Self-Supervised Graph Convolution Autoencoder for Social Networks Clustering

Hu Lu,Chao Chen,Haotian Hong,Hai Wang,Shaohua Wan

***Abstract*—In recent years, graph-based deep learning algorithms have attracted widespread attention. Still, most of the current graph neural networks are based on supervised learning or semi-supervised learning, which often relies on the actual labels given by the samples as auxiliary information, thus limiting the robustness and generalization ability of the model. To solve this problem, we propose a neural network model based on the self-supervised graph convolutional autoencoder structure and use it for social networks clustering. We divide the proposed model into two parts: a pretext task and a downstream task. In the pretext task, the model generates pseudo label by graph attention autoencoder structure, then adopts the proposed reliable sample filtering mechanism to gain a high confidential sample. This sample and corresponding pseudo label are selected as input of downstream task for helping downstream task to finish learning task. The model obtains the predictive labels of the sample from the downstream task. This model does not depend on the actual label sample for training and uses the pseudo labels produced by self-supervised learning to improve the clustering performance. We apply the proposed model to three commonly used public social network datasets to test its performance. Compared with the presented unsupervised clustering algorithms and other deep graph neural network algorithms, various metrics have shown that the proposed model consistently outperforms the state-of-the-art.**

***Index Terms*—Self-Supervised, Autoencoder, reliable sample selection, social networks clustering.**

I. Introduction

THE graph-based clustering algorithm divides the nodes into several subgraphs with no intersection according to the similarity between the features. It intends to enlarge the samples' similarities in different subgraphs and shrink that in the same subgraph. In recent years, graph-based Clustering tasks have been widely used in social media networks [1], citation networks [2] and brain network analysis [3], and so on. Social network data is a typical graph structure consisting of nodes and edges formed from nodes. The partitioning of a social network by a network composed of nodes and edges between nodes performs graph clustering. Graph clustering [4], [5] usually uses the feature similarity between samples to construct an adjacency matrix to construct a graph. The traditional graph clustering method mainly minimizes the similarity between subgraphs and subgraphs by dividing the vertices in the graph and finally completing the clustering task. Traditional graph clustering algorithms include spectral clustering [6], minimal graph cutting [7], and other methods. With the advent of the era of big data, more and more data sets have reached thousands or even tens of thousands. The traditional graph clustering algorithm has a large computation when dividing subgraphs. It is difficult to perform parallel operations (because the graph is constructed from the entire sample set and is a whole), so the performance under large data sets is often unsatisfactory. With the introduction of graph neural networks [8], [9], many related research works combine graph classification tasks with deep learning. These methods mine deep feature representations of samples through continuous optimization of neural networks, such as graph Convolutional networks [10] and graph attention networks [11]. To get rid of the model's dependence on the samples' actual labels, some unsupervised graph clustering algorithms have been proposed one after another. They reconstructed the adjacency matrix A through an autoencoder. They then cooperated with traditional clustering algorithms to realize the task of deep graph clustering, such as Variational Graph Auto-Encoders [12] and Deep Attentional Embedded Graph Clustering [13]. With the proposal of various deep graph clustering, the application of deep neural network in graph structure has been promoted. Similarly, some new methods combining deep autoencoders and graph convolutional networks have also appeared [13], [23], [25], [34], [35]. However, the current graph neural network still has some areas to be improved. For example, many unsupervised graph clustering methods directly use the graph neural network for feature learning and then directly perform clustering division and do not use the deep clustering of samples to obtain pseudo labels. In addition, in the process of graph convolution, the clustering results after deep feature extraction of samples are often greatly affected by noise data, and the performance is not satisfactory. Therefore, we propose a new self-supervised graph convolutional autoencoder clustering model, which fully uses the self-supervised information in data analysis. This model uses the pseudo-labels obtained by deep graph clustering to further act on the subsequent models. The reliable sample selection mechanism can help choose a sample and filter out noise data, increasing the clustering performance of the model.

The main contributions of this paper can be summarized as follows:

1) In this paper, we propose a novel self-supervised learning-based graph network partitioning framework for social networks clustering, which divides the deep graph clustering model into two parts, namely the pretext

H. Lu, C. Chen and H.T.Hong are with School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, China. (Email: luhu@ujs.edu.cn;) H. Wang is with School of Automotive and Traffic Engineering of Jiangsu University, Zhenjiang, 212013, China. (Email: wanghai1019@163.com) S.H. Wan is School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China. (Email: shaohua.wan@ieee.org)

task and the downstream task. In the pretext task, the pseudo-label is generated by unsupervised clustering and used to assist the downstream label prediction task. The framework combines graph convolutional neural networks with graph autoencoders. The whole model can still complete the task of deep clustering of data without actual labels of the sample.

2) We propose a new reliable sample selection mechanism, which can select high-confidence samples and their corresponding pseudo-labels from the clustered samples, thereby effectively reducing the impact of noisy data on the performance of the whole model.
3) The proposed method in this paper still belongs to unsupervised learning. To test the performance of our model with different evaluation criteria, extensive experiment results on three public benchmark datasets have demonstrated that our method is highly competitive and consistently outperforms the state-of-the-art ones with a preferable margin.

II. Related work

A. Graph Cut Clustering and Graph Convolutional Neural Networks

With the continuous expansion of social networks in recent years, people have tried to use graph structure to describe the relationship between things, and the graph structural data has increased. More and more scientific researchers have also focused on graph structure classification [14] or clustering [15]. Graph clustering aims to partition the graph to ensure that the sample similarity in different subgraphs is small after segmentation, and the similarity of the samples within the same subgraph is large. Spectral clustering is one of the more classic graph partitioning algorithms. The algorithm was first proposed by Hoffman et al. [16]. This algorithm transforms the problem of subgraph partitioning into the problem of finding the top K minimum eigenvalues of the Laplacian matrix and then uses K-means clustering on the eigenvectors. After that, Shi et al. proposed a normalization-based cut (N-cut) graph clustering algorithm by calculating the similarity between subgraphs and the sum within the subgraphs [17]. However, when these traditional graph clustering algorithms are faced with high-dimensional data or datasets with large sample sizes, the clustering effect still needs to be improved. With the emergence of deep graph neural networks, people began to analogize the convolution structure in neural networks to the graph structure to realize the convolution operation on the graph. In the graph convolutional neural network proposed by Kipf [10], the convolution computation method on the graph frequency domain and the spatial domain is presented. In 2018, Petar et al. [11] added the attention mechanism to the graph neural network. They adapted the adjacency matrix weight information of different nodes, realizing the local convolution operation. Many deep graph clustering algorithms use graph convolution for feature extraction and then cooperate with clustering algorithms to complete the final graph clustering task. The above graph neural network tasks have achieved good results on many public graph datasets through graph convolution. They still need to input the real labels of some samples as supervision information, which cannot achieve unsupervised effects.

