matrix of G. $A \in \mathbb{R}^{N\times N}$ denotes the adjacency matrix of the graph.

Most GAEs directly take the graph $G = \{A, X\}$ as the input, however, once structure or feature of the graph G is missing or is attacked, the effectiveness of the GAEs degrades significantly. To improve the robustness of the GAEs, we introduce the mask strategy into GAEs from both structure and feature. Specifically, for the structure masking, we generate a binary structure mask M_s with size $\|E\|$ (where $\|E\|$ is the number of edges of the input graph G) to mask the edges in the graph structure, which follows a Bernoulli distribution $\mathcal{B}(p_s)$, where the p_s denotes the masking rate of the graph structure. Similarly, we generate a binary mask with size M for each node to mask the node feature of the input graph, which also follows a Bernoulli distribution $\mathcal{B}(p_f)$, where the p_f denotes the masking rate of each node feature. Finally, the masked graph structure and node feature are defined as:

$$\begin{aligned} \tilde{A} &= A \odot M_s, \\ \tilde{X} &= X \odot M_f, \end{aligned} \tag{3}$$

where $\tilde{A} \in \mathbb{R}^{N\times N}$ denotes the masked adjacency matrix and $\tilde{X} \in \mathbb{R}^{N\times M}$ denotes the masked feature matrix. $M_s \in \mathbb{R}^{N\times N}$ denotes the structure mask matrix and $M_f \in \mathbb{R}^{N\times M}$ denotes the feature mask matrix. $\odot$ denots the Hadamard product.

3.2 Encoder

The masked graph $\tilde{G} = \{V, \tilde{A}, \tilde{X}\}$ is fed to the Encoder to learn the low-dimension embedding representation of the graph:

$$H = f_{Encoder}(\tilde{A}, \tilde{X}), \tag{4}$$

where H denotes the leaned low-dimension representation of the graph. $f_{Encoder}$ denotes the Encoder, specifically, we proposed a residual-based graph convolutional network as the Encoder, which is formulated as:

$$\begin{aligned} H^{(l+1)} &= f_{Encoder}(\tilde{A}, H^{(l)}) \\ &= \sigma(\tilde{A}H^{(l)}\Theta^{(l)} + \lambda H^0), \end{aligned} \tag{5}$$

where $\sigma(\cdot)$ denotes the activation function, and the commonly choice is $ReLU(\cdot) = max(0, \cdot)$. $\Theta^{(l)}$ denotes the learnable parameters of the l-th layer. $H^{(l)}$ denotes the representation of the l-th layer of the Encoder. Specially, to make the residual feature size the same as the feature of the hidden layer, we add a Multilayer Perceptron (MLP) to the original feature, i.e., $H^{(0)} = f_{MLP}(\tilde{X})$, which is the introduced initial feature residual connect layer. Further, the introduction of the initial feature residual connection is designed to alleviate

the over-emphasized structure information. Specifically, the original feature of the graph can be directly passed to the hidden layer, which can make the node retain its original feature in some extent to alleviate the over-emphasized structure information that most GAEs faced.

3.3 Decoder

As shown in Fig. 1, the Decoder consists of two parts: Structure Decoder and Feature Decoder. As for the Structure Decoder, which is a basic component of most GAEs. The main objective of the Structure Decoder is reconstructing the graph structure by using the low-dimension node representation learned by the Encoder. Specifically, we take the inner product of the latent node representation to implement the Structure Decoder, which is formulated as:

$$\hat{a}_{ij} = f_{S-Decoder}(h_i, h_j) = sigmoid\left(h_i^{\mathrm{T}} h_j\right), \tag{6}$$

where h_j denotes the latent representation of the i-th node learned by Encoder. $\hat{a}_{ij}$ denotes the reconstructed adjacency weight between i-th node and j-th node.

For the Feature Decoder, its main purpose is that reconstruct the original node feature by utilizing the latent representation of the node, which is formulated as:

$$\hat{x}_i = f_{F-Decoder}(h_i) = f_{F-MLP}(\theta_F, h_i), \tag{7}$$

where $\hat{x}$ denotes the reconstructed feature of the i-th node. f_{F-MLP} denotes an MLP network structure that is used to increase the dimension to reconstruct the original high-dimension feature. The activation of the f_{F-MLP} is $ReLU(\cdot) = max(0, \cdot)$.

