A Deep Graph Structured Clustering Network

XUNKAI LI\textsuperscript{1}, YOUPENG HU\textsuperscript{1}, YAOQI SUN\textsuperscript{2}, JI HU\textsuperscript{3}, JIYONG ZHANG\textsuperscript{2}, (Member, IEEE), AND MEIXIA QU\textsuperscript{1}

\textsuperscript{1}School of Mechanical and Information Engineering, Shandong University, Weihai 264209, China
\textsuperscript{2}School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China
\textsuperscript{3}School of Electronic Information, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Meixia Qu (mxqu@sdu.edu.cn)

This work was supported by the Natural Science Foundation, Shandong, under Grant ZR2018MA032.

ABSTRACT Graph clustering is a fundamental task in data analysis and has attracted considerable attention in recommendation systems, mapping knowledge domain, and biological science. Because graph convolution is very effective in combining the feature information and topology information of graph data, some graph clustering methods based on graph convolution have achieved superior performance. However, current methods lack the consideration of structured information and the process of graph convolution. Specifically, most of existing methods ignore the implicit interaction between topology information and feature information, and the stacking of a small number of graph convolutional layers leads to insufficient learning of complex information. Inspired by graph convolutional network and auto-encoder, we propose a deep graph structured clustering network that applies a deep clustering method to graph structured data processing. Deep graph convolution is employed in the backbone network, and evaluates the result of each iteration with node feature and topology information. In order to optimize the network without supervision, a triple self-supervised module is designed to help update parameters for overall network. In our model, we exploit all information of the graph structured data and perform self-supervised learning. Furthermore, improved graph convolution layers significantly alleviate the problem of clustering performance degradation caused by over-smoothing. Our model is designed to perform on representative and indirect graph datasets, and experimental results demonstrate that our model achieves superior performance over state-of-the-art models.

INDEX TERMS Autoencoder, deep graph convolutional network, deep graph clustering, unsupervised learning.

I. INTRODUCTION

Typically, graph structured data can be represented as \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) where \( \mathcal{V} \) represents node information and \( \mathcal{E} \) represents edge information. The graph data contains abundant information including node feature and topology information. Owing to the expressive power of the graph-based model, many applications in the real world can be represented as graph such as user-item interactions in recommendation systems, referrals in mapping knowledge domain, and protein molecular structures in biological science. Some recent research and applications have made considerable progress in graph data processing.

Graph convolution has been proposed, and many graph clustering methods based on it have been used to achieve state-of-the-art performance. Graph clustering is achieved by learning the distribution features of graph data. Graph convolutional network (GCN) \cite{1} is very important in the aspect of extracting the feature and topology information of graph. Existing deep graph clustering methods mainly rely on the topology structure of the restored graph data for the construction of the objective function. Some representative graph clustering methods have been proposed. Diederik P Kingma \textit{et al.} \cite{2} extracts the feature information of graph data to learn representations of node features. And the results of the model are expected to be as similar as possible to the node features of the original graph. Thomas Kipf \textit{et al.} \cite{3} adopts deep learning and graph convolution to perform clustering task. It uses results of model learning to reconstruct the topology of the original graph in the self-supervised training. Shirui Pan \textit{et al.} \cite{4} combines graph convolution operation with generative adversarial network (GAN). GAN is used to optimize the distribution of clustering results to improve performance. Xiaotong Zhang \textit{et al.} \cite{5} implements clustering
task efficiently by optimizing the classical graph convolution operation, which can only be used in few datasets to conduct clustering task.

All of the above methods aim to reconstruct the original topology or node features to optimize learning results so as to accomplish the clustering task, but they ignore the implicit interaction between feature information and topology information. And only a small number of GCN layers stack, resulting in not learning sufficient features. However, the graph clustering task requires sufficient consideration of graph structured data, and fully learn structured information.

Some clustering distribution optimization methods based on deep learning are widely applied. Xifeng Guo et al. [6] optimizes the label distribution of clustering results while preserving local structure information of data. Shirui Pan et al. [4] propose to optimize the clustering distribution results by taking the output results as the input of GAN. These methods are used to construct the self-supervised module of unsupervised graph clustering task.

Many clustering methods based on auto-encoder have been used to achieve state-of-the-art performance. In non graph data, the auto-encoder based on deep neural network (DNN) [2] learns the high-dimensional representation of the original data features to complete the clustering task. Because graph data contains more information, it is not suitable to deal with it directly. As mentioned above, the auto-encoder based on GCN [3] cannot comprehensively consider the node feature information when processing graph structured data. Because of the limitation of auto-encoder based on GCN and DNN, we propose a combination of two auto-encoders to process graph structured data.

In this article, we propose two auto-encoder modules based on GCN and DNN to fully consider the structured information in graph data, and we design deep GCN layers to learn structured information of the graph. However, as the number of GCN layers increases, an over-smoothing problem occurs, and it leads to the degradation of cluster performance. Our model has deep GCN layers with some tricks to alleviate the impact of the over-smoothing problem, and experimental results demonstrate the superior performance of our model. Inspired by the clustering distribution optimization method based on deep learning, we design a triple self-supervised module to perform end-to-end training of the model.

The main contributions of this article are in three folds:
(1) We propose a deep graph structured clustering network (DGSCN) that applies a deep clustering method to graph structured data processing. Both graph node feature information and topology information are considered.
(2) A method with deep GCN layers for clustering task is proposed. It improves the performance of clustering tasks and alleviates the influence of over-smoothing. The experiment mentioned in this article can be used for reference.
(3) We design a triple self-supervised module which contains the backbone network, GCN auto-encoder, and DNN auto-encoder to complete end-to-end self-supervised learning. The proposed DGSCN significantly outperforms state-of-the-art methods through our experiment.

II. RELATED WORK

Deep learning has many applications, such as complex image text detection [7], image understanding [8], video captioning [9], 3D room layout estimation [10], image retrieval [11]. The clustering task maintains high attention in academia. Recently some research about deep clustering have made considerable progress. Mathilde Caron et al. [12] adopts the clustering results as pseudo-labels in order to apply them to train deep neural networks with big data. Deep clustering method of [6] synthesizes the clustering loss of the auto-encoder, and reconstructing loss can jointly optimize clustering label assignments. Learning function et al. in [6] is suitable for clustering with local structure retention, and it helps auto-encoder to learn better data representations. The auto-encoder in [2] uses neural network to perform representations learning, which learns the high-dimensional interpretable features of original data. However, all of these methods focus on learning the representations of non-graph data feature.