B. Deep Clustering Algorithms

Deep clustering refers to an unsupervised learning algorithm realized by deep learning and clustering algorithm. Deep clustering has become a hot spot in unsupervised learning research in recent years. Xie et al. proposed a classical deep clustering algorithm named Deep Embedded Clustering (DEC) [18], which extracts sample features through a denoising stack autoencoder [19]. DEC utilizes KL divergence to calculate the distance between the features of the intermediate layer and target distribution, realizing an end-to-end deep clustering model. Guo et al. [20] proposed the Deep Convolution Embedded Clustering model based on the DEC model, which replaced the fully connected layer structure in the DEC with a convolutional layer. They obtained better clustering accuracy on the image datasets. The graph autoencoder (GAE) [12] is similar to the convolutional autoencoder, consisting of encoding and decoding layers. The difference is that the graph autoencoder calculates the loss by reconstructing the adjacency matrix. The decoder often adopts the feature matrix and its transposed matrix for the dot product operation. In 2019, Wang et al. [13] combined the idea of graph attention convolutional autoencoder [21] and embedded clustering [22] based on GAE and DEC models and proposed Deep attentional embedded graph clustering (DAEGC). The model achieves good clustering performance on the graph datasets. Another graph embedding cluster model is adversarially regularized graph auntocoder [23] proposed by Pan et al., which incorporates the adversarial idea in the generative adversarial networks [24] based on the graph autoencoder and then mines the prior distribution of the features of the sample embedding layer. In 2020, Zhang et al. [25] also proposed the Embedding Graph Auto-Encoder (EGAE) based on embedded graph clustering, which adopted a variant of GAE with dual decoders and K-means clustering algorithm to obtain the optimal inner product partitioning of the graph. Zhang et al. proposed a spectral embedding network (SENet) for attribute graph clustering [35]. Guo and Dai proposed a new variational graph autoencoder based on graph convolutional networks [34]. The model considers the joint generative model of graph structure and node attribute's enhanced impact on embedding output.

III. The proposed method

A. Motivation

Various existing deep graph clustering methods mainly rely on the feature learning ability of deep neural networks without the inherent supervised information of the data itself. In order to better utilize the self-supervised ability to learn the feature information of the graph structure, our proposed model takes the deep graph clustering task as our pretext task. In order to further utilize the pseudo-label information after clustering, we filter the clustered samples and select some samples with high confidence of pseudo-label to assist the downstream task for learning, and then obtain a better deep graph clustering model.

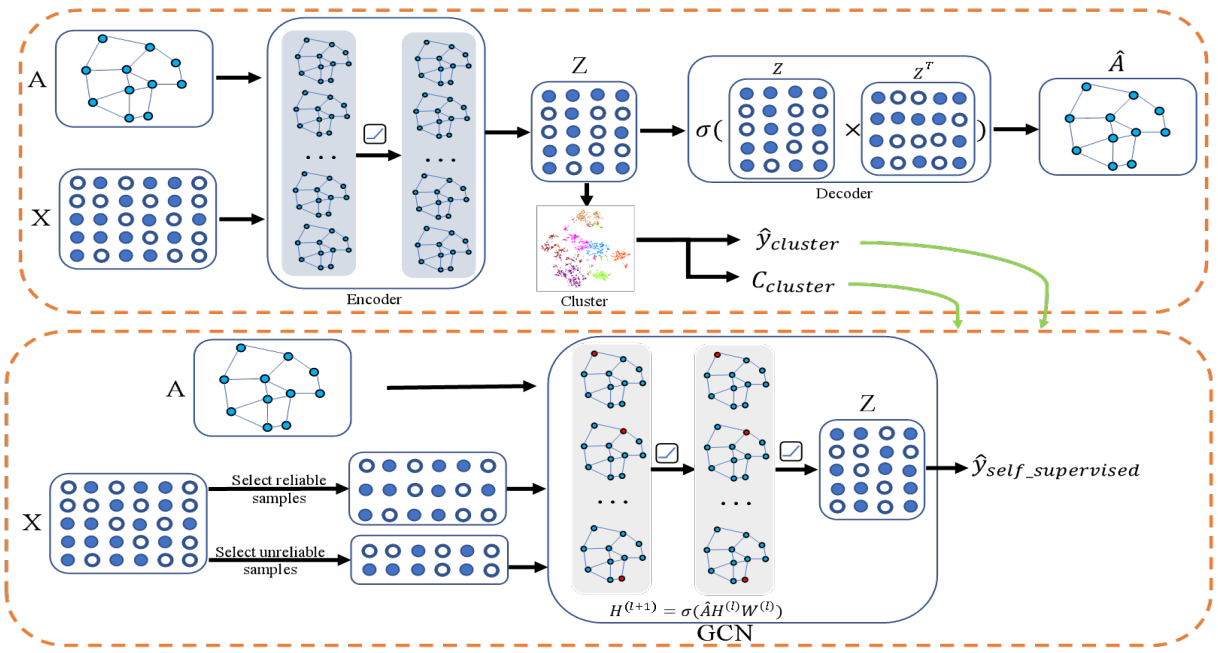


Fig. 1: The Conceptual illustration of Deep Self-Supervised Attentional Convolution Autoencoder Graph Clustering (DSAGC). The model is mainly divided into the pretext task and the downstream task. First, the sample feature information X and adjacency matrix A are used as input, which is passed to the graph attention autoencoder. The graph autoencoder is trained by reconstructing the matrix A and extracting each sample X's feature to obtain Z and clustering. Then, the samples are filtered according to the cluster centers and labels to assist the downstream tasks to perform self-supervised graph convolution training to obtain better clustering results.

B. Notations

A graph can be composed of three elements, i.e., $G = (V, E, X)$, where v_i denotes the i-th node in the graph, $v_i \in V$ and $|V| = n$. $e_{ij} = \langle v_i, v_j \rangle$ denotes edge between node i and node j, $e_{ij} \in E$. $X = \{x_0, x_1, \ldots, x_n\}$ denotes the feature of each node in the graph, where $X \in R^{n \times d}$, denotes the feature dimension of each sample is d. In graph convolution, it is also necessary to use edge information to obtain the adjacency matrix (adjacency matrix) A. In the topology of the graph, if there exists an edge between two nodes, that is $\langle v_i, v_j \rangle \in E$, then $A_{ij} = 1$, if $\langle v_i, v_j \rangle \notin E$, then $A_{ij} = 0$.

C. Overall Framework

The structural diagram of our proposed model is shown in Figure 1. The model is mainly divided into two parts. The first part is the graph attention autoencoder. The model uses the matrix A and features X as input to perform graph convolution to extract features to obtain intermediate layer features Z, as shown in Equation 1. Then features Z are used to reconstruct the matrix A. The reconstruction loss is used as the loss function during the model's training. Then, the encoded features Z are used for clustering and then obtain the cluster center and the corresponding pseudo-label of each sample.

$$\mathrm{Z} = f(\mathrm{X}, \mathrm{A}) \tag{1}$$

The second part of the model is the self-supervision module, which uses our proposed reliable sample select mechanism to screen samples. It transmits the pseudo-labels obtained from clustering as self-supervision information to downstream tasks for model training. In the self-supervised module, we mainly use the semi-supervised idea in the graph attention convolutional neural network to train the model, and the pseudo-labels of some samples are brought into the training process of the model so as to optimize the feature extraction ability of the model and improve the final clustering effect of the model.

