



Deep Self-Supervised Attributed Graph Clustering for Social Network Analysis

Hu Lu^{1,2} · Haotian Hong¹ · Xia Geng¹

Accepted: 12 March 2024 / Published online: 1 April 2024
© The Author(s) 2024

Abstract

Deep graph clustering is an unsupervised learning task that divides nodes in a graph into disjoint regions with the help of graph auto-encoders. Currently, such methods have several problems, as follows. (1) The deep graph clustering method does not effectively utilize the generated pseudo-labels, resulting in sub-optimal model training results. (2) Each cluster has a different confidence level, which affects the reliability of the pseudo-label. To address these problems, we propose a Deep Self-supervised Attribute Graph Clustering model (DSAGC) to fully leverage the information of the data itself. We divide the proposed model into two parts: an upstream model and a downstream model. In the upstream model, we use the pseudo-label information generated by spectral clustering to form a new high-confidence distribution with which to optimize the model for a higher performance. We also propose a new reliable sample selection mechanism to obtain more reliable samples for downstream tasks. In the downstream model, we only use the reliable samples and the pseudo-label for the semi-supervised classification task without the true label. We compare the proposed method with 17 related methods on four publicly available citation network datasets, and the proposed method generally outperforms most existing methods in three performance metrics. By conducting a large number of ablative experiments, we validate the effectiveness of the proposed method.

Keywords Graph clustering · Pseudo-labels · Reliable sample selection · Self-supervised learning

1 Introduction

Graph clustering with attribute features and graph structure information is a hot topic in the study of graph data. It refers to partitioning the nodes of a complete graph into various groups without the guidance of additional information. The samples in the same region have

✉ Hu Lu
luhu@ujs.edu.cn

¹ Computer Science and Communication Engineering, JiangSu University, JiangSu 212013, China

² Jiangsu Province Big Data Ubiquitous Perception and Intelligent Agricultural Application Engineering Research Center, Zhenjiang, China

higher similarity, while the samples between different regions have a relatively lower similarity. Some traditional clustering algorithms can only uncover node information or structural information of the graph, such as k-means [1], spectral clustering [2], and DeepWalk [3]. At the same time, with the growth of data volume, the computational efficiency and performance of traditional clustering are relatively low in the face of sparse, high-dimensional, non-Euclidean space data.

Clustering tasks for regularized data (e.g., in the image domain) are currently achieving impressive success by combining a deep neural network framework represented by Auto-encoder (AE) and its variants [4–6] with traditional clustering techniques, such as [7–14]. This has provided insight into graph clustering, and one of the main methods is to combine graph embedding learning with graph clustering. Firstly, graph embedding is used to reduce the dimensionality of the original data and map it to a lower dimensional space. Then, graph clustering is carried out using the embedding that extracts the discriminative information. The main graph embedding methods include graph convolutional network (GCN) [15], graph attention network (GAT) [16], and their combined product with auto-encoder, i.e., graph auto-encoder (GAE) [17] and its variants [18–24]. The training process of the above methods is two-steps, i.e., the clustering loss is independent of the optimization of the model, which will lead to sub-optimal training performance. EGAE [25] uses a relaxed k-means, which ensures the orthogonal property of the obtained embedding while allowing the clustering and reconstruction to be optimized simultaneously. However, the above approach focuses on the information obtained by model embedding and ignores the information generated by model clustering.

Recently, self-supervised learning, a new paradigm of representation learning, learns supervised information from the data itself without relying on manual labels. Two main methods that can effectively exploit cluster information: self-supervision and pseudo-supervision [26], both of which belong to unsupervised training methods. The former obtains a higher confidence auxiliary distribution by designing a pretext task and later uses it to supervise the target distribution training, such as DEC [7] and IDEC [27]. The latter guides the downstream model for semi-supervised training through the pseudo-labels obtained from clustering, such as Deep Cluster [28], IDCEC [29], and DSCNSS [30]. In addition to the above two methods, contrastive learning (CC) can achieve the purpose of self-supervised learning by constructing positive and negative sample sets to guide model training through data augmentation. However, there are relatively fewer studies based on self-supervised learning on graph data compared to other fields. Most of the methods mentioned above are based on the Euclidean distance between the embedding and the k-means cluster center to determine the confidence level of the sample, and they lack the exploration of the sample reliability mechanism of other clustering algorithms. Many self-supervised learning methods aim to learn representative features without label information. However, whether the self-supervised mechanism can improve the fusion of topological structure and node features in GCN remains to be explored.