For clustering work, graph clustering has been widely applied in various applications such as community networks [13], biological science [14], recommendation systems [15], network science [16] and 3D pose estimation [17]. In the past few years, various methods have been proposed for graph clustering. For indirect graph, a method of mining topology information [18] is proposed, a K nearest neighbor graph is calculated based on the original data to obtain the constraint relationship between the original data in adjacency matrix. This method provides the basis of structured clustering for indirect graph data.

Recently some graph clustering methods are based on shallow learning paradigms achieve state-of-the-art performance in network communities and attributed graphs. In network communities, a novel dubbed manifold regularized stochastic model (MrSBM) [19] was proposed, which is a probabilistic modeling method based on stochastic block models (SBMs). Besides modeling edges that are within or connecting blocks, MrSBM also considers modeling vertex features utilizing the probabilities of vertex-cluster preference and feature-cluster contribution. In order to consider both edge structure and node attributes, a novel latent factor model for community identification and summarization (LFCIS) [20] was proposed. LFCIS first formulates an objective function that evaluating the overall clustering quality taking into the consideration of both edge topology and node features in the network. In attributed graphs, FSPGA [21] was proposed to discover fuzzy structural patterns for graph analytics, which takes into consideration both the graph topology and attribute values. The competitive experimental results validate the effectiveness of them. The above works put forward a new way of processing graph structured data.

On the other hand, some graph clustering methods based on deep model have made great progress in original graph
In order to exploit information that considers both the data content and spatial structure of graph in specific tasks, a copolymerization method [22] and content dissemination [23] have been proposed. These methods do not take sufficient account into the topology information in the graph structure if applied directly to sparse raw graphs. In order to learn the fundamental structure information, some GCN-based clustering methods have been proposed. The unsupervised graph structure potential feature representations method [3] obtains the embedding of the nodes in the graph through the structure of the encoder-decoder to support the following tasks, such as link prediction. Chun Wang et al. [24] adopts attention networks to capture the importance of neighboring nodes, and uses KL divergence loss to supervise the training process of graph clustering. Shirui Pan et al. [4] combines GCN with GAN to optimize clustering distribution results. All the above clustering methods based on GCN hope to perform the clustering task by learning the topology relation of data, but they ignore the implicit interaction between feature information and topology information. Xiaotong Zhang et al. [5] optimizes convolution operations that are more suitable for clustering tasks, which outperforms over some specific datasets. However, it cannot be generalized. Deyu Bo et al. [25] considers information about the features of the data itself and adds potential representations of its own features to the GCN layer for integrated learning, but ignoring the spatial topology information of the graph structure itself. Furthermore, GCN layer of all the above clustering methods is only a simple stack, and it leads to insufficient learning of complex structured information. Some progress has been made in the deep GCN layers. Guohao Li et al. [26], Yu Rong et al. [27] increase the number of GCN layers, so the network can learn more high-dimensional features. And they alleviate over-smoothing problem caused by deep GCN layers and demonstrate superior performance in clustering. The limitation of the above method lies in the higher computational cost. In our method, we simplify the model as much as possible to reduce the extra cost of calculation.

In our case, we propose a special weighted network and deep graph convolution module. It contains topology information embedding module, feature information embedding module, deep graph convolution network module, and triple self-supervised module. The deep graph clustering method aims to combine deep representations learning with graph structure representations learning in clustering tasks. On this basis, GCN layers are optimized to alleviate the influence of over-smoothing and improve clustering performance.

### III. THE PROPOSED MODEL

In this section, we introduce the proposed deep graph structured clustering network model (DGSCN). Its overall framework is shown in Fig. 1 and the mathematical notations used
in the paper can be referred to Table 1. In this section, we’ll introduce their details

### TABLE 1. The mathematical notations used in the paper.

| Notation | Meaning |
|----------|---------|
| \( A \) | Adjacency matrix of graph data |
| \( D \) | Degree matrix of graph data |
| \( W \) | Weight matrix of model |
| \( GCN(\cdot) \) | Graph convolutional operation |
| \( p(\cdot) \) | Operation of probability matrix calculation |
| \( H^{(l)} \) | Output of GCN at the \((l-1)\) level |
| \( E(\cdot) \) | DNN encoding operation |
| \( D(\cdot) \) | DNN encoding operation |
| \( b \) | Bias matrix |
| \( X \) | Feature matrix |
| \( \phi \) | Nonlinear activation function, such as the ReLU(\cdot) |
| \( \mathcal{H} \) | DenseGCN convolution operation |
| \( F \) | GCN layer function shown in Eq. 1 |
| \( f \) | Vertex cascading function |
| \( \alpha, \beta, \gamma \) | Weight coefficient of variables |
| \( \eta(\cdot) \) | Splice function of the output vector in the previous layer |
| \( \epsilon \) | Weight representations of the final result distribution of the two auto-encoders |
| \( KL(\cdot) \) | KL divergence calculation operation |
| \( w_1, w_2, w_3 \) | Weight coefficient of variables in loss function |

Our model can be divided into four modules, and we introduce the process of proposing the model. In order to accomplish the clustering task well, it needs to consider the graph structured data comprehensively and use deep graph convolution networks to sufficiently learn. Graph data contains abundant structured information, including node feature information and topology information. Two auto-encoder modules based on DNN and GCN are proposed to learn node feature information and topology information respectively. These two auto-encoder modules are used to learn graph structured information. Deep GCN module uses the learning results of auto-encoders to generate clustering distribution results through deep graph convolution operation, and triple self-supervised module combines the above three modules to complete the network self-supervised training.

For two auto-encoder modules, the objective function is based on the reconstruction of its adjacency matrix and feature matrix. The result of their high-dimensional code is used to generate the clustering result distribution \( Q \), and then Student’s t distribution [28] is used by us as the kernel to generate target distribution \( P \). We expect the distribution of \( P \) and \( Q \) to be as similar as possible. They have the same meaning as \( P \), \( Q \) in Fig. 1. In the learning stage of the auto-encoder, the learning effect of backbone network layer is evaluated in real time. GCN methods are used in the first few layers, with which we use the first-order approximation of Chebyshev inequality as the convolution kernel to perform graph convolution operation [3]. DenseGCN [26] constructs dense links between layers, which is greatly enriched by data content. It is necessary for GCN to learn structured features of the data while alleviating the impact of the over-smoothing problem. After deep GCN extraction of the structured features at the last layer, predicted clustering result is generated.