3.4 Prediction

As shown in Fig. 1, the latent node embedding learned by Encoder is finally used for the downstream tasks. In this paper, the main concern is focused on the node classification task, thus we design an MLP network for the classification task, which is formulated as:

$$\hat{Y} = f_{C\text{-}MLP}(\theta_C, H), \tag{8}$$

where $\hat{Y}$ denotes the final prediction category distribution. $f_{C\text{-}MLP}$ denotes the MLP network that used for the node classification task. θ_C denotes the trainable parameters of the $f_{C\text{-}MLP}$. The activation of the $f_{C\text{-}MLP}$ is *softmax*, which is formulated as:

$$\mathrm{softmax}(Y_{ij}^*) = \frac{\exp(Y_{ij}^*)}{\sum_{j=1}^{C} \exp(Y_{ij}^*)}, \tag{9}$$

where C denotes the number of categories.

3.5 Learning target

For the learning objective, the loss for the proposed DRGNN consists of two parts: reconstruction loss and classification loss. Specifically, the reconstruction loss contains graph structure reconstruction loss and feature reconstruction loss. For the feature reconstruction loss, different from other traditional GAEs, we directly recover the original features for each node under the masked scenario. It should be noted that several existing GAEs adopted the MSE loss as the feature reconstruction loss [56, 57]. However, the MSE loss may cause the global over-smoothing problem, while the graph representation learning follows the manifold hypothesis, i.e., the local smoothing assumption. This is a main reason that why few existing GAEs adopt the feature reconstruction loss (MSE loss) as the unique loss function to train the model. To alleviate the above issue, we introduce a Dirichlet regularization constraint into the feature reconstruction loss:

$$\mathcal{L}_{Feature} = \mathcal{L}_{MSE} + \mathcal{L}_{Dirichlet}, \tag{10}$$

where $\mathcal{L}_{Feature}$ denotes the feature reconstruction loss function. $\mathcal{L}_{MSE}$ denotes the MSE loss, which is formulated as:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2, \tag{11}$$

where N is the number of nodes. x_i is the raw feature of the i-th node. $\hat{x}_i$ is the reconstructed feature of the i-th node. The Dirichlet regularization constraint is formalized as:

$$\mathcal{L}_{Dirichlet} = trace(H^{\mathrm{T}} L H), \tag{12}$$

where H denotes the latent node representation learned by encoder. L denotes the Laplacian matrix. For Eq. (12), it is the definition of the Dirichlet (semi) norm, which is an analogue of the corresponding concept from complex/harmonic analysis. Recall the graph Laplacian is the analogue of the Laplacian in $\mathbb{R}^n$, essentially the Dirichlet regularization is reasonable to take as a possible measure of how much f deviates from constant, so we have the quantity $trace(H^{\mathrm{T}} L H)$ (i.e., a special case of Laplacian regularization).

For the graph structure reconstruction, we also reconstruct the original structure under the masked condition. Specifically, the structure reconstruction loss $\mathcal{L}_{Structure}$, calculating the loss between the reconstructed structure (i.e., the reconstructed adjacency matrix $\hat{A}$) and the raw graph structure (i.e., the raw adjacency matrix A). The structure reconstruction loss is formulated as:

$$\mathcal{L}_{Structure} = -\frac{1}{\|E\|}\Bigg(\sum a_{ij}\log \hat{a}_{ij} + (1-a_{ij})\log(1-\hat{a}_{ij})\Bigg), \tag{13}$$

where a_{ij} denotes the value of the i-th row and j-th column in original adjacency matrix A. The value of a_{ij} is 0 or 1, which denotes the neighborhood relationship between the i-th node and j-th node. $\hat{a}_{ij} \in [0, 1]$ denotes the element of the i-th row and j-th column in reconstructed adjacency matrix $\hat{A}$. $\|E\|$ denotes the number of edge number of the original graph. The main objective of the structure reconstruction loss is to make the reconstructed graph structure be as closely as possible with the original graph structure.