To fully use the supervised information from graph clustering and reduce the damage of noise in this information to the accuracy of downstream models, we use pseudo-labels generated by unsupervised learning in the pretext task as supervised information to guide the partitioning of sample nodes. We propose a new self-supervised graph clustering model and elaborate the symmetric graph auto-encoder clustering model in the pretext task. In this model, a self-training module is added to the symmetric graph auto-encoder to optimize the clustering and embedding learning simultaneously. At the same time, pseudo-labels obtained by clustering are used to select samples with high confidence to train the downstream model.

Specifically, our contributions are summarized as follows:

1. We propose a new self-supervised graph clustering model in which we divide the graph clustering task into two relatively independent processes: a pretext task that performs unsupervised clustering and a downstream task that performs semi-supervised learning. In the absence of real labels, we use the pseudo-labels obtained after sufficient training of the pretext task as supervised information to guide the training of the downstream model.
2. We propose a new self-training method that does not require k-means cluster centers to further improve the accuracy of the upstream model. We also propose a reliable sample selection mechanism to reduce the negative impact of the cluster's noisy samples at the decision boundary on the model training and improve the quality of the pseudo-label as supervised information.
3. We conduct extensive experiments on four major citation network datasets. It shows that the pretext model can guide the downstream model to achieve higher performance. At the same time, compared with the unsupervised graph clustering algorithms in recent years, our DSAGC is competitive.

2 Related Work

2.1 Attribute Graph Clustering

Attribute graph clustering aims to divide nodes of a graph into disjoint regions by using node features and topology information of graph data. However, some traditional clustering methods have limited ability to extract graph information and can only extract graph node information or topology information, such as k-means [1], spectral clustering [2], and DeepWalk [3]. Thanks to the development of graph neural networks (GNNs), especially GCNs [15, 31, 32], there has been a significant breakthrough in attribute graph clustering techniques. Meanwhile, many current graph clustering algorithms combine graph embedding and graph clustering. A typical example is GAE [17] and its variants. MGAE [21] proposes to use noise to destroy node features and later induce the GAE to learn the features after marginalizing noise to improve the model's efficiency. ARGAE [18] proposes to use an adversarial training approach to force the latent representations to match a certain prior distribution. The performance of traditional GCN decreases significantly when the number of layers exceeds two. AGC [23] proposes an adaptive graph convolution method to capture the information of high-order neighborhoods by using high-order graph convolution. GALA [20] proposes a symmetric graph auto-encoder whose decoder is learnable and symmetric to the encoder. It adopts Laplacian sharpening as a convolutional filter, which speeds up the process of model reconstruction. The above models utilize graph neural networks for representation learning, and the resulting embeddings are used for subsequent clustering tasks. Since the clustering process is independent of the model optimization, the embedding learned by the model hardly guarantees that the clustering task is optimal.

2.2 Self-Supervised Clustering

Self-supervised learning can use the information carried by the data itself to guide model training without manual labels. Nowadays, self-supervised learning has achieved good applications in various fields, such as natural language processing and computer vision. DeepCluster [28] proposes to use clustering to generate pseudo-labels, which are then used

to guide the training of classifiers in an end-to-end manner. Deep subspace clustering [8, 33] maps the feature matrix to the embedding space, after which a self-expression matrix is generated, allowing nodes to be linearly represented by other nodes in the same subspace. DEC [7] generates high-confidence soft distributions by using student t-distributions and guides model training in a self-optimizing manner. DSSEC [34] builds on DEC by preserving local structure while using a stacked sparse auto-encoder to allow the model to learn more representative representations. SDCN [41] employs AE to extract attribute features and uses GNN to extract topological information, guiding the training of both modules simultaneously through a high-confidence soft distribution.