KL divergence is calculated in order to expect the predicted distribution \( Z \) and the target distribution \( P \) to be as similar as possible. Finally, end-to-end self-supervised training is possible through the triple self-supervised module we proposed.

### A. GCN MODEL FOR AUTO-ENCODER

Through KNN method [18], the graph structure representations of the original data without topology relation is obtained. We use GCN module to learn the interpretability of the topology information. As the inputs of the network, feature matrix of graph nodes \( X \) and the adjacency matrix \( A \) represents the topology relation between nodes. The graph auto-encoder consists of GCN layers, which aims to learn the high-dimensional representations of the structural features in graph space. GCN layer can be expressed by the following formula:

\[
H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})
\]

where \( \tilde{A} = A + I_N \) is the adjacency matrix of the self-connected undirected graph, \( I_N \) is the identity matrix, \( \tilde{D} = \sum \tilde{A}_{ij} \) is degree matrix, \( W^{(l)} \) is the trainable weighted matrix of each layer, \( \sigma(\cdot) \) is a nonlinear activation function, such as the ReLU(\cdot), \( H^{(0)}_{l,j} \in \mathbb{R}^{N \times D} \) represents the input of the GCN at the \( l \) level, so \( H^{(0)} = X \). \( N \) represents the number of nodes in the graph, and \( D \) represents the feature dimension.

The GCN module is used to construct an auto-encoder to learn the high-dimensional feature representations of spatial features in encoding stage. Overall process can be summarized by the following formula:

\[
Z = GCN(X, A) \tag{2}
\]

where \( GCN(\cdot) \) is a graph convolutional operation. Specifically, \( W^{(l)}_{ge} \) is weighted matrix of layer \( l \) of GCN auto-encoder. Adjacency matrix \( A \) of the original data and the feature matrix \( X \) as model input. \( GCN \) module generates the representations of the specified dimensional, and the output can be finally found through the iterative process \( Z \in \mathbb{R}^{N \times f} \) where \( f \) is the dimensional of the expected representation. After the high-dimensional feature representations is obtained, the adjacency matrix of the original data is reconstructed on this model, and objective function is constructed to evaluate the efficiency of graph embedding.

The probability that two nodes are connected can be predicted by \( p(\hat{A}|Z) \). In other words, we train a connection prediction layer based on graph embedding:

\[
p(\hat{A}|Z) = \prod_{i=1}^{n} \prod_{j=1}^{n} p(\hat{A}_{ij}|z_i, z_j) \tag{3}
\]

\[
p(\hat{A}_{ij} = 1|z_i, z_j) = \text{sigmoid}(z_i^T \cdot z_j). \tag{4}
\]

Here, original data is treated as an undirected graph, and adjacency matrix as a symmetric matrix. Therefore, efficient inner product is used as the decoder. This can be illustrated by the following formula:

\[
\hat{A} = \sigma(ZZ^T). \tag{5}
\]
The adjacency matrix determines the structure of the graph. We set \( Y = q(Z | X, A) \), \( \hat{Y} = p(\hat{A} | Z) \), both \( Y \) and \( \hat{Y} \) are adjacency matrix. We expect that the reconstructed adjacency matrix is similar to the original adjacency matrix, so the objective function is defined as follows:

\[
E(i, j) = -[(Y_{ij} \cdot \log(\hat{Y}_{ij})) + (1 - Y_{ij}) \cdot \log(1 - \hat{Y}_{ij})]
\]

\[
L_{\text{adj}} = \sum_{i=1}^{n} \sum_{j=1}^{n} E(i, j)
\]

where \( q(Z | (X, A)) \) represents the adjacency matrix of the input graph data, \( n \) is the number of nodes in dataset, and \( E(i, j) \) represents the cross entropy loss function.

**B. DNN MODEL FOR AUTO-ENCODER**

It is not sufficient to perform clustering task if topology information is only embedded. We propose DNN auto-encoder to learn the high-dimensional representations of each node and weight it to participate in the backbone graph convolution module for iteration. There are several unsupervised learning methods to embed feature information in high dimensions. The dependency relationship between nodes is ignored temporarily, and only the features of nodes are considered.

DNN auto-encoder efficiently learns feature information, which can be abstractly represented as a high-dimensional feature. First, each node feature is split to generate the feature matrix \( X_f \). This feature matrix is calculated as the input of the fully connected neural network, so that we can obtain high-dimensional embedded \( \hat{X}_f \) for the original data. After obtaining the high-dimensional embedding, fully connected network decoder it, and finally calculate the decoding result of \( \hat{X}_f \). At this point, network structure of the coder and decoder layers should be perfectly symmetrical. Therefore, final generated \( \hat{X}_f \) and \( \hat{X}_f \) should have the same dimensionality and their values should be as similar as possible. The above process can be represented by the following equation:

\[
X_e = E(\phi(WX_f + b))
\]

\[
\hat{X}_f = D(\phi(WX_e + b))
\]

\[
L_{\text{res}} = \frac{1}{2N} \sum_{i=1}^{N} ||x_i - \hat{x}_i||_2^2 = \frac{1}{2N} ||X_f - \hat{X}_f||_2^2
\]

where \( E(\cdot) \) represents the encoder of the multi-layer fully connected neural network, \( D(\cdot) \) represents the decoder of the multi-layer fully connected neural network, \( \phi(\cdot) \) represents the nonlinear activation function, such as the ReLU(\cdot), \( W^{(l)} \) and \( \hat{W}^{(l)} \) are the weighted matrix of layer \( l \) of DNN encoder and decoder. \( b_{de}^{(l)} \) and \( b_{ad}^{(l)} \) are the bias matrix of layer \( l \) of DNN encoder and decoder.