D. Graph Attentional Autoencoder

In this paper, we use the graph attention autoencoder as our initial model. The specific model structure diagram is shown in the upper part of Figure 1. The autoencoder includes two layers of graph convolutional layers for extracting sample features. The graph convolution layer using the attention mechanism can assign corresponding weights to different neighboring nodes of each node without the need for complex calculations using matrices such as Laplacian and only requirements to use the feature of neighbor nodes to update the feature of the current node. We take the original feature of sample X and the adjacency matrix A as the input, denoted as $h^0 = \{x_0, x_1, \ldots, x_n\}$, after the graph convolution, the output feature h_i^l of the i-th node can be obtained by the weighted sum of neighbor nodes according to Equation 2.

$$h_i^l = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W h_j^{l-1}\right) \tag{2}$$

Where h_i^l denotes the output of the i-th node in the l-th layer, $\sigma(\cdot)$ denotes the activation function, and N_i represents the set of neighbor nodes of the node i, which can be obtained by using the adjacency matrix A. α_{ij} is the attention coefficient, which indicates the importance of the node j to the node i, i.e., the idea of the attention mechanism. W is the shared weights to be trained by the model, $W \in R^{d_{l-1} \times d_l}$. The weight matrix can map the features of nodes from d_{l-1} dimension to d_l dimension. In the forward propagation process of the graph attention neural network, the features of nodes i and j are multiplied by the weight matrix to obtain a two dimensional vector d_l. The two vectors are stitched to get a $2 \times d_l$ dimensional vector. Then, a real number is obtained by multiplying the stitched vector with a shared attention weight vector, which represents the attention coefficient. The specific calculation is as follows:

$$\varepsilon_{ij} = LeakyReLU\left(\vec{a}^T \left[Wh_i \parallel Wh_j\right]\right) \tag{3}$$

Where $\parallel$ denotes the concatenation operation of vectors. $LeakyReLU(\cdot)$ is the activation function. The final attention coefficient also needs to be normalized, i.e., for each node, the sum of the attention coefficients of its neighbors should be 1. Therefore, the attention coefficients can be expressed as:

$$\alpha_{ij} = softmax(\varepsilon_{ij}) = \frac{exp(\varepsilon_{ij})}{\sum_{k \in N_i} exp(\varepsilon_{ik})} \tag{4}$$

To represent the attention coefficients between nodes stably, we also introduce the multi-head attention mechanism to calculate the attention coefficients, i.e., by calculating T different shared weight W to obtain corresponding attention. We average the results of their computation of these attentions to obtain the final output vector. The specific calculation method is as follows:

$$h_i^l = \sigma\left(\frac{1}{T}\sum_{t=1}^{T}\sum_{j \in N_i} \alpha_{ij}^t W^t h_j^{l-1}\right) \tag{5}$$

The encoding layer of the graph attention autoencoder includes two layers of graph convolution layers. The node features are updated in the forward propagation process according to Equation 5. The difference is that the first graph convolution layer adopts a LeakReLU activation function, and the second layer adopts the softmax activation function, i.e.:

$$z_i = Softmax\left(\frac{1}{T}\sum_{t=1}^{T}\sum_{j \in N_i} \alpha_{ij}^t W^t ReLU\left(\frac{1}{T}\sum_{t=1}^{T}\sum_{j \in N_i} \alpha_{ij}^t W^t X\right)\right) \tag{6}$$

The decoding layer of the graph attention autoencoder reconstructs the original graph structure by calculating the inner product of the intermediate layer features Z. The decoding process can be represented by Equation 7.

$$\hat{A}_{ij} = sigmoid\left(z_i \cdot z_i^T\right) \tag{7}$$

This part of the model uses the reconstruction loss to optimize the weight parameters of the model, i.e., the mean square error between the reconstructed adjacency matrix and the original adjacency matrix is calculated as the loss value.

$$L_1 = \sum_{i=0}^{n} \text{MSE}\left(A, \hat{A}\right) = \sum_{i=0}^{n}\sum_{j=0}^{n}\left\|A_{ij} - \hat{A}_{ij}\right\|_2 \tag{8}$$

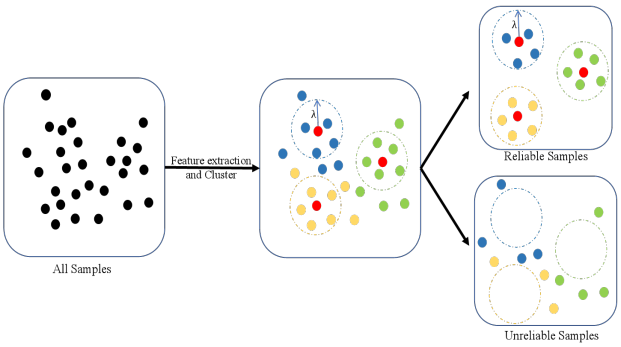


Fig. 2: Schematic diagram of the reliable samples selection mechanism. The black points in the graph represent the sample data of unknown categories, and the red samples represent cluster centers. Otherwise, points of the same color indicate that they belong to the same category after clustering. It can be seen from the diagram that after using the reliable samples selection mechanism, the clustered samples are divided into two categories. The samples with closer cluster centers are marked as reliable samples due to the high confidence of the cluster labels, and the rest of the samples are marked as unreliable samples.

We can obtain each sample's middle layer features Z with the above neural network model optimization. We perform the K-means clustering on the intermediate layer features Z to obtain the pseudo-label s_i of each sample and cluster center C_k, which is prepared for the next step in reliable sample selection.

E. Reliable Samples Selecting and Self-Supervised Training

In 2017, Thomas [10] et al. proposed the graph convolutional neural network model structure. The paper indicated that only a small number of sample labels need to be given in a huge dataset. The model can achieve high recognition accuracy with the help of the adjacency matrix and sample features. In the case of the Cora dataset, the total sample size is 2708, and the GCN model can achieve an accuracy of around 80.1 by only relying on the ground-truth labels of 440 randomly given samples. Our proposed self-supervised model is derived to some extent from this model. Since we conduct unsupervised experiments with unknown real labels of samples, we cannot provide actual sample labels to assist model training. But after the feature extraction and clustering of the graph attention autoencoder, we can obtain the cluster center C and the pseudo-label S of each sample. Then we can randomly select the pseudo-label of some samples and bring them into the GCN model for self-supervised training. It's worth noting that not all pseudo labels obtained from clustering are correct. Suppose the pseudo-labels of some samples are randomly selected as self-supervised information and passed to the downstream task for further learning. In that case, the model may be affected by some of the wrongly clustered noise data, thereby weakening the performance of the downstream model. Therefore, we propose a reliable sample selection mechanism.