In recent years, many graph neural network models based on self-supervised methods have also emerged in the graph data domain. M3S [35] proposes a multi-stage training framework to compensate for the generalization ability of GCNs when there are fewer labels by continuously adding pseudo-labels to the labeled training set. SCRL [36] proposes to construct a feature graph by using node features and then use the shared information between the feature graph and the topology graph as the supervisory signal to guide the model's training. CGCN [37] proposes to combine an attribute graph clustering network composed of variational graph auto-encoder and Gaussian mixture model with a semi-supervised classification network. If the pseudo-label of the unsupervised model is consistent with the semi-supervised prediction, the performance of semi-supervision is improved by adding the unlabeled nodes to the labeled set. GCA [38] and GraphCL [22] apply contrastive learning to the domain of graph datasets, forming two different views through data augmentation and enhancing the consistency of different levels of object information between views through carefully designed contrastive loss. DAEGC [39] employs a graph attention network to form an auto-encoder and guides the self-training of graph datasets by constructing higher confidence distributions. IDCEC [29] proposes to screen reliable samples according to the Euclidean distance between latent representations and cluster centers and then use these samples and their pseudo-labels to guide the training of downstream models. However, most of the above approaches focus on the task of graph node classification, while the exploration of the task of graph clustering is lacking. Therefore, we intend to design a new pretext task to guide the graph clustering task in a self-supervised manner.

3 Method

In this section, we will specifically introduce our proposed DSAGC model. The structure is shown in Fig. 1. Specifically, we describe the key modules in the model, propose an auxiliary distribution based on spectral clustering and a reliable sample selection method, and discuss the training strategy using the upstream model to guide the downstream model training.

3.1 Notations

A graph is represented as $G = (V, E, X)$, where $V = \{v_1, v_2, \dots, v_n\}$ denotes the nodes set with $|V| = n$, $E = \{e_{ij}\}$ denotes the edges set. Adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $A_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $A_{ij} = 0$. Feature matrix $X = \{x_1^T, x_2^T, \dots, x_n^T\}$, where $x_i \in \mathbb{R}^d$ is a attribute vector associated with node v_i . $\hat{y}_{cluster}$ denotes the pseudo-labels generated by the upstream task, which has the simplified form \hat{y}^c . $\hat{y}_{self_supervised}$ denotes the pseudo-labels generated by the downstream task, which has the simplified form \hat{y}^s .

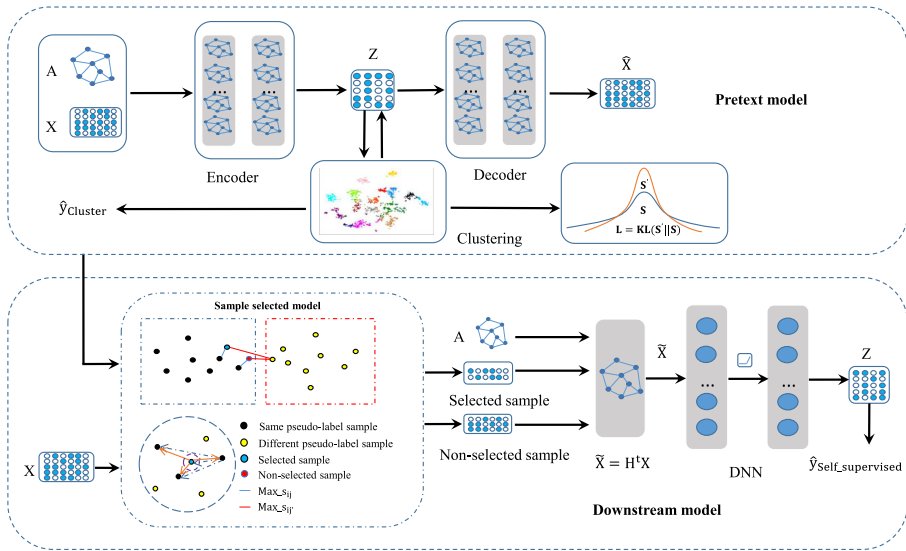


Fig. 1 The overall architecture of DSAGC. The model is mainly divided into three parts: the pretext task, the downstream task, and the reliable sample selection module. In the pretext task, the feature matrix X and adjacency matrix A are used as inputs. The self-supervised symmetric graph auto-encoder is trained by combining the self-supervised loss and reconstruction loss. After that, we use the obtained embedding matrix Z to construct a k -nearest neighbor matrix G_1 and use it to perform spectral clustering. When the pretext model is fully trained, the samples with high confidence are screened according to the information of the pseudo-labels. In the downstream task, we perform α -order generalized Laplacian smooth filter on the reliable samples and other samples and then input them into the DNN model. The pseudo-labels corresponding to the reliable samples are used as supervised information for training

