Deep Self-Supervised Graph Attention Convolution Autoencoder for Networks Clustering

Chao Chen, Hu Lu, Haotian Hong, Hai Wang, *Senior Member, IEEE*, and Shaohua Wan, *Senior Member, IEEE*

***Abstract*—In recent years, graph-based deep learning algorithms have attracted widespread attention in the field of consumer electronics. Still, most of the current graph neural networks are based on supervised learning or semi-supervised learning, which often relies on the true labels of the given samples as auxiliary information. To solve this problem, we propose a Deep Self-Supervised Attention Convolution Autoencoder Graph Clustering (DSAGC) model 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, we obtain pseudo-labels of samples by graph attention autoencoder and clustering, then adopt the proposed reliable sample selection mechanism to gain a high confidential sample. These samples and corresponding pseudo-labels are selected as input of downstream task for helping the downstream task to finish training. The model obtains the predictive labels of the sample from the downstream task. This model does not depend on the true label of the sample for training and uses the pseudo-labels produced by unsupervised 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. The proposed method can be used for consumer behavior analysis. We release the source code at https://github.com/hulu88/DSAGC.**

***Index Terms*—Self-supervised, autoencoder, reliable sample selection, social networks clustering.**

I. Introduction

THE SOCIAL relationships between consumer electronics users can also automatically generate social networks, and it is necessary to use an unsupervised clustering method to analyze these consumer behaviors. The graph-based clustering algorithm divides the nodes into several subgraphs with no intersection according to the similarity between the features. Graph clustering can realize the partitioning of a social network. Graph clustering [1], [2] 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 by dividing the vertices in the graph and finally completes the clustering task [3], [4]. With the introduction of graph neural networks [5], [6], 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 [7] and graph attention networks [8]. To get rid of the model's dependence on the true labels of the samples, some unsupervised graph clustering algorithms have been proposed one after another. These models reconstructed the adjacency matrix A through an autoencoder and then cooperated with traditional clustering algorithms to realize the task of deep graph clustering, such as Variational Graph Auto-Encoders [9] and Deep Attentional Embedded Graph Clustering [10].

With the proposal of various deep graph clustering, the application of deep neural networks in graph structure has been promoted. Some new methods combining deep autoencoders and graph convolutional networks have also appeared [10], [20], [22], [31], [32]. However, these aforementioned methods rely mainly on the feature extraction capability of graph neural networks without making full use of their clustering information. Meanwhile, the clustering performance after deep feature extraction of samples is often unsatisfactory due to the noise present in the data itself.

Considering these problems above, we therefore propose a new self-supervised graph convolutional self-encoder clustering model that makes full use of the self-supervised information in data analysis. The model uses the deep graph clustering task as a pretext task to obtain clustering information. At the same time, we use a new reliable sample selection mechanism to extract some pseudo-labels with high confidence. These pseudo-labels and their samples are then used to assist the learning of downstream tasks to improve the 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 network clustering. The framework integrates graph convolution neural network, graph autoencoder and clustering algorithm, and can still complete graph network division tasks without true labels of samples

Manuscript received 26 February 2023; accepted 17 May 2023. Date of publication 25 May 2023; date of current version 21 February 2024. This work was supported in part by the Key Project of Shenzhen City Special Fund for Fundamental Research under Grant 202208183000751. *(Corresponding author: Hu Lu.)*

Chao Chen, Hu Lu, and Haotian Hong are with the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China (e-mail: michealcchao@163.com; luhu@ujs.edu.cn; 2877153190@qq.com).

Hai Wang is with the School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang 212013, China (e-mail: wanghai1019@163.com).

Shaohua Wan is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China (e-mail: shaohua.wan@uestc.edu.cn).

Digital Object Identifier 10.1109/TCE.2023.3279836

1558-4127 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

in the model. At the same time, we also designed a complete model framework for it, which divides the deep graph clustering model into two parts, namely the pretext 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.