The purpose of reconstruction loss is to recover the original graph information, while the latent node embedding that dominated by the reconstruction task may not completely fit for the classification task. Thus, we introduce the classification loss into the loss function to make the latent node embedding more suitable for the classification task. For the classification loss, we choose the commonly used cross-entropy error in classification tasks as the loss function, which is formulated as:

Table 1 Statistics of the datasets

Dateset	Cora	Citeseer	Pubmed	ogbn-arxiv	Reddit
Nodes	2708	3327	19,717	169,343	231,443
Edges	5429	4732	44,338	1,166,243	11,606,919
Features	1433	3703	500	128	602
Classes	7	6	3	40	41

$$\mathcal{L}_{Classification} = -\sum_{k\in V_L}\sum_{j=1}^{C} Y_{kj}\ln \hat{Y}_{kj}, \tag{14}$$

where V_L represents the labeled nodes set. $Y \in \mathbb{R}^{N\times C}$ represents the label indicator matrix. $\hat{Y} \in \mathbb{R}^{N\times C}$ represents the predicted label indicator matrix learned by the DRGNN. Finally, the overall learning objective to train the whole model is formulated as:

$$\mathcal{L} = \mathcal{L}_{Feature} + \mathcal{L}_{Structure} + \mathcal{L}_{Classification}, \tag{15}$$

where $\mathcal{L}$ denotes the final loss function. The learning procedures of the proposed DRGNN are outlined in Algorithm 1.

Algorithm 1 Framework of DRGNN.

Input: Original graph: $G = \{V, A, X\}$; Encoder: $f_{Encoder}$ (see Eq. (4) and 5)); Feature Decoder: $f_{F-Decoder}$ (see (7)); Structure Decoder: $f_{S-Decoder}$ (see Eq. (6)).
Output: The predicted label indicator matrix $\hat{Y}$.

Initialize all parameters $\boldsymbol{\theta}$ of the whole model randomly;
while *not converged* **do**
  Generate masked graph structure $\tilde{A}$ and masked feature $\tilde{X}$ via Eq. (3);
  $H \leftarrow f_{Encoder}(\tilde{A}, \tilde{X})$;
  $\hat{A} \leftarrow f_{S-Decoder}(H)$;
  $\hat{X} \leftarrow f_{F-Decoder}(H)$;
  $\hat{Y} \leftarrow f_{C-MLP}(H)$;
  $\mathcal{L} \leftarrow \mathcal{L}_{Feature} + \mathcal{L}_{Structure} + \mathcal{L}_{Classification}$;
  Update $\boldsymbol{\theta}$ by gradient descent based on $\mathcal{L}$;
end while
return $\hat{Y}$.

Table 2 Summary of the semi-supervised node classification accuracy (%) on different datasets

Method	Datasets				
	Cora	Citeseer	Pubmed	Ogbn-arxiv	Reddit
DeepWalk	67.2	43.2	65.3	–	0.691
Planetoid	75.7	64.7	77.2	–	–
GAE	74.9	65.6	74.2	–	–
CCA-SSG	83.5	72.9	81.0	71.24	95.07
BGRL	82.7	71.1	79.6	71.64	94.22
ARVGA	79.5	66.0	81.5	–	–
GRACE	81.9	71.2	80.6	71.51	94.72
Graph-SAGE	75.3	68.2	77.4	–	–
GCN	81.5	70.3	79.0	71.74	95.3
GAT	83.0	72.5	79.0	71.10	95.9
DGI	82.3	71.8	76.8	70.34	94.0
GraphMAE	83.8	73.0	80.9	71.7	95.9
DRGNN (ours)	83.8 ± 0.2	73.1 ± 0.3	81.9 ± 0.2	72.8 ± 0.4	95.9 ± 0.2

4 Experiments

In this section, we compare the proposed DRGNN with several state-of-the-art methods under three benchmark datasets to verify its performance and robustness. The code is available on GitHub.[1]

4.1 Baselines

To demonstrate the effectiveness of our DRGNN, we compare it against several baseline models. Including traditional graph embedding models e.g., DeepWalk [58] and Planetoid [59]. GNN-based graph representation learning models, e.g., GraphSAGE [39], GCN [6], GAT [43], BGRL [24], CCA-SSG [60] and DGI [30]. Graph autoencoder-based models, e.g., GAE [28], ARVGA [61], GraphMAE [21], NWR-GAE [62]. Semi-supervised learning methods based on pseudo-label extension [63, 64], e.g., Co-training, Self-training, Union, Intersection, MultiStage, M3S, GRACE [65].