**C. DenseNet**

DenseNet [30] is a deep layers method of convolutional neural network (CNN), which uses close link between layers to improve information flow in the network, allowing the result to be reused between each layer. We apply a similar idea to GCN and specifically construct dense links from each GCN layer to exploit information at each layer. This method is applied to the graph clustering model. DenseGCN structure can be referred to Fig. 2. The framework of our model can be expressed by the following formula:

\[
G_{l+1} = \mathcal{H}(G_l, W_l)
\]

\[
= T(\mathcal{F}(G_l, W_l), G_l)
\]

\[
= T(\mathcal{F}(G_l, W_l), \ldots, \mathcal{F}(G_0, W_0), G_0)
\]

where \( \mathcal{H} \) is DenseGCN convolution operation, which can be divided into \( \mathcal{F} \) and \( T \). \( \mathcal{F} \) is the GCN layer function shown in Eq.1 and \( W^{(l)} \) is weighted matrix of layer \( l \) of backbone GCN network. \( T \) is a vertex cascading function, which intensively fuses the input graph \( G_0 \) with all intermediate GCN layers, so \( G_{l+1} \) contains the results of all GCN output from the previous layer. For example, if \( \mathcal{F} \) produces a \( D \) dimension feature matrix, the dimension of input graph \( G_0 \) is \( D_0 \), and the dimension of each node feature in \( G_{l+1} \) is \( D_0 + D \times (l + 1) \). DenseGCN and GCN comprise our backbone network.

Obviously, \( G_l = (V_l, E_l) \) represents the information of the data itself at the GCN of \( l \) layer. \( G_l \) contains node information \( V_l \) and edge information \( E_l \). The specific representations of node information is a feature matrix. In our proposed model, the feature matrix of the data itself is expressed as \( H^{(l)} \). Among them, \( V_l \) represents the feature information of the data itself, \( E_l \) represents the dependency formula between data. We evaluate the results learned from DNN auto-encoder and GCN auto-encoder, which are applied to GCN layers and DenseGCN layer to perform clustering tasks. The specific method can be expressed as:

\[
V_l = H^{(l)}
\]

\[
H^{(l)} = \alpha H^{(l)}_G + \beta H^{(l)}_f + \gamma H^{(l)}_{\text{adj}}
\]
the learning result of GCN auto-encoder. \( H^{(l)}_{\text{gcn}} \) represents the natural iteration result of the backbone GCN layer. \( \alpha, \beta, \) and \( \gamma \) represents weight coefficient of each variable, which definition domain are [0, 1]. Specifically, the content of iterative learning in backbone network is obtained by weighting learning results of two auto-encoders and original input. For classical GCN layers, the convolution formula is Eq.1. The inspiration of DenseGCN originated from close connection between layers. When computing the convolution of the graph, the output of previous layers should be considered significantly. In order to increase the content of information flow to construct deep GCN layers, \( H^{(l+1)} \) stands for the result of DenseGCN, which can be computed as:

\[
H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\eta(H^{(l)}, H^{(l-1)}, \ldots, H^{(0)})W^{(l)})
\]

(14)

where \( \eta(.) \) represents the splice function of the output vector in the previous layer, \( l \) represents \( l \)th layer of the backbone network. In our model, \( H^{(l+1)} \) represents clustering dimension results.

Clustering distribution result \( Z \) of GCN module in the backbone network can be defined by the following formula:

\[
Z = \text{softmax}(H^{(l)}).
\]

(15)

where \( H^{(l)} \) represents the output of the last layer of backbone GCN network.

**D. TRIPLE SELF-SUPERVISED MODEL**

Through two auto-encoders, our model obtains the data representations of feature and topology information of graph. However, the DNN model and GCN model are not suitable for unsupervised training. Considering the characteristics of graph clustering task and the proposed model, we propose a triple self-supervised module based on optimizing the distribution of clustering results. The triple self-supervised module calculates the KL divergence of the clustering result distribution and the target distribution to form a part of the loss function. This module combines the backbone GCN network, DNN auto-encoder, and GCN auto-encoder in a unified module to realize the end-to-end self-supervised training of the model.

The clustering result is generated from the learning results of two auto-encoders, and then the similarity between the clustering result and the cluster center is calculated. Specifically, for the \( i \)th sample and the \( j \)th cluster, we use Student’s t distribution [28] as the kernel to measure the similarity between the data representations and the clustering centers. We use \( Q = \{q_{ij}\} \) as the target distribution. After obtaining the clustering distribution \( Q \), we aim to improve cluster purity and normalize loss contribution of each centroid to prevent large clusters from distorting the hidden feature space. Therefore, the clustering result distribution \( Q \) and the target distribution \( P \) is calculated as follows:

\[
p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j} q_{ij}^2/f_j}
\]

(17)

where \( h_i^f \) is the output of the \( i \)th layer of the DNN auto-encoder, \( h_i^{adj} \) is the output of the \( i \)th layer of the GCN auto-encoder, \( \mu_j \) is initialized by K-means learned by the auto-encoder before training. \( v \) is the degree of freedom of Student’s t distribution [28], \( q_{ij} \) is regarded as the probability of assigning the sample \( i \) to the cluster \( j \). \( \epsilon \) represents the weight representations of the final result distribution of the two auto-encoders. \( f_j = \sum_{i} q_{ij} \) is the soft cluster frequency. In the target distribution \( P \), the square sum of each distribution in \( Q \) is normalized. In this way, the spatial feature information of the graph structure and the features of the data itself are considered. In order to make the distribution with higher confidence, we can derive our objective function from the following equation:

\[
L_{clus} = KL(P||Q) = \sum_{i} \sum_{j} p_{ij}\log \frac{p_{ij}}{q_{ij}}
\]

(18)

Here the target \( P \) distribution is obtained. Next, we use the \( P \) distribution to train the cluster distribution \( Z \) output by the backbone network. Then objective function can be formulated as follows:

\[
L_{gcn} = KL(P||Z) = \sum_{i} \sum_{j} p_{ij}\log \frac{p_{ij}}{z_{ij}}
\]

(19)

Now the result distribution of the backbone network and the learning results of the two auto-encoders constitute our objective function. Since our \( P \) distribution was initially obtained by evaluating the results of DNN and GCN auto-encoder modules respectively, DNN and GCN modules benefit from the target distribution \( P \) in low-dimensional feature representations by minimizing the KL difference loss. \( P \) is used to train \( Z \) generated by the backbone GCN network. Objective function is computed as follows:

\[
L = L_{res} + w_1 L_{clus} + w_2 L_{gcn} + w_3 L_{adj}
\]

(20)

where \( w_1 \in [0, 1] \) represents weight ratio of the KL divergence of the clustering result distribution \( P \) and the target distribution \( Q \). \( P \) contains the learning results of GCN auto-encoder and DNN auto-encoder. \( w_2 \in [0, 1] \) represents the weight ratio of the KL divergence of the target distribution \( P \) and the distribution of clustering results \( Z \) of backbone GCN output. \( w_3 \in [0, 1] \), unless the data itself comes from an indirect graph dataset \( w_3 = 0 \), because graph structured data generated by KNN [18] can not describe the true dependency. \( L_{adj} \) represents the learning effect of the GCN auto-encoder. This means the difference between the spatial structure of the original data after reconstruction and the original data. We call it triple self-supervised module.