2) We propose a new reliable sample selection mechanism, which can select some sample points that are closer to the cluster center from the clustering results as reliable samples. The cluster labels of these samples tend to have high confidence and contain less noise. Using these data to train the downstream model can effectively mitigate the impact of noise data on the performance of the downstream 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 [11] or clustering [12]. 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 in [13]. 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 [14]. 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 [7], the convolution computation method on the graph frequency domain and the spatial domain is presented. Reference [8] added the attention mechanism to the graph neural network. They utilized 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. But graph datasets tend to be large, the annotation is expensive, and it is difficult to annotate all of them, thus these factors limit supervised learning methods.

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. Reference [15] proposed a classical deep clustering algorithm named Deep Embedded Clustering (DEC), which extracts sample features through a denoising stack autoencoder [16]. 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. Reference [17] 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) [9] is similar to the convolutional autoencoder, consisting of encoder and decoder 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. Reference [10] combined the idea of graph attention convolutional autoencoder [18] and embedded clustering [19] 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 [20], which incorporates the adversarial idea in the generative adversarial networks [21] based on the graph autoencoder and then mines the prior distribution of the features of the sample embedding layer. Reference [22] 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. Reference [32] proposed a spectral embedding network (SENet) for attribute graph clustering. Reference [31] proposed a new variational graph autoencoder based on graph convolutional networks. The model considers the joint generative model of graph structure and node attribute's enhanced impact on embedding output.

All of the above mentioned works neglect the exploitation of clustering information. Relatively little work has been done to exploit clustering labels in unsupervised tasks of graphs. M3C [37] performs both clustering and semi-supervised classification tasks, and selects predictions with consistent classification and clustering results as pseudo-labels through an alignment mechanism, which is used to expand to the labeled dataset, followed by iterative training to optimize the model. ICF-GCN [36] takes a similar approach to M3C [37] in an expectation-maximizing way, while using a relational matrix to solve the alignment problem. While these works also use pseudo-label, their acquisition relies on an

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

upstream semi-supervised task, in which the results of the semi-supervised task need to be aligned to obtain reliable pseudo-label. In contrast, our pseudo-label generation does not rely on labeled data, and we propose a new reliable sample selection mechanism to improve the confidence of the pseudo-label.

III. The Proposed Method

A. Overall Framework

The structural diagram of our proposed model is shown in Fig. 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 the formula (1). Then features Z are used to reconstruct the matrix A. The reconstruction loss is used as the loss function during the training of the model. 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 selection mechanism to select 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.

B. 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 Fig. 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 formula (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:

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

where || 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:

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

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 encoder 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 formula (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 decoder layer of the graph attention autoencoder reconstructs the original graph structure by calculating the inner product of the intermediate layer features Z. The decoder process can be represented by the formula (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.

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

We can obtain each sample's middle layer features Z with the above neural network model optimization. We perform the

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

Fig. 1. The Conceptual illustration of Deep Self-Supervised Attention 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. Then, the model extracts the features of each sample X to obtain Z and clustering. Next, the samples are selected according to the cluster centers and labels to assist the downstream tasks to perform self-supervised graph convolution training, which can obtain better clustering results.

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.

C. Reliable Samples Selecting and Self-Supervised Training

Reference [7] proposed the graph convolutional neural network model structure. The paper indicated that given only a small number of sample labels in a huge graph dataset, the model could achieve high classification 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-labels of some samples and bring them into the GCN model for self-supervised training. It is 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.

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. 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 higher confidence level of the cluster labels, and the rest of the samples are marked as unreliable samples.

Fig. 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 the formula (9).

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

The sample label depends on the cluster center closest to the sample, and its update method is shown in the formula (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 sample 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 Fig. 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 the formula (11).

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

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 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 formula (12):

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

According to the improved formula (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 Fig. 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,

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

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 formula (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.

Our algorithm is described in Algorithm 1.

IV. Experiment

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.