4.2 Datesets

Three benchmark citation datasets, Cora, Citeseer and Pubmed, are used in the experiments. For these datasets, the nodes and edges represent the papers and citation relations, respectively. The node feature is a vector of bag-of-words that represents the corresponding paper, each paper belongs to a research field that corresponding to its category. The details of datasets are summarized in Table 1. It should be noted that the public split for training/validation/testing sets [66] is used in the simulations. To verify the superiority of the proposed framework in a large data environment, we further conduct several comparative experiments under large-scale datasets, such as Ogbn-arxiv and Reddit.

4.3 Experiment setup

For the node classification task, the dimension of the latent layer in Encoder is set to 16. The optimizer used in experiments to train the proposed model is Adam [67] with learning rate 0.01 and weight decay 0.0005. The trainable weights of all layers are initialized by the Glorot uniform initializer [68]. The dropout operation [69] with dropping rate 0.5 is implemented for all layers. Experimental environment information is as follows: Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz, 36 kernel, 512 G memory, NVIDIA RTX 3090 GPU.

4.4 Results

The results of node classification task are summarized in Table 2. In particular, the experimental results of the compared models are obtained from their corresponding papers. As we can see, the experimental results show that the performance of the graph neural networks-based models is obviously better than traditional graph embedding-based models. Further, it is observed that the proposed DRGNN provides more apparent effectiveness comparing to traditional GAEs. This confirms the fact that the latent representation learned by the Encoder dominated by the generated task (graph reconstruction task) may not completely fit for the classification task. In other words, the proposed semi-supervised learning framework is demonstrated to have better representation capability. Further, to verify the superiority of the proposed framework in large data environment, we also conduct comparative experiments under large-scale datasets, the results show that our model still perform well in

[1] https://github.com/sxu-yaokx/DRGNN.

Table 3 Results of baseline methods under different number of layers

Datasets	Methods	Layers			
		2	8	32	64
Cora	GCN	80.4	69.5	60.3	28.7
	GCN (DropEdge)	82.0	75.8	62.5	49.5
	JKNet	80.2	80.7	81.1	71.5
	JKNet (DropEdge)	83.3	82.6	82.5	83.2
	IncepGCN	77.6	76.5	81.7	80.0
	IncepGCN (DropEdge)	82.9	82.5	83.1	83.5
	DRGNN (Ours)	83.7	83.9	84.0	84.5
Citeseer	GCN	67.6	30.2	25.0	20.0
	GCN (DropEdge)	70.6	61.4	41.6	34.4
	JKNet	80.2	80.7	81.1	71.5
	JKNet (DropEdge)	72.6	71.8	70.8	72.2
	IncepGCN	69.3	68.4	68.0	67.5
	IncepGCN (DropEdge)	72.7	71.4	72.6	71.0
	DRGNN (Ours)	73.0	73.3	73.7	73.9

Table 4 Classification accuracy (%) on Cora datasets with different label rates

Method	Label rate				
	0.5%	1%	2%	3%	4%
LP	56.4	62.3	65.4	67.5	69.0
GCN	50.9	62.3	72.2	76.5	78.4
Co-training	56.6	66.4	73.5	75.9	78.9
Self-training	53.7	66.1	73.8	77.2	79.4
Union	58.5	69.9	75.9	78.5	80.4
Intersection	49.7	65.0	72.9	77.1	79.4
MultiStage	61.1	63.7	74.4	76.1	77.2
M3S	61.5	67.2	75.6	77.8	78.0
DRGNN (ours)	62.7	71.5	76.3	79.6	82.0

Table 5 Classification accuracy (%) on Citeseer datasets with different label rates

Method	Label rate				
	0.5%	1%	2%	3%	4%
LP	34.8	40.2	43.6	45.3	46.4
GCN	43.6	55.3	64.9	67.5	68.7
Co-training	47.3	55.7	62.1	62.5	64.5
Self-training	43.3	58.1	68.2	69.8	70.4
Union	46.3	59.1	66.7	66.7	67.6
Intersection	42.9	59.1	68.6	70.1	70.8
MultiStage	53.0	57.8	63.8	68.0	69.0
M3S	56.1	62.1	66.4	70.3	70.5
DRGNN (ours)	58.5	62.5	69.0	70.6	71.5