Figure 2 is a diagram of the sample selection mechanism. Its purpose is to select some samples with high confidence in pseudo-labels and their labels from the huge datasets for self-supervised training. In this way, we can efficiently use the pseudo-labels obtained from the pretext task and at the same time minimize the impact of the noisy data in the clustering on the downstream tasks, which really plays a role in assisting the downstream model training. We use the K-means clustering algorithm in the clustering implementation process. The core idea of the algorithm is to optimize the clustering centers C alternately and the sample label S. The cluster center C is the centroid of all samples in the clustering. Its update method is shown in Equation 9. The sample label depends on the cluster center closest to the sample, and its update method is shown in Equation 10

$$c_k = \frac{1}{|c_i|} \sum_{z_i \in c_k} z_i \tag{9}$$

$$s_i = \underset{k}{\operatorname{argmin}} \|z_i - c_k\|_2 \tag{10}$$

It is not difficult to see from the K-means clustering algorithm that the closer the sample is to the center of the cluster, the greater the probability that the sample belongs to this category. In other words, if a sample is just at the edge of a certain category, it is likely to be divided into other clusters with the fine-tuning of the cluster center, and the probability of misclassification of this kind of samples is relatively high. The confidence level of the sample labels is relatively low. Based on this idea, we design a new reliable sample selection mechanism, as shown in Figure 2, which judges the reliability of the pseudo-label after clustering by comparing the distance of each sample to the cluster center. we then decide whether the pseudo-label obtained after clustering the sample is discarded or retained. The calculation method of the distance D_{ik} from the sample z_i to the cluster center c_k is shown in Equation 11.

$$D_{ik} = \frac{\|z_i - c_k\|^2}{\sum_{z_j \in c_k} \|z_j - c_k\|^2} \tag{11}$$

There may be uneven distribution among samples, resulting in sample numbers in some clusters becoming 0 after clustering. Taking one of the classic network datasets Wiki dataset as an example, there are 406 samples with label 2 in this dataset, but only nine samples with the label 15. To ensure that each cluster still contains at least one sample in the case of unbalanced samples, we will replace the original cluster center with a sample point closest to the cluster center, as shown in Equation 12:

$$c_k = \underset{z_i}{\operatorname{argmin}} \|z_i - c_k\|^2 \tag{12}$$

According to the improved Equation 12, the distance of each sample to its cluster center is calculated, and its normalized result is compared with the set distance threshold λ to judge whether the sample can be marked as a reliable sample. If $D_{ik} \leq \lambda$, it means that the sample is closer to the center of the cluster, and the pseudo-label obtained after the sample is clustered has high confidence. Mark it as a reliable sample, retain its pseudo-label, and assist downstream tasks for self-supervised training. If $D_{ik} > \lambda$, the pseudo-label of the sample has low confidence, and the pseudo-label of the sample will be discarded. After the aforementioned reliable sample selection mechanism, we can obtain some samples with pseudo-labels. The pseudo-labels of this part of the samples have a high confidence level, which can be used as supervision information for downstream tasks. The pseudo-labels of some samples are discarded, and these unlabeled samples will re-acquire new labels in the self-supervised training of downstream tasks. As shown in Figure 1, our downstream task model in self-supervised training uses a graph convolutional neural network. The network consists of two graph convolutional layers. The model continuously extracts the deep features of the samples during forwarding propagation, i.e., the output of the model after the graph convolution can be expressed as

$$h^{l+1} = \sigma\left(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} h^l W^l\right) \tag{13}$$

Where $\widetilde{A} = A + I_N$ is the self-loop adjacency matrix of the undirected graph G, I_N is the identity matrix, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{jj}$ is the degree matrix, and $W^{(l)}$ is the weight parameter that the graph convolutional neural network needs to train. $\sigma(\cdot)$ is the activation function and $h^{(l)}$ is the sample feature of the l-th layer. Among them,$H^{(0)} = X$.

The activation function of the first graph convolution layer of the model is the ReLU activation function. The activation function of the second graph convolution layer is the Softmax activation function. In order to prevent the model from believing in pseudo labels too much, we added a Dropout layer after the graph convolution layer to alleviate the overfitting phenomenon of the model by randomly discarding a certain proportion of neurons. Therefore, the output of the model can be expressed as,

$$y = g(\mathrm{X}, \mathrm{A}) = \text{dropout}\left(\text{softmax}\left(\widehat{A}\text{ReLU}\left(\widehat{A}XW^{(0)}\right)W^{(1)}\right)\right) \tag{14}$$

Where the $g(\cdot)$ function denotes the graph convolutional neural network, which $\widehat{A} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix. Since the datasets are extracted by GAE and clustered to obtain pseudo-labels of partial samples, the model is divided into two parts: training and testing. For training, the model takes the samples with pseudo-labels as supervision information and inputs an adjacency matrix A to optimize the entire graph convolutional neural network. The model uses the cross-entropy loss function to calculate the error, and the formula is shown in Equation 15.

$$L_2 = -\sum_{l \in y_L} \sum_{i=1}^{n} Y_{li} ln S_{li} \tag{15}$$

Where y_L represents the sample set with pseudo labels, Y_{li} denotes the predicted label of the i-th sample with pseudo-label after passing through the graph convolutional neural network, and S_{li} represents the pseudo-label obtained by the i-th reliable sample after clustering. When the model is tested, the trained model weight parameters are frozen, and all datasets X and adjacency matrices A are used as input to predict the class of all samples, including the previous untrustworthy samples. So far, the label information of all samples can be predicted.

Algorithm 1 Deep Self-Supervised Attentional Convolution Autoencooder Graph Clustering (DS-AGC)

1:**input:**Input data:$\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$;
2: Number of clusters:$\mathbf{k}$;
3: Hyperparameter:distance threshold:λ;
4: dropout rate:γ; Maximum iterations: $\mathbf{T}_{max}$.
5:**Output:**Graph Attention Autoencoder;
6: Graph Convolutional Neural Network;
7:**Labels:**$\hat{y}_{cluster}$,$\hat{y}_{self\_supervised}$
8:Pre-training **GAE**'s weight minimizing Eq.(8);
9:Get the hidden embedding $\mathbf{Z}$ using the encoder of **GAE** according to Eq.(8).
10:Using K-means clustering to get $\hat{y}_{cluster}$ and cluster centers $\mathbf{C}$;
11:Compute the distance between $\mathbf{x}_i$ and $\mathbf{c}_i$ via Eq.(11-12) and judge whether $\mathbf{x}_i$ is a reliable sample;
12:**while** $iter \leq \mathbf{T}_{max}$**do**
13: Train Graph Convolutional Neural using reliable samples and $\mathbf{A}$;
14:**End**

IV. Experionment

In this section, we present the experimental results of the proposed algorithm on three commonly used network datasets and compare them with recent deep graph convolutional clustering algorithms.