3.2 Deep Self-Supervised Attribute Graph Cluster (DSAGC)

Our proposed model is divided into three main parts: a pretext symmetric self-supervised graph auto-encoder module, a downstream semi-supervised classification module, and a reliable sample selection module. In addition to the basic reconstruction loss, the pretext symmetric self-supervised graph auto-encoder module also uses the information provided by spectral clustering to generate a high-confidence distribution. Then, it guides the target distribution for training, which enables simultaneous optimization of representation learning and clustering. When the upstream model is sufficiently trained, the reliable sample selection module utilizes the obtained pseudo-labels and sets thresholds to select samples with high confidence for each cluster, reducing the influence of noisy samples at the cluster boundaries. The downstream model is a semi-supervised model. Unlike other graph self-supervised algorithms that expand the labeled set by using pseudo-labels with high confidence, it only uses the reliable samples passed in from the pretext task with the corresponding pseudo-labels for training, without a real label involved in the training.

3.2.1 Pretext Task: Self-Supervised Symmetric Graph Auto-Encoder (SSGAE)

In the pretext task, we adopt a symmetric graph auto-encoder similar to the one in GALA[20]. In the encoder part, we use a two-layer GCN network where the structure of each layer is represented as follows: $H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$, where $\tilde{A} = A + I$, $\tilde{D}_{ii} =$

$\sum_{j=1}^n \tilde{A}_{ij}$. In the decoder part, we use a two-layer Laplacian sharpening filter, where the structure of each layer is represented as follows: $H^{(l+1)} = \sigma \left(\widehat{D}^{-\frac{1}{2}} \widehat{A} \widehat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$, where $\widehat{A} = 2I - A$, $\widehat{D} = 2I + D$. At the same time, we borrow from SGC [40] and keep the activation function only in the first layer of the encoder network and the last layer of the decoder network to reduce the probability of model overfitting. The encoder part of the model makes the features of neighboring samples gradually similar, and the decoder part restores the differences between samples so that the resulting embedding obtains more sample information. The basic loss function of the model is a reconstruction of the feature matrix X , which is expressed as follows.

$$L_{re} = \min \|\widehat{X} - X\|_F^2 \tag{1}$$

Where \widehat{X} is the reconstructed feature matrix and $\|\cdot\|_F$ mainly refers to the Frobenius norm. However, it is difficult to rely solely on the reconstruction of graph data to ensure that learned embeddings are appropriate for specific downstream tasks. Some literatures [7, 27, 39, 41] have used the Student’s T-distribution to construct auxiliary distributions with higher confidence, using the additional information generated by clustering for guidance and obtaining better model performance.

The Student’s t-distribution is based on the Euclidean distance, and one of its basic assumptions is that the closer the distance of a sample to the cluster center, the higher the probability that it corresponds to this cluster. However, some combinations of graph auto-encoders and k-means are not ideal, which limits the effectiveness of this self-training method. One of our basic assumptions is that the larger the inner product distance, the more similar the samples are. Therefore, the higher the probability of belonging to the same cluster. We use the obtained embedding Z as well as the cosine similarity to construct the similarity matrix $S \in \mathbb{R}^{n \times n}$:

$$S_{ij} = \frac{Z_{i,:} Z_{j,:}^T}{\|Z_{i,:}\|_2 \|Z_{j,:}\|_2} \tag{2}$$

Here, $\|\cdot\|_2$ denotes the L2 normalization. After that, we use the label information obtained by spectral clustering to divide the similarity matrix S . We multiply the values of node pairs in the same cluster by a factor $(1 + t)$, while for node pairs that do not belong to the same cluster by a factor $(1 - t)$ to obtain the auxiliary distribution S' , where $t \in (0, 1)$ is the hyper-parameter, when its value is larger, further widens the difference between the features of samples in the same cluster and those in different clusters:

$$S'_{ij} = \begin{cases} S_{ij} \times (1 + t), & j \in C_i \\ S_{ij} \times (1 - t), & j \notin C_i \end{cases} \tag{3}$$

The auxiliary distribution S' is normalized to get:

$$S'_{ij} = \frac{S'_{ij}}{\sum_k S'_{ik}} \tag{4}$$

The model then performs self-training via KL divergence:

$$L_{se} = KL(S' \| S) = \sum_i \sum_j S'_{ij} \log \frac{S'_{ij}}{S_{ij}} \tag{5}$$

We jointly optimize the embedding of the symmetric graph auto-encoder and clustering learning by defining the overall loss function of the pretext model as:

$$L = L_{re} + \gamma L_{se} \tag{6}$$

Where L_{re} and L_{se} represent reconstruction loss and self-supervised loss, respectively, γ is the trade-off between them. Meanwhile, to avoid the instability of self-supervised optimization in the training process, we update the auxiliary distribution S' every five iterations in the experiment.