Table 6 Classification accuracy (%) on Pubmed datasets with different label rates

Method	Label rate		
	0.03%	0.05%	0.1%
LP	61.4	66.4	65.4
GCN	60.5	57.5	65.9
Co-training	62.2	68.3	72.7
Self-training	51.9	58.7	66.8
Union	58.4	64	70.7
Intersection	52.0	59.3	69.4
MultiStage	57.4	64.3	70.2
M3S	59.2	64.4	70.6
DRGNN (ours)	62.8	70.7	73.2

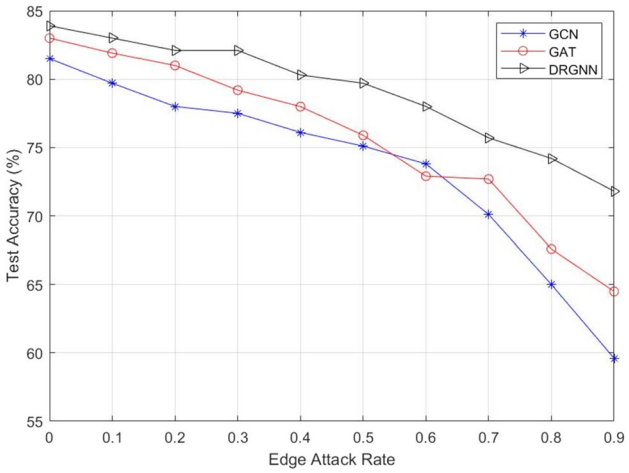

Fig. 2 Test performance comparison of GCN, GAT, and the proposed DRGNN on Cora dataset with different levels of random topology attack (RTA)

large-scale datasets. It should be noted that several results of large datasets not reported are because of unavailable code or out-of-memory. Furthermore, as for the over-smoothing issue, we compared the proposed model and baseline models with different number of layers. Results show that the performance of baseline models decreased with the increase of layer number, while the proposed DRGNN still perform well with the increase of layer number, as shown in Table 3.

4.5 Robustness analysis

As noted above, the superiority of the proposed DRGNN on the node classification task has been verified by comparison experiments. To indicate the robustness of the proposed DRGNN, we first implement baseline methods under different label rate on three benchmark datasets. The low label rate can lead to potential perturbations and affect

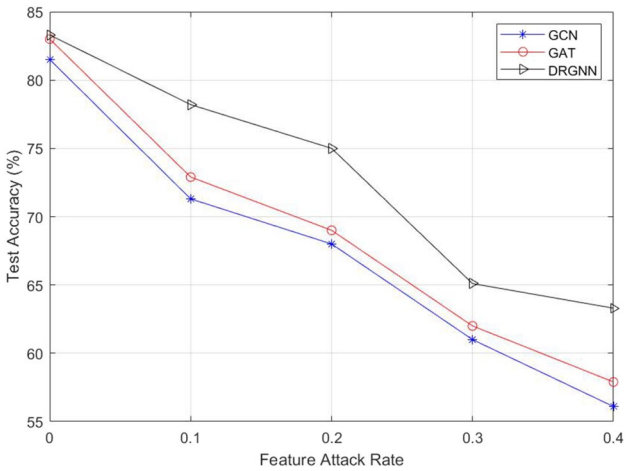

Fig. 3 Test performance comparison of GCN, GAT, and the proposed DRGNN on Cora dataset with different levels of random feature attack (RFA)

Table 7 Summary of the semi-supervised node classification accuracy (%) on different datasets

Method	Datasets		
	Cora	Citeseer	Pubmed
DRGNN-FR	78.3	68.2	78.3
DRGNN-SR	76.7	63.4	75.1
DRGNN-DR	78.5	71.3	78.6
DRGNN	83.8	73.1	81.9

the classification performance. Specifically, for the Cora and Citeseer datasets, we choose five different label rates $\{0.5\%, 1\%, 2\%, 3\%, 4\%\}$ to train the proposed DRGNN and baseline methods. For the Pubmed, the label retes are set to $\{0.03\%, 0.05\%, 0.1\%\}$. The results are summarized in Tables 4, 5 and 6. As we can see, the performance of our model is better than the baseline self-supervised learning methods under different label rates, which claims that the superiority of the proposed DRGNN in dealing with practical problems where the original graph data has few labels.