**E. ALGORITHM OVERVIEW AND COMPLEXITY ANALYSIS**

Algorithm 1 summarizes DGSCL workflow.

We set \( d \) as the dimension of the input data, and assume data dimension of each layer in auto-encoder
The time complexity of the clustering task is concluded by

\[ T(\text{task}) = \mathcal{O}(d_1, d_2, \ldots, d_l, N) \]

where \( d_1, d_2, \ldots, d_l \) and \( N \) are the number of clustering categories. For the GCN auto-encoder module, the time complexity is \( \mathcal{O}(N d_1^2 \ldots d_l^2) \). For the GCN auto-encoder module, sparse matrix operation is used, time complexity is \( \mathcal{O}(|\mathcal{E}|d_1d_2 \ldots d_l) \). For the GCN module of the backbone network, we use a deep layer strategy DenseGCN, and sparse matrix operation. Time complexity is linearly related to the number of edges \( |\mathcal{E}| \) and the cumulative number of DenseGCN layers \( n \). Therefore, time complexity of GCN module of the backbone network is \( \mathcal{O}(|\mathcal{E}| d_1d_2 \ldots d_l) \).

Finally, considering the complexity of clustering task itself, we assume that the number of clusters is \( K \) in our final task. According to the result in [31], time complexity of the clustering task is concluded by \( \mathcal{O}(NK + N \log N) \). Time complexity of our model is \( \mathcal{O}(N d_1^2 \ldots d_l^2 + |\mathcal{E}| d_1d_2 \ldots d_l + NK + N \log N) \). The number of edges increases linearly, which means computational cost of GCN is related to the size of the dataset and the cumulative number of GCN layers. Although deep GCN brings extra computational cost, we have simplified the DenseGCN module of the backbone network, assuming that the number of clusters is \( K \).

### IV. EXPERIMENTS

#### A. DATASETS

Our proposed model (DGSCN) is experimented on five datasets in Table 2.

1. Wiki [32]: It is a webpage network where nodes are webpages and are connected if one links the other. It is a typical dataset with a small number of nodes, but a abundant spatial topology relationship between nodes.

2. Reuters [33]: Reuters is a text dataset, which contains approximately 810,000 English news stories marked with category trees. We use 4 root categories as labels and cluster from a random subset of 10,000 examples.

3. Citeseer: This dataset contains the sparse word bag feature vectors of each document and a list of citation links between the documents. The access to the dataset: http://citeseerx.ist.psu.edu/index.

4. DBLP: DBLP is an integrated database system with the author as the core of research results in the field of computer science. We mainly focus on the cooperation among the authors. If two authors are co-authors, there is an edge relationship between them, and the authors are divided into four areas. This dataset marks each author’s research area based on the meeting they submitted. The access to the dataset: https://dblp.uni-trier.de.

5. ACM: ACM is a publication relationship dataset. If two papers are written by the same author, then there is an advantage between them. Keywords are featured in ACM. We choose papers published on KDD, SIGMOD, SIGCOMM and MobiCOMM, and divide them into three categories according to research fields. The access to the dataset: http://dl.acm.org/.

#### B. BASELINES

We experimented with our proposed model and nine other models. The details are as follows:

1. K-means [34]: K-means is a classical machine learning method, which randomly selects K seed points, works out the distance between all points and the seed point, and puts the points into the nearest seed point group. After all the points

| Dataset | Nodes | Edges | Features | Classes |
|---------|-------|-------|----------|---------|
| Wiki    | 2405  | 17981 | 4973     | 17      |
| Reuters | 10000 | 15003 | 2000     | 4       |
| Citeseer| 3327  | 4732  | 3703     | 6       |
| DBLP    | 4058  | 3528  | 334      | 4       |
| ACM     | 3025  | 13128 | 1870     | 3       |

To research clustering task further, we implement experiments with Wiki and ACM. These two datasets have few nodes but abundant spatial topology information.

### Algorithm 1 DGSCN

**Require:** Feature matrix of graph \( X \); Adjacency matrix of graph \( A \); Number of clusters \( K \); Number of iteration \( N \);

**Ensure:** Clustering results \( R \);

1. Initialize \( W_{ge}^{(l)} \) with GCN auto-encoder; \( W_{de}^{(l)}, b_{de}^{(l)}, W_{dd}^{(l)}, b_{dd}^{(l)} \) with pre-train DNN auto-encoder; \( W^{(l)} \) with GCN backbone network;

2. Initialize \( (\mu_1, \mu_2, \ldots, \mu_k) \) with K-means on representations learned by pre-train auto-encoder;

3. **for** \( ite = 0, 1, \ldots, N \) **do**

   - **for** \( ie = 0, 1, \ldots, l \) **do**

   1. Calculate DNN auto-encoder result \( H_{d}^{(l)} \);
   2. Calculate GCN auto-encoder result \( H_{adj}^{(l)} \);
   3. if \( l \neq (l - 1) \) then
      1. Generate GCN network result \( H_{G}^{(l)} \) via Eq.1;
   4. else
      1. Calculate bGCN network result \( H_{G}^{(l)} \) via Eq.14;
   5. end if
   6. Use \( \alpha = 1, \beta = 1, \gamma = 1 \) to calculate \( H_{d}^{(l)} \) via Eq.13;
   7. End for
   8. Use \( \epsilon = 0.1, v = 1, H_f^{(l)}, H_{adj}^{(l)} \);
   9. Calculate soft assignment distribution \( Q \) via Eq.16;
   10. Calculate target distribution \( P \) via Eq.17;
   11. Calculate predicted result \( Z \) via Eq.15;
   12. Generate GCN reconstruction result via Eq.5;
   13. Generate DNN reconstruction result via Eq.9;
   14. Use \( w_1 = 0.1, w_2 = 0.01, w_3 = 1 \) to generate loss function via Eq.20;
   15. Back propagation and update parameters \( W_{ge}^{(l)}, W_{de}^{(l)}, W_{dd}^{(l)}, b_{de}^{(l)}, b_{dd}^{(l)}, W^{(l)} \) in DGSCN;
   16. End for
   17. Calculate the clustering results \( R \) based on distribution \( Z \);

24: return \( R \);
are contained in the group, moving the seed point to the center of the group and iterating the above steps until the seed point does not move.