A. Benchmark Datasets

To verify the effectiveness of the method proposed in this paper, we conduct experiments on three standard network datasets. Since the proposed model is a self-supervised model, the true labels of all samples do not involve in the training of the model and are only used to measure the indicators and performance of the model. The datasets include: Core [26] , Citeseer [26] , and Wiki [27] . These three datasets are commonly used as benchmark datasets in social network analysis. The specific data are introduced as follows: The Cora dataset is a dataset consisting of machine learning papers classified into seven categories. There are a total of 2708 papers in the entire corpus, and only 1433 unique words are retained as the features of the samples after pre-processing. Each node in the graph represents a paper. Papers that cite or are cited to each other form the edge structure of the graph. The Citeseer dataset is similar to the Cora dataset in that it is a graph structure composed of citation relationships among the world's top conference papers. There are 3312 nodes in the graph, and each node contains 3703-dimensional features, which are classified into six categories. The Wiki dataset is mainly collected from the data in the Wikipedia database. Unlike the first two datasets, this dataset contains nearly 18,000 pieces of side information and 17 categories, so it is more challenging to identify than the other two. Detailed constitutions of these datasets are summarized in Table 1.

TABLE 1: DETAILS OF THE DATASETS

Datasets	Nodes	Features	Edge	Classes
Cora	2708	1433	5429	7
Citeseer	3312	3703	4732	6
Wiki	2405	4973	17981	17

B. Baseline Methods and Evaluation Metrics

We compare the proposed algorithm with related models from recent years. The comparison algorithms mainly include two categories. Clustering methods that only use node features or graph structure and the unsupervised clustering algorithms based on deep learning. Algorithms for unsupervised clustering using only features or structures include: K-means [28] , Spectral Cluster(SC) [29] , Graph-Encoder [15], Deep Walk [30] , DNGR [31] , and TADW [27] . The deep unsupervised clustering algorithms include: Graph Autoencoder(GAE) [12] , Variational Graph Autoencoder (VGAE)[12] , Adversarial Regularized Graph Autoencoder(ARGE) [23] , Adversarial Regularized Variational Graph Autoencoder(ARVGE) [23] , Deep Attentional Embedded Graph Clustering(DAEGC) [13] , Embedding Graph Autoencoder with Joint Clustering via Adjacency Sharing(EGAE-JOCAS) [32] , Embedding Graph Autoencoder(EGAE) [25] , Adaptive Graph Convolution (AGC) [33] , Graph Clustering via Variational Graph Embedding (GC-VGE) [34], and Spectral embedding network (SENet) [35]. The experiments were conducted on the data sets mentioned above, and the metrics are commonly used unsupervised metrics: accuracy (ACC), normalized mutual information (NMI), adjusted Rand coefficient (ARI), and F1. ACC is a measure used to measure the similarity between the predicted label and the true label. Its formula is as follows:

$$\text{ACC} = \frac{\sum_{i=1}^{n} \delta\left(y_i, map\left(\hat{y}_i\right)\right)}{n} \tag{16}$$

where y_i denotes the true label of the i-th sample and $\hat{y}_i$ denotes the corresponding predicted label.$\delta(x, y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases}$, n denotes the number of samples. NMI is the mutual information between the predicted label and the real label to measure the similarity between them, and its formula is as follows:

$$\text{NMI} = \frac{2\text{MI}}{\text{H}\left(y\right) + \text{H}\left(\hat{y}\right)} \tag{17}$$

Where MI stands for mutual information, and its form is $\text{MI} = H\left(y\right) - H\left(y|\hat{y}\right)$. It indicates that the predicted label contains a large amount of information about the true label. If there is a category label, the clustering result can also calculate the precision and recall like classification, and it can also calculate ARI and F1. The formula for ARI is as follows:

$$\text{ARI} = \frac{(\text{RI}) - \text{E(RI)}}{max(\text{RI}) - \text{E(RI)}} \tag{18}$$

Where RI stands for Rand index, defined as $\text{RI} = \frac{(\text{TP+TN})}{(\text{TP+FP+TN+FN})}$. The formula for F1 is as follows:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{19}$$

Where precision stands for the precision rate, defined as $precision = \frac{\text{TP}}{\text{TP+FP}}$; recall stands for the recall rate, defined as $recall = \frac{\text{TP}}{\text{TP+FN}}$.

C. Experiment Setting

We use a graph convolutional autoencoder based on the attention mechanism for our pretext task. The encoder consists of two layers of graph convolution layers. The feature dimension after the first layer of graph convolution is 256. After the second graph convolution, the feature dimension is 16, which is the feature $Z \in R^{n\times 16}$ of the middle layer. The Adam optimizer is used for model training with a learning rate of 0.001. The maximum number of iterations of the model is 100. Then the obtained intermediate embedding layer features are clustered, and reliable samples are selected. Where the distance threshold λ is set as a hyperparameter. After several experimental debugging, we found that the best performance was achieved with $\lambda = 0.30$ on the Cora dataset, and the best performance was divided into $\lambda = 0.25$ and $\lambda = 0.05$ on the Citeseer and Wiki datasets separately. We use a graph attention convolutional neural network for the downstream tasks, composed of two graph convolution layers. The output dimension of the last layer of the model is the total number of categories in the dataset. The dropout layer of the model has a dropout rate of 0.2. The Adam optimizer is used to optimize the model with a learning rate of 0.005, and the maximum number of iterations of the model is 200. We implement our method and other frameworks with the public toolbox of PyTorch 1.7.0. We run all the experiments on the platform of Ubuntu Linux 16.04 with NVIDIA GeForce GTX 1060 Graphics Processing Units (GPUs) and 12 GB memory size.

D. Comparison with the State-of-the-art Methods

The experimental results on the three benchmark datasets are shown in Table 2 and 3. The bolded data in the table represents the best performance tested under this metric. If the compared algorithms have not been tested on this dataset, the results are obtained experimentally using the hyperparameters mentioned in their papers. The corresponding result is replaced with a minus sign (-) if the code is not published or cannot be tested on the dataset. The specific experimental results are shown in table 2:

The experimental results shown in Tab.2 indicate that the proposed algorithm in this paper has good performance on both Cora and Citeseer datasets. The corresponding ACC indicators reached 75.44 and 67.87, respectively, which achieved the best experimental results compared with related works in recent years. The experimental results illustrate that our proposed method can effectively utilize the self-supervised model to make the model achieve a better clustering effect. The results are given in Table 3 separately for the Wiki dataset because the used comparison methods are different from the first two datasets. The model proposed in this paper still has a high ACC index on the Wiki dataset. Our model is slightly lower than the AGC and DAEGC methods in terms of the NMI and ARI indicators. Compared with other methods such as ARVGE, our model still has obvious performance advantages.