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Deep Self-Supervised Attention Convolution Autoencooder Graph Clustering (DSAGC)

Require: Input data:$G = (V, E, X)$; Number of clusters:k; distance threshold:λ; dropout rate:γ; Maximum iterations: $\mathbf{T}_{max}$.

Ensure: Graph Attention Autoencoder; Graph Convolutional Neural Network; Labels: $\hat{y}_{cluster}$,$\hat{y}_{self_supervised}$.

Pre-training **GAE**'s weight minimizing Eq. (8);
Get the hidden embedding $\mathbf{Z}$ using the encoder of **GAE** according to Eq. (6).
Using K-means clustering to get $\hat{y}_{cluster}$ and cluster centers $\mathbf{C}$;
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;
while $iter \leq \mathbf{T}_{max}$ **do**
    Train Graph Convolutional Neural using reliable samples and $\mathbf{A}$;
end while
Input X and A to GCN and get the predicted label $\hat{y}_{self_supervised}$
return the predicted labels of all samples.

TABLE I
DETAILS OF THE DATASETS

Dataset	Nodes	Features	Edge	Classes
Cora [23]	2708	1433	5429	7
Citeseer [23]	3312	3703	4732	6
Wiki [24]	2405	4973	17981	17

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: Cora [23], Citeseer [23], and Wiki [24]. These three datasets are commonly used as benchmark datasets in social network analysis. Detailed constitutions of these datasets are summarized in Table I.

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 unsupervised clustering algorithms based on deep learning. Algorithms for unsupervised clustering using only features or structures include: K-means [25], Spectral Cluster(SC) [26], Graph-Encoder [12], Deep Walk [27], DNGR [28], and TADW [24]. The deep unsupervised clustering algorithms include: Graph Autoencoder(GAE) [9], Variational Graph Autoencoder (VGAE) [9], Adversarial Regularized Graph Autoencoder(ARGE) [20], Adversarial Regularized Variational Graph Autoencoder(ARVGE) [20], Deep Attentional Embedded Graph Clustering(DAEGC) [10], Embedding Graph Autoencoder with Joint Clustering via Adjacency Sharing(EGAE-JOCAS) [29], Embedding Graph Autoencoder(EGAE) [22], Adaptive Graph Convolution (AGC) [30], Graph Clustering via Variational Graph Embedding (GC-VGE) [31], and Spectral embedding network (SENet) [32].

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.

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 PyTorch 1.7.0. We run all the experiments on the platform of Ubuntu Linux 16.04 with NVIDIA GeForce GTX 1060.

D. Comparison With the State-of-the-Art Methods

The experimental results on the three benchmark datasets are shown in Tables II and III. 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 II.

The experimental results shown in Table II 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.

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

TABLE II
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 [25] (MacQueen et al.1967)	50.00	31.70	23.90	37.60	54.40	31.20	28.50	41.30
Spectral Cluster [26](Ng et al.2001)	39.80	29.70	17.40	33.20	30.80	9.00	8.20	25.70
Graph Encod [12](F.Tian et al.2014)	30.10	5.90	4.60	23.00	29.30	5.70	4.30	21.30
Deep Walk [27](Perozzi et al.2014)	52.90	38.40	29.10	43.50	39.00	13.10	13.70	30.50
TADW [24] (C.Yang et al.2015)	53.60	36.60	24.00	40.10	52.90	32.00	28.60	43.60
DNGR [28] (S.Cao et al.2016)	41.90	31.80	14.20	34.00	32.60	18.00	4.30	30.00
GAE [9](Kipf et al.2016)	53.00	39.70	29.30	41.50	38.00	17.40	14.10	29.70
VGAE [9] (Kipf et al.2016)	59.20	40.80	34.70	45.60	39.20	16.30	10.10	27.80
ARGE [20](S.Pan et al.2018)	64.00	44.90	35.20	61.90	57.30	35.00	34.10	54.60
ARVGE [20](S.Pan et al.2018)	63.80	45.00	37.40	62.70	54.40	26.10	24.50	52.90
DAEGC [10](C.Wang et al.2019)	70.40	52.80	49.60	68.20	67.20	39.70	41.00	**63.60**
EGAE-JOCAS [29](X.Li et al.2020)	68.60	53.60	-	-	67.40	40.50	-	-
EGAE [22] (H.Zhang et al.2020)	72.42	53.96	47.22	-	67.42	41.18	**43.18**	-
SENet [32](Zhang et al. 2021)	71.90	55.10	49.0	-	67.50	**41.70**	42.40	-
GC-VGE [31](Lin Guo et al.2022)	70.70	53.60	48.20	69.50	66.60	40.90	41.50	63.40
DSAGC	**75.44**	**55.16**	**52.82**	**73.85**	**67.87**	41.43	42.29	62.78