2) DeepWalk [35]: This model is divided into two parts: random walk and feature vector generation. Firstly, random walk algorithm extracts some vertex sequence from the graph. Secondly, natural language processing tool word2vec represents each vertex. Deepwalk is a graph embedding method.

3) AE [36]: Auto-Encoder has two parts: encoder and decoder. The node feature of original data is obtained through the encoder, and the original data is restored by decoder. Then, K-means clustering method is applied to cluster the learned high-dimensional feature and obtain the result. This approach completely ignores topology information.

4) IDEC [6]: IDEC jointly optimize the cluster label allocation and learn functions suitable for clustering using local structure to improve auto-encoder learn performance.

5) GAE [3]: This is an unsupervised graph embedding method that adopts GCN to learn data representations, which performs tasks by clustering high-dimensional feature. GAE only considers the reconstruction of graph topology information.

6) ARGE&ARVGE [4]: Based on the classical graph clustering algorithm GAE, an adversarial generation training mechanism is proposed. The clustering result generated by GAE is used as the input of generative adversarial network (GAN) to learn the distribution of clustering results. Clustering is accomplished by combining the two modules. The algorithm fails to fully consider the graph structure information.

7) DAEGC [24]: It uses attention networks to learn node representations and clustering loss to supervise graph clustering. Different from GCN, it adopts graph attention mechanism to replace the graph convolution operation for iteration to conduct the clustering task. DAEGC only considers the reconstruction of graph topology.

8) SDCN [25]: It uses GCN and DNN auto-encoders to jointly construct a clustering network. And the self-supervised module is proposed to update the parameters of graph convolution module and DNN auto-encoder module. There are some limitations in the processing of topology information.

9) AGC [5]: It is a frequency-domain method in the process of graph convolution to make the convolution more suitable for performing clustering tasks. AGC method performs well on a single dataset but lacks the generalization ability.

10) DGSCN: Our proposed method.

C. EVALUATION INDICATORS

We adopt four popular clustering evaluation indicators. Accuracy (ACC) represents the accuracy of the clustering result. Normalized Mutual Information (NMI) is the amount of correct clustering information contained in our clustering results. Average Rand Index (ARI) means the evaluation of clustering results by calculating the similarity between two clusters, which is an improvement based on Rand Index (RI). Macro F1-score (F1) represents the evaluation index obtained by weighting the accuracy rate and recalling rate of clustering results. For each evaluation indicator, the higher the evaluation value is, the better the clustering task performance is.

D. PARAMETER SETTINGS

For fair comparison, all the methods are conducted on the same hardware platform of GTX 2080Ti GPU. For DNN-based (AE + K-means, DEC, IDEC, SDCN) and GCN-based clustering methods, a pre-train auto-encoder is adopted. We perform all data points with 30 epochs to train the DNN auto-encoder end-to-end with the learning rate is $10^{-3}$. In order to be consistent with the previous method, the size of the auto-encoder is $d$-500-500-2000-10, where $d$ is the size of input data. These numbers represent the number of layers chosen by the previous method in turn.

For GCN-based methods, the dimension of GAE and VAGE is d-256-16, and train 30 epochs for all datasets. For DAEGC, the settings of [27] is used. For SDCN method, the hyperparameter settings proposed in [24] are adopted, and the best result is listed. For AGC, it uses adaptive graph convolution iteration times, so there is no need to set it. And the rest of the parameter settings can refer to [5].

The parameters of our model DGSCN are set as follows: Since the priori knowledge and spatial structure is more significant, we choose the form with the objective function. For the structural data of KNN graph, spatial structure feature is insufficient, so this item in the loss function is ignored. For the five datasets, the layer number $n$ of DenseGCN is 36. For the weight of the loss function, we uniformly set it to $w_1 = 0.1$, $w_2 = 0.01$, and $w_3 = 1$ in the experiment. The weight proportions of the feature vectors participating in the backbone network iteration are $\alpha = 1$, $\beta = 1$, and $\gamma = 1$. The weight strategy we used when generating $P$ distribution of clustering results is $\epsilon = 0.1$, and for all experiments, in Eq.16 let $\nu = 1$. K-means algorithm is used to generate clustering results of all models.

E. ANALYSIS OF CLUSTERING RESULTS

For Reuters, we obtain the graph data type representations through the KNN method [18]. The remaining four datasets Wiki, ACM, DBLP, and Citeseer are original graphical representation of the dataset. For specific experimental data, please refer to Table 3 and Table 4. In order to indicate the model stability, we list The average clustering performance with standard deviations of the proposed method. The standard deviations of Acc, NMI, and ARI F1 are 0.31%, 0.25%, 0.37%, 0.19% on Citeseer, 0.21%, 0.23%, 0.32%, 0.15% on Reuters, 0.67%, 0.72%, 0.93, 0.45% on DBLP, 0.51%, 0.29%, 0.67%, 0.15% on Wiki, 0.11%, 0.16%, 0.22%, 0.09% on ACM, respectively.

From Table 3 we draw two conclusions:

1) Our method has obvious advantage when the datasets itself has an original adjacency matrix representations. Owing to full consideration of the feature and topology information from the original graph, superior baseline results are obtained.
from DBLP and Citeseer. The reconstruction of the graph dependency is added to the self-supervised module for learning, and then two auto-encoder modules are adopted to update the parameters.

(2) If dataset does not have the original graph structure, KNN [18] is used to generate the graph structure data. However, the adjacency matrix of graph constructed by the KNN method is not reliable. Because it does not fully reflect the dependencies of data itself, if the topology information is generated by KNN, our model may not work well. And in the backbone network module, we have deep GCN layers. This is equivalent to magnifying the influence of topology information weight to a certain extent, which makes our model unable to highlight its advantages.

One conclusion drawn by Table 4: If topology information between nodes is abundant, our model shows obvious superiority because topology enhances the representations. Topology learning results are added to the backbone GCN layer for iteration. We have achieved excellent experimental results on Wiki dataset.

We fully consider the weight of data itself and the dependencies represented by adjacency matrix, so our approach is superior to the other methods only considering the graph structure of clustering such as GAE, VEGA, DAEGC. Because the auto-encoder module of graph reconstruction is added, our method is also superior to the method SDCN with only data features. Furthermore, compared with the spectral method AGC, our model has wider applications.