To further verify the robustness of the proposed model, we test the performance of DRGNN, GCN and GAT when dealing with two uncertainty issues (feature/structure missing) in the node classification tasks. That is, we use the Cora dataset and conduct two types of uncertainty: random topology attack (RTA) and random feature attack (RFA), which can lead to potential perturbations and affect the classification performance. Specifically, we random delete some edges/features with a given ratio to get the modified graph structure (i.e. adjacency matrix) and graph feature matrix. The experimental results are shown in Figs. 2 and 3. As we can see, the performance of all methods decays rapidly with respect to the random attack rate, while it is clear that the proposed DRGNN consistently outperforms GCN and GAT.

4.6 Ablation study

To investigate how the different parts of the proposed algorithm contribute to the final prediction accuracy, we conduct different setup of the proposed model to verify the effectiveness of different parts. Specifically, we consider three different setup: DRGNN Without Feature Reconstruction (DRGNN-FR), DRGNN Without Structure Reconstruction (DRGNN-SR), DRGNN Without Dirichlet Regularization (DRGNN-DR). The comparison results are summarized in Table 7. As we can see, the deletion of different parts affect the performance of the proposal model, which indirectly testifies the effectiveness of different parts.

5 Conclusion

In this paper, we propose a semi-supervised learning framework, i.e., robust graph neural network with Dirichlet regularization and residual connection, termed DRGNN. Specifically, we introduced a mask strategy into the graph autoencoder, i.e., the topology structure and the features of the original graph are masked before sent into the Encoder. Additionally, we add an initial residual connection into the graph representation learning block (Encoder) to directly transmit the original node feature to the last layer to retain the inherent information of the node itself. Finally, we introduce a Dirichlet regularization constraint into the learning objective to dominate the latent node representation into a local smoothing scenario, which is more conforms with the manifold assumption of the graph representation learning. We test the proposed DRGCN on several benchmarks and verify that the proposal is superior to the state-of-the-art on the node classification task. Further, we compare the proposed model with baseline methods under some uncertainty issues in the node classification tasks, the results show the superiority and robustness of the proposed DRGNN.

Acknowledgements This work was supported by the National Natural Science Foundation of China (No.U21A20473, 62176244, 62172370).

Data availability All relevant data are within the paper.

References

1. Xia J, Wang S, Wang X, Xia M, Xie K, Cao J (2022) Multi-view bayesian spatio-temporal graph neural networks for reliable traffic flow prediction. Int J Mach Learn Cybern 1–14