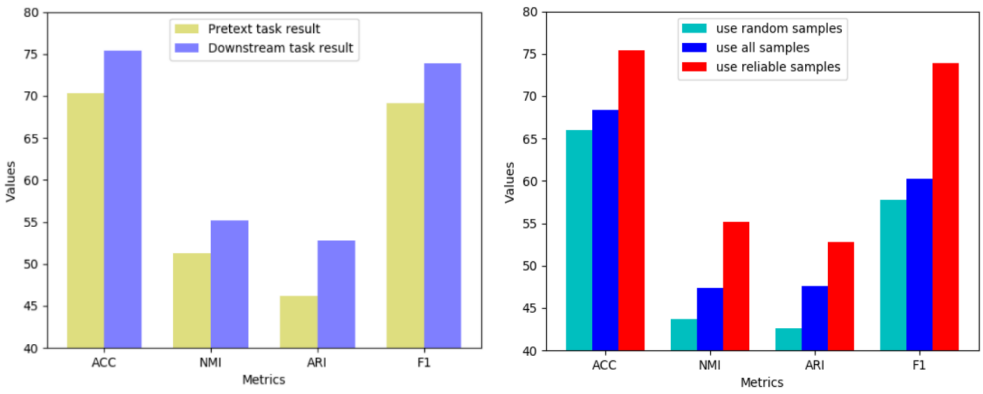
E. Experiment Discussion

1) Ablation experiments of the model: The model we used in the pretext task is a graph convolutional autoencoder based on the attention mechanism. The model obtains preliminary pseudo-labels by extracting the features of each sample and then uses the clustering algorithm, then selects samples by our designed reliable sample selection mechanism, and finally performs self-supervised learning to obtain the results. To verify the effectiveness of the innovations proposed in this paper on performance improvement, we mainly conduct two sets of comparative experiments to verify the effectiveness of our reliable sample selection mechanism and our proposed self-supervised model, respectively. Taking the Cora dataset as an example, as shown in Figure 3(a) below, the yellow bars in the figure are the clustering accuracy results obtained by only using the pretext task, and the purple is the clustering accuracy result after self-supervision. From the figure, it can be seen that the performance indicators of the self-supervised model have been improved compared with the pretext tasks, and the ARI indicator has increased by nearly 7%. It can be seen that our proposed self-supervised model plays a role in improving performance. Figure 3(b) shows our validation of the reliable sample selection mechanism. The green bars in the figure represent the clustering results we obtained by 50% of the randomly selected samples and their pseudo-labels to train our downstream model. The blue bars represent our results from supervised training with all samples and their pseudo-labels. The red bars represent results obtained from self-supervised training with samples screened by reliable sample selection mechanism and corresponding pseudo labels. From the Figure 3, it can be seen that since the randomly selected samples will have certain noise data, the downstream model may over-trust the pseudo label information of these random samples, thereby resulting in low accuracy. Suppose all samples are used to train the downstream model. In that case, it can be seen that the accuracy is similar to the clustering results obtained by the pretext task because it treats all the sample pseudo-labels as correct labels to train the downstream model. The output predicted label of the downstream model gradually approximates the pseudo label obtained by the pretext task. In contrast, the adaptive selection of reliable samples can effectively filter out the desired samples, thereby further improving the overall performance of the proposed model in this paper.

2) The influence of parameters λ in the model: In the experiment, we pay special attention to the distance threshold parameter for reliable sample screening. In this subsection, we will conduct an in-depth analysis of the impact of parameter changes on the model. First, the distance threshold λ, as a hyperparameter of the model, determines the selection of reliable samples, i.e., if the distance from the sample to the cluster center is less than In the experiment, we pay special attention to the distance threshold parameter for reliable sample screening. In this subsection, we will conduct an in-depth analysis of the impact of parameter changes on the model. First, the distance threshold , as a hyperparameter of the model, determines the selection of reliable samples, i.e., if the distance from the sample to the cluster center is less than λ, it will be marked as a reliable sample. Therefore, according to the meaning of this parameter, we can conclude that as the λ value increases, more and more samples will be marked as reliable samples. According to equation (11),

TABLE 2: Clustering results of various methods on Cora and Citeseer, Best results are highlighted in bold

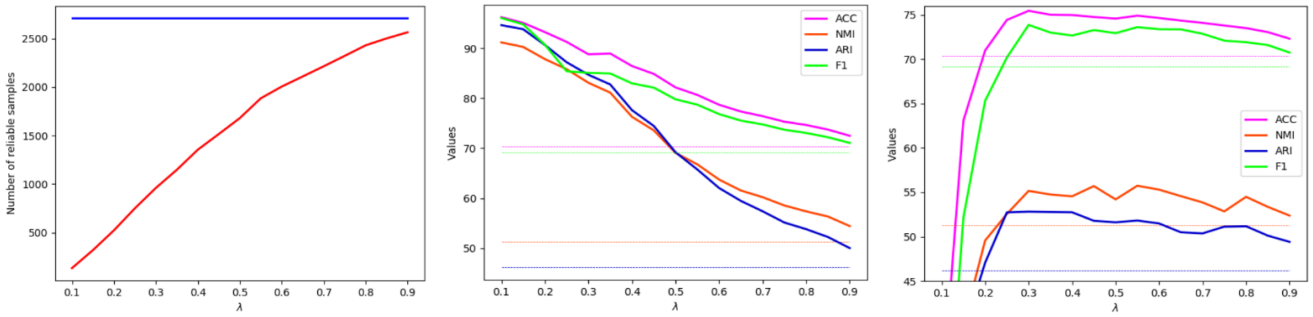
Dataset	Cora				Citeseer			
Metric	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
K-means[28] (MacQueen et al., 1967)	50	31.7	23.9	37.6	54.4	31.2	28.5	41.3
Spectral Cluster [29](Ng,Jordan,& Weiss,2001)	39.8	29.7	17.4	33.2	30.8	9	8.2	25.7
Graph Encod er[15](F.Tian et al.2014)	30.1	5.9	4.6	23	29.3	5.7	4.3	21.3
Deep Walk[3 0](Perozzi,Al-Rfou, Skiena,2014)	52.9	38.4	29.1	43.5	39	13.1	13.7	30.5
DNGR[31](S.Cao et al.,2016)	41.9	31.8	14.2	34	32.6	18	4.3	30
TADW[27](C.Yang et al.,2015)	53.6	36.6	24	40.1	52.9	32	28.6	43.6
GAE[12](Kipf et al.,2016)	53	39.7	29.3	41.5	38	17.4	14.1	29.7
VGAE[12](Kipf et al.,2016)	59.2	40.8	34.7	45.6	39.2	16.3	10.1	27.8
ARGE[23](S.Pan et al.,2018)	64	44.9	35.2	61.9	57.3	35	34.1	54.6
ARVGE[23](S.Pan et al.,2018)	63.8	45	37.4	62.7	54.4	26.1	24.5	52.9
DAEGC[13]C.Wang et al.,(2019)	70.4	52.8	49.6	68.2	67.2	39.7	41	63.6
GC-VGE[34](Lin Guo et al.,2022)	70.7	53.6	48.2	69.5	66.6	40.9	41.5	63.4
SENet[35](Zhang, Xiaotong et al., 2021)	71.9	55.1	49.0	-	67.5	41.7	42.4	-
EGAE-JOCAS [32](X.Li et al.,2020)	68.6	53.6	-	-	67.4	40.5	-	-
EGAE[25](H.Zhang et al.,2020)	72.42	53.96	47.22	-	67.42	41.18	43.18	-
DSAGC	75.44	55.16	52.82	73.85	67.87	41.43	42.29	62.78