F. ANALYSIS OF DenseGCN LAYERS

Novel deep GCN layers are proposed in the model. The result of deep GCN layers on the overall clustering task is analyzed as follows. In order to show our experimental results in details, we choose four different results from experimental datasets.

We fix the other parameters as the weight of loss function, which are set to $w_1 = 0.1$, $w_2 = 0.01$, $w_3 = 1$. The weight parameters of the feature in iteration are $\alpha = 1$, $\beta = 1$, and $\gamma = 1$. The weight when generating the $P$ distribution of the clustering result is $\epsilon = 0.1$. From our experimental result Fig. 3, we can draw the following conclusions:

(1) According to our experimental results, the number of layers increases, the overall trend of clustering accuracy first increases and then decreases after the peak. For different datasets, the information content determines the peak accuracy of clustering task. However, we can conclude that the deep GCN layers has a positive result on the accuracy of clustering tasks. We need to adjust the most appropriate number of GCN layers based on the dataset.

(2) Curve analysis of four datasets reveals that the dataset with the smallest dimension of node features is DBLP.
FIGURE 3. Results of the difference of deep GCN layers $n$ on clustering accuracy. The x-axis represents the number of convolutional layers, and y-axis represents the accuracy of clustering tasks.

DBLP curve reflects that it peaks at the earliest epoch, as the number of layers increasing clustering accuracy declines due to over-smoothing. Reuters is the largest dataset in our experiments, which contains rich feature information. Reuters curve reflects that the strategy of deep layers is more suitable for the datasets with large information content. For smaller datasets, clustering performance will decrease. Over-smoothing usually leads to the infeasibility of stacking simple GCN layers, so we propose DenseGCN to alleviate the influence of over-smoothing.

(3) Deep GCN layers improve clustering performance, but corresponding strategies should be formulated according to the feature of the dataset itself. However, for the clustering tasks that have been performed well, the improvement of intuitive accuracy calculated by deep GCN is not obvious.

G. ANALYSIS OF WEIGHTS BASED ON GCN AUTO-ENCODER

In order to exploit overall information of graph data, we add the learning results of graph reconstruction to the backbone network. In our experiment, $\gamma = 0$ means that only the DNN auto-encoder module performs weight operation. The results of our experiment can be referred to Fig. 4, and the conclusions can be drawn as follows:

(1) Since we use deep GCN layers strategy, our experimental results are superior to SDCN [25] when $\gamma = 0$. Results indicate that when the value of $\gamma$ is small, the graph reconstruction feature is insufficient. Moreover, objective function is considered in the loss of graph reconstruction, so the performance will be worse. As the weight coefficient is increasing, the result tends to be stable.

(2) When our model is applied to datasets with superior average performance, weight method of graph reconstruction has no obvious superiority, especially for ACM dataset.

(3) When the dataset is complicated, such as Wiki dataset, especially when the dataset has abundant topology information, our model can perform clustering tasks more efficiently, the specific result can be referred to Fig. 5.

FIGURE 4. Influence of different feature weights $\gamma$ on clustering accuracy. The x-axis represents the weight proportion of GCN auto-encoder, and the y-axis represents the accuracy of clustering tasks.

H. ANALYSIS OF OVER-SMOOTHING

Over-smoothing is one of the most important challenge in graph convolutional networks (GCN). If over-smoothing problem is not considered, it will lead to the increase of training iteration. Furthermore, accuracy rate of clustering and related evaluation indexes decline. Therefore, the deep GCN layers in our proposed model not only perform the clustering task efficiently, but also consider whether over-smoothing can be alleviated. Here we compare our proposed method DGSCN with SDCN [25] and GAE [3] in Citeseer and Wiki dataset. The results are as follows in Fig. 6. Iteration number is 200, we choose 150 iterations after the model reaches the peak. Conclusions are as follows:

(1) Because our model is more complicated and has more parameters, SDCN and GAE costs fewer training times to achieve peak performance. However, after reaching the peak,
our model significantly alleviates the over-smoothing influence, and the clustering accuracy only slightly decreases. (2) SDCN and GAE method has fewer model parameters, which can achieve the best effect faster. However, as the number of training increases, instability of the model leads to the performance fluctuation. Furthermore, the clustering accuracy of the whole model shows a downward trend and reaches the unstable status.

(3) For the Wiki dataset, which has 17 clustering categories. The problem of over-smoothing does not have much impact on the bad performance methods, because the data features cannot be learned effectively and the abundant categories influence it.

V. CONCLUSION

In this paper, we propose a deep graph structured clustering network (DGSCN) for processing graph structured data with deep clustering method. This model considers both graph node feature and topology information to improve model performance. Our model has novel deep GCN layers in clustering tasks, and a triple self-supervised module is applied to perform end-to-end unsupervised training. Experimental results demonstrate that our method achieves superior performance over state-of-the-art models.

Our proposed method also has some limitations. Deep layers mean more computational cost, we are committed to finding more efficient deep GCN methods. And different datasets own different graph structure. Many datasets own less nodes feature but more relation feature. We hope to find an adaptive processing method, which is one of our future research focuses.

REFERENCES

[1] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907. [Online]. Available: http://arxiv.org/abs/1609.02907

[2] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: http://arxiv.org/abs/1312.6114

[3] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, arXiv:1611.07308. [Online]. Available: http://arxiv.org/abs/1611.07308

[4] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized graph autoencoder for graph embedding,” 2018, arXiv:1802.04407. [Online]. Available: http://arxiv.org/abs/1802.04407

[5] X. Zhang, H. Liu, Q. Li, and X.-M. Wu, “Attributed graph clustering via adaptive graph convolution,” 2019, arXiv:1906.01210. [Online]. Available: http://arxiv.org/abs/1906.01210

[6] X. Guo, L. Gao, X. Liu, and J. Yin, “Improved deep embedded clustering with local structure preservation,” in Proc. 26th Int. Joint Conf. Artif. Intell., Aug. 2017, pp. 1753–1759.

[7] C. Yan, H. Xie, J. Chen, Z. Zha, X. Hao, Y. Zhang, and Q. Dai, “A fast uyghur text detector for complex background images,” IEEE Trans. Multimedia, vol. 20, no. 12, pp. 3389–3398, Dec. 2018.