TABLE III
EXPERIMENTAL RESULTS ON THE WIKI DATASET

Method	ACC	NMI	ARI
K-means [25](MacQueen et al.1967)	40.43	42.91	15.03
Spectral Cluster [26](Ng et al.2001)	22.04	18.17	1.46
Graph Encod [12](F.Tian et al.2014)	20.67	12.07	0.49
Deep Walk [27](Perozzi et al.2014)	38.46	32.38	17.03
TADW [24](C.Yang et al.2015)	30.96	27.13	4.54
DNGR [28](S.Cao et al.2016)	37.58	35.85	17.97
GAE [9](Kipf et al.2016)	32.85	29.02	7.8
ARGE [20](S.Pan et al.2018)	38.05	34.45	11.22
ARVGE [20](S.Pan et al.2018)	38.67	33.88	10.69
AGC [30](X.Zhang et al.2019)	47.65	45.28	**34.3**
DAEGC [10](C.Wang et al.2019)	48.2	44.8	33.1
GC-VGE [31](Lin Guo et al.2022)	48.8	**47.6**	28.4
DSAGC	**50.35**	42.93	30.8

The results are given in Table III 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 the left side of Fig. 3(a) below, the green bars in the figure are the clustering accuracy results without the pretext task and the downstream task. The blue bars in the figure are the clustering accuracy results obtained by only using the pretext task, and the pink 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.

Fig. 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 Fig. 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 λ, it will be marked as a reliable sample. Therefore, according to the meaning of this parameter, we can

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

Fig. 3. Ablation experiment comparison. (a) Comparison of the results of pretext task and downstream tasks. (b) The effect of reliable sample selection mechanism on experimental results.

conclude that as the λ value increases, more and more samples will be marked as reliable samples. According to formula (11), 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.

As shown in Fig. 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 Fig. 4(b), we compare pseudo-labels of reliable samples screened by different thresholds λ with the ground truth 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 Fig. 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 Fig. 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

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

(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.

(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.

(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 points with different colors in the figure correspond to different categories of each sample.

by minimizing the formula (8). In order to verify the convergence of our model, we plot the change of loss in each iteration. As shown in Fig. 5 (a), it can be seen that the loss of the model decreases significantly in the first 40 epochs. After the 40-th 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 formula (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 Fig. 5 (b), the loss value of the model tends to stabilize after 50 epochs, and the model begins to converge. 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 visualization. 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, Fig. 6(a) shows the distribution of the original features of the samples after dimensionality reduction. The samples are classified into seven categories, and different colors represent different categories. Fig. 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. Fig. 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. Fig. 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. Fig. 6(d) shows the visualization effect of our proposed model after dimensionality

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.

reduction. From Fig. 6(d), 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. In this framework, the pseudo-labels obtained by the pretext tasks are used to construct supervision information to assist in optimizing the downstream tasks, thereby further improving the model's performance. We also propose a reliable sample selection mechanism. A large number of experimental comparisons also demonstrate the effectiveness of the DSAGC. The proposed method also has shortcomings. 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 clustering algorithm [33] and semi-supervised learning [34], [35] to improve the clustering performance.