(a) (b)

Fig. 3: Ablation experiment comparison

we can see that the value of λ should be set between 0 and 1. In the experiments, we set the value to 0.10, 0.15, 0.20, ..., 0.90 for the Cora dataset, setting a value every 0.5. As shown in Figure 4(a), the blue line is the total number of 2708 samples in the Cora dataset, and the red line is the number of selected reliable samples. It can be seen that as the value λ increases, the number of reliable samples increases. The number of samples is getting closer and closer to the total sample size, which is consistent with the above conclusions. In order to verify that the selected reliable samples have achieved the effect of removing part of the noise, we conducted the following comparative experiments. As shown in Figure 4(b), we compare pseudo labels of reliable samples screened by different thresholds λ with ground truth



(a) (b) (c)

Fig. 4: The effect of parameters λ on the model. (a) The total number of samples in the dataset and the influence of the λ value on the number of reliable samples. (b) The influence of the λ value on the accuracy of the reliable samples. (c) The influence of the λ value on the downstream classification results.

TABLE 3: Experimental results on the Wiki Dataset

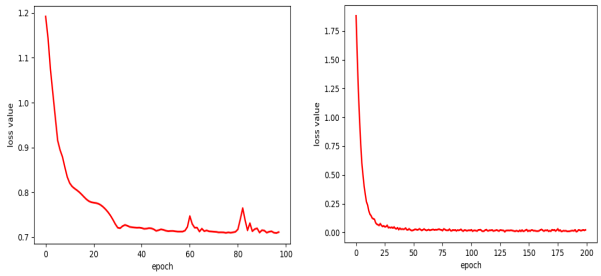
Method	ACC	NMI	ARI
K-means[28]	40.43	42.91	15.03
Spectral Cluster[29]	22.04	18.17	1.46
Graph Encoder[15]	20.67	12.07	0.49
Deep Walk[30]	38.46	32.38	17.03
DNGR[31]	37.58	35.85	17.97
TADW[27]	30.96	27.13	4.54
GAE[12]	32.85	29.02	7.8
ARGE[23]	38.05	34.45	11.22
ARVGE[23]	38.67	33.88	10.69
GC-VGE[34]	48.8	47.6	28.4
AGC[33]	47.65	45.28	34.3
DAEGC[13]	48.2	44.8	33.1
DSAGC	50.35	42.93	30.8

of samples. Through the comparison results, we can draw the following conclusions: when the λ value is smaller, the more stringent the screening conditions are, and the selected samples are less, but the pseudo-labels are very similar to the real labels, and the pseudo labels are more reliable, the purer the sample, and the less noisy data it contains. Conversely, as the λ value is increased, the screening conditions will be relaxed, the sample size will increase, and the accuracy of the reliable samples will gradually decrease. However, the accuracy of the reliable samples is still higher than that of the unprocessed samples (shown by the dashed line in Figure 4(b)). Therefore, it can be seen that the screened samples can effectively remove some noise, thereby improving the accuracy of the reliable samples.

However, in order to provide more supervised information for the downstream model, we need to adopt a compromise solution, i.e., to select a parameter that can not only ensure that there are enough sample pseudo-labels as supervised information to assist in training the downstream model but also ensure that the confidence level of the pseudo-labels is high to prevent noise data from interfering with the model. After comparison, it can be seen that in the Cora dataset, when λ approaches 0, although the pseudo label confidence of the sample is high, the number of reliable samples is too small, resulting in too little supervision information provided by the pretext task. Therefore, it cannot assist downstream tasks to improve model performance. When λ approaches 1, almost all samples participate in the training of the downstream task, but the samples will contain some noise, and the model accuracy will gradually approach the output accuracy of the pretext task, thus losing the training value of the downstream task itself. From Figure 4 (c), it can be seen that when λ takes 0.30, the indicators of the sample are optimal. At this time, the model takes into account the supervision information provided by maximizing the pseudo-label of the sample and takes into account the confidence of the pseudo-label.

3) Convergence of the model and variation of the loss function: In the pretext task, the model we used is the graph attention convolutional auto-encoder. In the backpropagation process of the model, we use the mean square error to calculate the reconstructed adjacency matrix and the original adjacency matrix error as the objective function to optimize the model by minimizing equation (8). In order to verify the convergence of our model, we plot the change of loss in each iteration. As shown in Figure 5 (a), it can be seen that the loss of the model decreases significantly in the first 40 epochs. After the 40th generation, the loss no longer reduces significantly and gradually tends to be smooth. When the model converges to a certain extent, there will be small fluctuations, but the overall trend is declining, i.e., as the iteration period in the pretext task increases, the loss gradually decreases, and the model gradually converges. Our downstream task uses a graph convolutional neural network, which uses reliable samples and their pseudo labels for training. After training, all samples are used as input for testing to obtain the final result. The model is trained with the cross-entropy loss function. And the model is optimized by minimizing the equation (15). Like the model in the pretext task, we record the changes in epoch and loss to observe the convergence of the model. As shown in Figure 5 (b), the loss value of the model tends to stabilize after 50 epochs, and the model begins to converge. In summary, our proposed model can gradually converge as the epoch increases during the training process.

4) Visualization results of sample distribution: In this subsection, we observe the distribution of samples through visual-



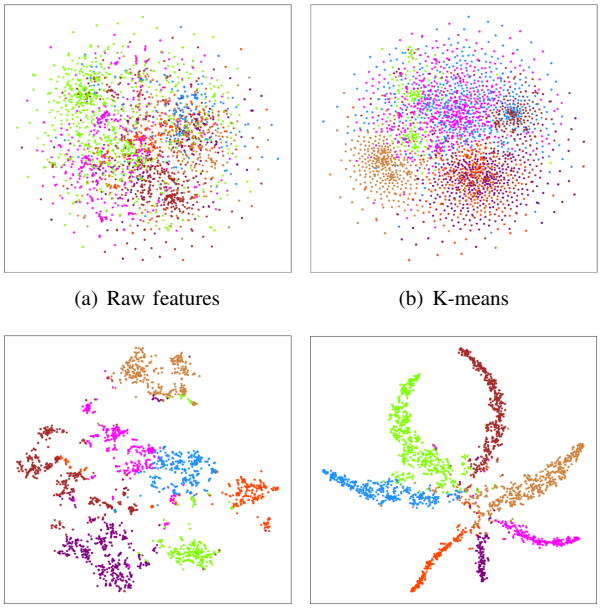
(a) (b)