[8] C. Yan, L. Li, C. Zhang, B. Liu, Y. Zhang, and Q. Dai, “Cross-modality bridging and knowledge transferring for image understanding,” IEEE Trans. Multimedia, vol. 21, no. 10, pp. 2675–2685, Oct. 2019.

[9] C. Yan, Y. Tu, X. Wang, Y. Zhang, X. Hao, Y. Zhang, and Q. Dai, “Corrections to ‘STAT: Spatial-temporal attention mechanism for video captioning,’” IEEE Trans. Multimedia, vol. 22, no. 3, p. 830, Mar. 2020.

[10] C. Yan, B. Shao, H. Zhao, R. Ning, Y. Zhang, and F. Xu, “3D room layout estimation from a single RGB image,” IEEE Trans. Multimedia, early access, Jun. 17, 2020, doi: 10.1109/TMM.2020.2967645.

[11] C. Yan, B. Gong, Y. Wei, and Y. Gao, “Deep multi-view enhancement hashing for image retrieval,” IEEE Trans. PAMI, vol. 42, no. 8, pp. 1592–1604.

[12] C. Li and D. Goldwasser, “Encoding social information with graph convolutional networks for political perspective detection in news media,” in Proc. 57th Ann. Meeting Assoc. Comput. Linguistics, 2019, pp. 2594–2604.

[13] P. Han, P. Yang, P. Zhao, S. Shang, Y. Liu, J. Zhou, X. Gao, and P. Kalnis, “GCN-MF: Disease-gene association identification by graph convolutional networks and matrix factorization,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 705–713.

[14] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 132–149.

[15] C. Li and D. Goldwasser, “Encoding social information with graph convolutional networks and matrix factorization,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 705–713.

[16] A. Bojchevska, O. Shchur, D. Žügner, and S. Günnemann, “NetGAN: Generating graphs via random walks,” 2018, arXiv:1803.00816. [Online]. Available: http://arxiv.org/abs/1803.00816

[17] Y. Shang, “Generalized K-Core percolation in networks with community structure,” SIAM J. Appl. Math., vol. 80, no. 3, pp. 1272–1289, Jun. 2020.

[18] Y. Cai, L. Ge, J. Liu, J. Cai, T.-J. Cham, J. Yuan, and N. M. Thalmann, “Exploiting spatial-temporal relationships for 3D pose estimation via graph convolutional networks,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 2272–2281.

[19] Y. Shang, “Generalized K-Core percolation in networks with community structure,” SIAM J. Appl. Math., vol. 80, no. 3, pp. 1272–1289, Jun. 2020.

[20] Y. Cai, L. Ge, J. Liu, J. Cai, T.-J. Cham, J. Yuan, and N. M. Thalmann, “Exploiting spatial-temporal relationships for 3D pose estimation via graph convolutional networks,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 2272–2281.

[21] P. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, “Towards K-means-like spaces: Simultaneous deep learning and clustering,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 3861–3870.

[22] T. He, L. Bai, and Y.-S. Ong, “Manifold regularized stochastic block model,” in Proc. IEEE 31st Int. Conf. Tools with Artif. Intell. (ICTAI), Nov. 2019, pp. 800–807.

[23] T. He and K. C. C. Chan, “Discovering fuzzy structural patterns for graph analytics,” IEEE Trans. Fuzzy Syst., vol. 26, no. 5, pp. 2785–2796, Oct. 2018.

[24] T. He and K. C. C. Chan, “Discovering fuzzy structural patterns for graph analytics,” IEEE Trans. Fuzzy Syst., vol. 26, no. 5, pp. 2785–2796, Oct. 2018.

[25] T. Guo, S. Pan, X. Zhu, and C. Zhang, “CFOND: Consensus factorization for co-clustering networked data,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 4, pp. 706–719, Apr. 2019.

[26] L. Liu, L. X. Z. Wang, and E. Chen, “Community detection based on structure and content: A content propagation perspective,” in Proc. IEEE Int. Conf. Data Mining, Nov. 2015, pp. 271–280.

[27] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, “Attributed graph clustering: A deep attentional embedding approach,” 2019, arXiv:1906.06532. [Online]. Available: http://arxiv.org/abs/1906.06532

[28] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, “Structural deep clustering network,” in Proc. Web Conf., Apr. 2020, pp. 1400–1410.

[29] G. Li, M. Muller, A. Thabet, and B. Ghanem, “Deeppgns: Can GCNS go as deep as CNNs?” in Proc. IEEE Int. Conf. Comput. Vis., Oct. 2019, pp. 9267–9276.

[30] Y. Rong, W. Huang, T. Xu, and J. Huang, “DropEdge: Towards deep graph convolutional networks on node classification,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2020. [Online]. Available: https://openreview.net/forum?id=Hkx1qkrKPr
X. Li et al.: Deep Graph Structured Clustering Network

[28] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, Nov. 2008.

[29] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, “Community preserving network embedding,” in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 203–209.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2261–2269.

[31] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in Proc. Int. Conf. Mach. Learn., vol. 48, 2015, pp. 478–487.

[32] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, “Network representation learning with rich text information,” in Proc. 24th Int. Joint Conf. Artif. Intell., 2015, pp. 2111–2117.

[33] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, “RCV1: A new benchmark collection for text categorization research,” J. Mach. Learn. Res., vol. 5, pp. 361–397, Dec. 2004.

[34] J. A. Hartigan and M. A. Wong, “A K-means clustering algorithm,” Appl. Stat., vol. 28, no. 1, pp. 100–108, 1979.

[35] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2014, pp. 701–710.

[36] G. E. Hinton, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[37] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in Proc. 27th Int. Conf. Mach. Learn., 2010, pp. 807–814.

[38] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for Web-scale recommender systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 974–983.

[39] J. A. Hartigan and M. A. Wong, “A K-means clustering algorithm,” Appl. Stat., vol. 28, no. 1, pp. 100–108, 1979.

[40] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2014, pp. 701–710.

[41] G. E. Hinton, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[42] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in Proc. 27th Int. Conf. Mach. Learn., 2010, pp. 807–814.

[43] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for Web-scale recommender systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 974–983.

[44] J. A. Hartigan and M. A. Wong, “A K-means clustering algorithm,” Appl. Stat., vol. 28, no. 1, pp. 100–108, 1979.

[45] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2014, pp. 701–710.

[46] G. E. Hinton, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[47] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in Proc. 27th Int. Conf. Mach. Learn., 2010, pp. 807–814.

[48] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for Web-scale recommender systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 974–983.