References

[1] S. E. Schaeffer, "Graph clustering," *Comput. Sci. Rev.*, vol. 1, no. 1, pp. 27–64, 2007.

[2] M. C. Nascimento and A. C. De Carvalho, "Spectral methods for graph clustering—A survey," *Eur. J. Oper. Res.*, vol. 211, no. 2, pp. 221–231, 2011.

[3] Z. He, S. Wan, M. Zappatore, and H. Lu, "A similarity matrix low-rank approximation and inconsistency separation fusion approach for multi-view clustering," *IEEE Trans. Artif. Intell.*, early access, May 1, 2023, doi: 10.1109/TAI.2023.3271964.

[4] F. Nie, C. Ding, D. Luo, and H. Huang, "Improved MinMax Cut graph clustering with nonnegative relaxation," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2010, pp. 451–466.

[5] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

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

[7] T. N. Kipf and M. Welling, 'Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

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

[9] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.

[10] 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*.

[11] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4438–4445.

[12] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. AAAI Conf. Artif. Intell.*, vol. 28, 2014, pp. 101–105.

[13] W. E. Donath and A. J. Hoffman, "Lower bounds for the partitioning of graphs," in *Selected Papers of Alan J Hoffman: With Commentary*. London, U.K.: World Sci., 2003, pp. 437–442.

[14] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[15] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[16] 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," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, 2010.

[17] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 373–382.

[18] Y. Han, Q. Fang, J. Hu, S. Qian, and C. Xu, "GAEAT: Graph auto-encoder attention networks for knowledge graph completion," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 2053–2056.

[19] X. Guo, E. Zhu, X. Liu, and J. Yin, "Deep embedded clustering with data augmentation," in *Proc. Asian Conf. Mach. Learn.*, 2018, pp. 550–565.

[20] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*.

[21] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

[22] H. Zhang, R. Zhang, and X. Li, "Embedding graph auto-encoder for graph clustering," 2020, *arXiv:2002.08643*.

[23] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

[24] 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.

[25] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probab.*, vol. 1. Oakland, CA, USA, 1967, pp. 281–297.

[26] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 849–856.

[27] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 701–710.

[28] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016, pp. 1145–1152.

[29] X. Li, H. Zhang, and R. Zhang, "Embedding graph auto-encoder with joint clustering via adjacency sharing," 2020, *arXiv:2002.08643v1*.

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

[31] L. Guo and Q. Dai, "Graph clustering via variational graph embedding," *Pattern Recognit.*, vol. 122, Feb. 2022, Art. no. 108334.

[32] X. Zhang, H. Liu, X. Zhang, and X. Liu, and X.-M. Wu, "Spectral embedding network for attributed graph clustering," *Neural Netw.*, vol. 142, pp. 388–396, Oct. 2021.

[33] H. Lu, C. Chen, H. Wei, Z. Ma, K. Jiang, and Y. Wang, "Improved deep convolutional embedded clustering with re-selectable sample training," *Pattern Recognit.*, vol. 127, Jul. 2022, Art. no. 108611.

[34] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, "A semisupervised recurrent convolutional attention model for human activity recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 1747–1756, May 2020.

[35] M. Luo, X. Chang, L. Nie, Y. Yang, A. G. Hauptmann, and Q. Zheng, "An adaptive semisupervised feature analysis for video semantic recognition," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 648–660, Feb. 2018.

[36] Z. Hu, G. Kou, H. Zhang, N. Li, K. Yang, and L. Liu, "Rectifying pseudo labels: Iterative feature clustering for graph representation learning," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 720–729.

[37] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage selfsupervised learning for graph convolutional networks on graphs with few labeled nodes," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 5892–5899.

Authorized licensed use limited to: Indian Institute of Technology (ISM) Dhanbad. Downloaded on August 21,2025 at 18:26:57 UTC from IEEE Xplore. Restrictions apply.