Fig. 5: The variation of the loss function of the proposed model. (a) loss change in the pretext task. (b) loss change in the downstream task.

ization. We use the t-SNE dimensionality reduction algorithm to reduce the features of the samples to two dimensions and then visualize them. Taking the results of the Cora dataset as an example, as shown in Figure 6, Figure 6 (a) shows the distribution of the original features of the samples in the Cora dataset after dimensionality reduction. The samples are classified into seven categories, and different colors represent different categories. Figure 6 (a) shows that it is not easy to make a clear division of each category after the sample has been dimensionality reduced without the aid of the graph structure. Figure 6 (b) shows the label distribution obtained from the Cora dataset after clustering by K-means. Only K-means is used to divide the sample categories, ignoring the graph structure information, and the clustering effect also has certain limitations under high-dimensional data. It can be seen from the figure that the distribution of samples after K-means clustering is also chaotic, but the samples have a preliminary division. Figure 6(c) shows the results obtained by clustering after feature extraction by graph convolutional autoencoder. This feature uses the feature of the intermediate embedding layer of the graph autoencoder. Therefore, considering the graph structure information, the feature of the intermediate embedding layer is reduced in dimension. After that, the obtained visualization has obvious cluster classification, but some misclassified samples are still at the cluster's edge. Figure 6 (d) shows the visualization effect of our proposed model after dimensionality reduction. From the figure, it can be seen that our model can provide a better cluster classification of samples. Compared with the phenomenon of edge noise in the GAE model, we have also obtained better optimization, resulting in higher sample similarity between clusters and better clustering.

V. Conclusion

In this paper, we propose a novel self-supervised graph convolutional neural network framework for graph datasets in social network analysis. In this framework, the classification of sample categories can be completed without the real labels of the samples. The pseudo-labels obtained by the pretext tasks are used to construct supervision information to assist in optimizing the downstream tasks, thereby further improving



(a) Raw features (b) K-means

(c) GAE (d) DSAGC

Fig. 6: Visualization of the common latent representation given by different methods with t-SNE on the Cora dataset.

the model's performance. In addition, we also propose a reliable sample selection mechanism, which effectively mitigates the impact of noisy data on the whole self-supervised framework. Extensive experiments showed that our model has overall performance on several public social network public datasets. A large number of experimental comparisons also demonstrate the effectiveness of the DSAGC algorithm. In the case of unbalanced sample distribution, although the effect of our model is not very bad, such as the results on the wiki dataset, the performance still needs to be improved. Therefore, we will also focus on datasets for unbalanced samples for further exploration and research. We will introduce an improved spectral clustering algorithm [36] and semi-supervised learning [37] to improve the clustering performance further.

A. CRediT authorship contribution statement

Hu Lu: Conceptualization, Methodology, Writing - original draft. Chao Chen: Software, Data curation, Validation. Haotian Hong: Investigation, Visualization. Hai Wang: Supervision, Writing - review editing. Shaohua Wan: Writing - review editing.

B. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Sun, S. Yin, H. Li, L. Teng, and S. Karim, "Gpogc: Gaussian pigeon-oriented graph clustering algorithm for social networks cluster," *IEEE Access*, vol. 7, pp. 99 254–99 262, 2019. I

[2] L. Shi, H. Tong, J. Tang, and C. Lin, "Vegas: Visual influence graph summarization on citation networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3417–3431, 2015. I

[3] C. Lin, Y.-R. Cho, W.-C. Hwang, P. Pei, and A. Zhang, "Clustering methods in protein-protein interaction network," *Knowledge Discovery in Bioinformatics: techniques, methods and application*, pp. 1–35, 2007. I

[4] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007. I

[5] M. C. Nascimento and A. C. De Carvalho, "Spectral methods for graph clustering–a survey," *European Journal of Operational Research*, vol. 211, no. 2, pp. 221–231, 2011. I

[6] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 2005, pp. 274–285. I

[7] F. Nie, C. Ding, D. Luo, and H. Huang, "Improved minmax cut graph clustering with nonnegative relaxation," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 451–466. I

[8] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008. I

[9] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018. I

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016. I, II-A, III-E

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017. I, II-A

[12] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016. I, II-B, IV-B, IV-B

[13] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," *arXiv preprint arXiv:1906.06532*, 2019. I, II-B, IV-B

[14] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-second AAAI conference on artificial intelligence*, 2018. II-A

[15] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014. II-A, IV-B

[16] W. E. Donath and A. J. Hoffman, "Lower bounds for the partitioning of graphs," in *Selected Papers Of Alan J Hoffman: With Commentary*. World Scientific, 2003, pp. 437–442. II-A

[17] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000. II-A

[18] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487. II-B

[19] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010. II-B

[20] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International conference on neural information processing*. Springer, 2017, pp. 373–382. II-B

[21] Y. Han, Q. Fang, J. Hu, S. Qian, and C. Xu, "Gaeat: Graph auto-encoder attention networks for knowledge graph completion," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2053–2056. II-B

[22] X. Guo, E. Zhu, X. Liu, and J. Yin, "Deep embedded clustering with data augmentation," in *Asian conference on machine learning*. PMLR, 2018, pp. 550–565. II-B

[23] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018. I, II-B, IV-B, IV-B

[24] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018. II-B

[25] H. Zhang, R. Zhang, and X. Li, "Embedding graph auto-encoder for graph clustering," *arXiv preprint arXiv:2002.08643*, 2020. I, II-B, IV-B

[26] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008. IV-A, IV-A

[27] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Twenty-fourth international joint conference on artificial intelligence*, 2015. IV-A, IV-B

[28] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297. IV-B

[29] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001. IV-B

[30] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710. IV-B

[31] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016. IV-B

[32] X. Li, H. Zhang, and R. Zhang, "Embedding graph auto-encoder with joint clustering via adjacency sharing," *arXiv e-prints*, pp. arXiv–2002, 2020. IV-B

[33] X. Zhang, H. Liu, Q. Li, and X.-M. Wu, "Attributed graph clustering via adaptive graph convolution," *arXiv preprint arXiv:1906.01210*, 2019. IV-B

[34] L. Guo and Q. Dai, "Graph clustering via variational graph embedding," *Pattern Recognition*, vol. 122, p. 108334, 2022. I, II-B, IV-B

[35] X. Zhang, H. Liu, X.-M. Wu, X. Zhang, and X. Liu, "Spectral embedding network for attributed graph clustering," *Neural Networks*, vol. 142, pp. 388–396, 2021. I, II-B, IV-B

[36] "An efficient Nyström spectral clustering algorithm using incomplete Cholesky decomposition," *Expert Systems with Applications* pp.115813,2021 V

[37] Z. Jiang, Y. Zhan, Q. Mao, and Y. Du, "Semi-supervised clustering under a compact-cluster assumption," *IEEE Transactions on Knowledge and Data Engineering*, 2022. V