2. Yi Z, Song R, Li J, Xu H (2022) Neighbor-enhanced graph transition network for session-based recommendation. Int J Mach Learn Cybern 1–15
3. Sless L, Hazon N, Kraus S, Wooldridge M (2018) Forming k coalitions and facilitating relationships in social networks. Artif Intell 259:217–245
4. Xin Z, Chen G, Chen J, Zhao S, Wang Z, Fang A, Pan Z, Cui L (2022) Mgpool: multi-granular graph pooling convolutional networks representation learning. Int J Mach Learn Cybern 13(3):783–796
5. Yu S, Yang X, Zhang W (2019) Pkgcn: prior knowledge enhanced graph convolutional network for graph-based semi-supervised learning. Int J Mach Learn Cybern 10:3115–3127
6. Kipf T.N, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations
7. Gilmer J, Schoenholz S.S, Riley P.F, Vinyals O, Dahl G.E (2017) Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning, pp 1263–1272
8. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2020.2978386
9. Zhang Z, Cui P, Zhu W (2020) Deep learning on graphs: a survey. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2020.2981333
10. Hasanzadeh A, Hajiramezanali E, Boluki S, Zhou M, Duffield N, Narayanan K, Qian X (2020) Bayesian graph neural networks with adaptive connection sampling. In: Proceedings of the International Conference on Machine Learning, pp 4094–4104
11. Yang L, Li M, Liu L, Wang C, Cao X, Guo Y et al (2021) Diverse message passing for attribute with heterophily. Adv Neural Inf Process Syst 34:4751–4763
12. He K, Fan H, Wu Y, Xie S, Girshick R (2020) Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9729–9738
13. Zhang X, Wang S, Wu Z, Tan X (2022) Unsupervised image clustering algorithm based on contrastive learning and k-nearest neighbors. Int J Mach Learn Cybern 13(9):2415–2423
14. Duan W, Xuan J, Qiao M, Lu J (2022) Learning from the dark: boosting graph convolutional neural networks with diverse negative samples. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 36, pp 6550–6558
15. Duan W, Xuan J, Qiao M, Lu J (2022) Graph convolutional neural networks with diverse negative samples via decomposed determinant point processes. arXiv preprint arXiv:2212.02055
16. You Y, Chen T, Sui Y, Chen T, Wang Z, Shen Y(2020) Graph contrastive learning with augmentations. In: Proceedings of Advances in Neural Information Processing Systems, pp 5812–5823
17. Zhu Y, Xu Y, Yu F, Liu Q, Wu S, Wang L (2021) Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference, pp 2069–2080
18. Lee N, Lee J, Park C (2022) Augmentation-free self-supervised learning on graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 36, pp 7372–7380
19. Yu J, Yin H, Xia X, Chen T, Cui L, Nguyen Q.V.H (2022) Are graph augmentations necessary? simple graph contrastive learning for recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 1294–1303
20. Yang Y, Guan Z, Wang Z, Zhao W, Xu C, Lu W, Huang J (2022) Self-supervised heterogeneous graph pre-training based on structural clustering. In: Proceedings of the Advances in Neural Information Processing Systems, pp 16962–16974
21. Hou Z, Liu X, Cen Y, Dong Y, Yang H, Wang C, Tang J (2022) Graphmae: Self-supervised masked graph autoencoders. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp 594–604
22. You Y, Chen T, Shen Y, Wang Z (2021) Graph contrastive learning automated. In: Proceedings of the 38th International Conference on Machine Learning, pp 12121–12132
23. Tong Z, Liang Y, Ding H, Dai Y, Li X, Wang .C Directed graph contrastive learning. In: Proceedings of Advances in Neural Information Processing Systems
24. Thakoor S, Tallec C, Azar M.G, Azabou M, Dyer E.L, Munos R, Veličković P, Valko M (2021) Large-scale representation learning on graphs via bootstrapping. In: International Conference on Learning Representations
25. Qiu J, Chen Q, Dong Y, Zhang J, Yang H, Ding M, Wang K, Tang J (2020) Gcc: Graph contrastive coding for graph neural network pre-training. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1150–1160
26. Weng R, Lu J, Tan Y-P, Zhou J (2016) Learning cascaded deep auto-encoder networks for face alignment. IEEE Trans Multimed 18(10):2066–2078
27. Wang H, Sun J, Gu X, Song W (2022) A novel multi-scale and sparsity auto-encoder for classification. Int J Mach Learn Cybern 13(12):3909–3925
28. Kipf T.N, Welling M (2016) Variational graph auto-encoders. In: Proceedings of the NIPS 2016 Workshop on Bayesian Deep Learning
29. Tang M, Li P, Yang C (2021) Graph auto-encoder via neighborhood wasserstein reconstruction. In: International Conference on Learning Representations
30. Veličković P, Fedus W, Hamilton W.L, Liò P, Bengio Y, Hjelm R.D (2019) Deep graph infomax. In: International Conference on Learning Representations
31. Zhou X, Hu K, Wang H (2023) Robustness meets accuracy in adversarial training for graph autoencoder. Neural Netw 157:114–124
32. Yao K, Liang J, Liang J, Li M, Cao F (2022) Multi-view graph convolutional networks with attention mechanism. Artif Intell 307:103708
33. Hu Z, Dong Y, Wang K, Chang K.-W, Sun Y (2020) Gpt-gnn: Generative pre-training of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1857–1867
34. Zhou Y, Yang J, Zhang H, Liang Y, Tarokh V (2018) Sgd converges to global minimum in deep learning via star-convex path. In: International Conference on Learning Representations
35. Lai W-S, Huang J-B, Ahuja N, Yang M-H (2018) Fast and accurate image super-resolution with deep laplacian pyramid networks. IEEE Trans Pattern Anal Mach Intell 41(11):2599–2613
36. Chung F.R (1997) Spectral graph theory
37. Shuman DI, Narang SK, Frossard P, Ortega A, Vandergheynst P (2013) The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process Mag 30(3):83–98
38. Estrach J.B, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and deep locally connected networks on graphs. In: International Conference on Learning Representations
39. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of Advances in Neural Information Processing Systems, pp 1024–1034
40. Monti F, Bronstein M, Bresson X (2017) Geometric matrix completion with recurrent multi-graph neural networks. In: Proceedings of Advances in Neural Information Processing Systems, pp 3697–3707

41. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: International Conference on Learning Representations
42. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of Advances in Neural Information Processing Systems, pp 3844–3852
43. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: International Conference on Learning Representations
44. Yang Y, Wang X, Song M, Yuan J, Tao D (2019) SPAGAN: Shortest path graph attention network. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp 4099–4105
45. Chen J, Zhu J, Song L (2018) Stochastic training of graph convolutional networks with variance reduction. In: Proceedings of 35th International Conference on Machine Learning, pp 941–949
46. Chen J, Ma T, Xiao C (2018) FastGCN: fast learning with graph convolutional networks via importance sampling. In: International Conference on Learning Representations
47. Li M, Ma Z, Wang YG, Zhuang X (2020) Fast Haar transforms for graph neural networks. Neural Netw 128:188–198
48. Xu B, Shen H, Cao Q, Qiu Y, Cheng X (2019) Graph wavelet neural network. In: International Conference on Learning Representations
49. Suresh S, Li P, Hao C, Neville J. Adversarial graph augmentation to improve graph contrastive learning. In: Proceedings of Advances in Neural Information Processing Systems
50. Xu D, Cheng W, Luo D, Chen H, Zhang X. Infogcl: Information-aware graph contrastive learning. In: Proceedings of Advances in Neural Information Processing Systems
51. Xia J, Wu L, Wang G, Chen J, Li S.Z (2022) Progcl: Rethinking hard negative mining in graph contrastive learning. In: Proceedings of the 39th International Conference on Machine Learning, pp 24332–24346
52. Zhao H, Yang X, Wang Z, Yang E, Deng C (2021) Graph debiased contrastive learning with joint representation clustering. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, pp 3434–3440
53. Veličković P, Fedus W, Hamilton W.L, Liò P, Bengio Y, Hjelm RD (2019) Deep graph infomax. In: International Conference on Learning Representations
54. Hassani K, Khasahmadi AH (2020) Contrastive multi-view representation learning on graphs. In: Proceedings of the 37th International Conference on Machine Learning, pp 4116–4126
55. Guo Z, Wang F, Yao K, Liang J, Wang Z (2022) Multi-scale variational graph autoencoder for link prediction. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp 334–342
56. Park J, Lee M, Chang HJ, Lee K, Choi J (2019) Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 6519–6528
57. Wang C, Pan S, Long G, Zhu X, Jiang J (2017) Mgae: Marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp 889–898
58. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 701–710
59. Yang Z, Cohen W, Salakhudinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33nd International Conference on Machine Learning, pp 40–48
60. Zhang H, Wu Q, Yan J, Wipf D, Yu PS (2021) From canonical correlation analysis to self-supervised graph neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, pp 76–89
61. Pan S, Hu R, Long G, Jiang J, Yao L, Zhang C (2018) Adversarially regularized graph autoencoder for graph embedding. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp 2609–2615
62. Tang M, Li P, Yang C (2022) Graph auto-encoder via neighborhood wasserstein reconstruction. In: International Conference on Learning Representations
63. Li Q, Han Z, Wu X-M (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 3538–3545
64. Sun K, Lin Z, Zhu Z (2020) Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 5892–5899
65. Zhu Y, Xu Y, Yu F, Liu Q, Wu S, Wang L (2020) Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131
66. Yang Z, Cohen W.W, Salakhutdinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33nd International Conference on Machine Learning, pp 40–48
67. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: International Conference on Learning Representations
68. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp 249–256
69. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958