3.2.2 Reliable Sample Selection

Most existing self-supervised works focus on using pseudo-labels generated by clustering as a complement to real labels for semi-supervised tasks. One of the main reasons for this is the presence of noise in the pseudo-labels, which accumulates errors with the training process and affects the overall performance of the model. IDCEC [29] proposes to use k-means to cluster the embeddings of image data and to use the Euclidean distance between the embedded nodes and the cluster centers as a measure of sample confidence. We choose the samples whose distances are less than some fixed threshold as reliable samples. This method effectively reduces the effect of sample embedding misclassification at the clustering boundary. However, this method depends on the quality of the clustering centers and is also affected by the number of samples in each cluster. Inspired by the above ideas, we explore a new mechanism for reliable sample selection based on the spectral clustering algorithm.

One of our basic ideas is that the selected samples need to satisfy the requirement of expanding the similarity between the target node and the rest of the samples in the same cluster while narrowing the similarity between it and the samples in different clusters.

Firstly, we use the pseudo-labels divide the samples into k disjoint clusters, $C_j, j = 1, \dots, k$. Also, we use the similarity matrix S in Eq. 5 to select samples that belong to the same cluster C_j and sum the similarity of each of these target nodes u with the remaining nodes v to obtain a vector of $|C_j|$:

$$D_{C_j} = \{d_1, d_2, \dots, d_{|C_j|}\}, d_u = \sum_{v \in C_j} s_{uv} \tag{7}$$

It measures the importance of the similarity of sample u in cluster C_j . After that, we select the samples corresponding to the top $k_1\%$ points with the largest values among them as the reliable samples $sample_1$ corresponding to cluster C_j .

In addition, we add an interval constraint ζ between the similarity of samples in the same cluster and the similarity of samples in different clusters in order to filter out the noisy samples near the decision boundary. We subtract the maximum similarity s_{ua} of the target sample u in the same cluster and the maximum similarity s_{ub} in different clusters.

$$\zeta_u = s_{ua} - s_{ub}, \begin{cases} s_{ua} = \max s_{ui}, & i \in C_j \\ s_{ub} = \max s_{uj}, & j \notin C_j \end{cases} \tag{8}$$

Considering that the similarity gap between same-cluster samples and opposite-cluster samples varies by cluster class, we decide to sort the ζ_u of each cluster from largest to smallest and select the $k_2\%$ -th of them as the interval of each cluster m_j . Afterwards, for convenience, we let the minimum value of m_j be the uniform interval m . If the resulting difference is greater than or equal to the set interval m , we make it a reliable sample $sample_2$. Ultimately, only

samples that satisfy both of the previous two conditions can be selected as reliable samples *sample*, i.e.:

$$sample = sample_1 \cap sample_2 \quad (9)$$

After selecting reliable samples, we use them and their corresponding pseudo-labels as a training set to guide the downstream classification task in a semi-supervised way.

3.2.3 Downstream Task

In most recent works, the pseudo-label is only used as an expanded supervised information to improve the performance of semi-supervised classification tasks. However, this approach still requires the use of real labels. In this paper, we try to use only pseudo-labels and the corresponding reliable samples as supervised information for the model. Meanwhile, we introduce a Laplacian smoothing filter mentioned in the Adaptive Graph Encoder (AGE) [24] to better filter the high-frequency noise in the reliable samples and improve the performance of the downstream model. We compare the differences in performance from two different graph convolution methods in the ablation study section.

First, the feature matrix is multiplied by the α -layers graph convolution filter:

$$\tilde{X} = H^\alpha X \quad (10)$$

Here, H is a Laplacian smoothing filter followed by AGE, i.e., $H = I - \beta \tilde{L}$, where \tilde{L} represents the symmetric normalized graph Laplacian matrix and β represents the inverse of the corresponding spectral radius. The graph filter acts as a low-pass filter over the entire range of eigenvalues of the Laplacian matrix.

After that, we input the feature matrix \tilde{X} processed by the graph convolution filter into a two-level DNN model.

$$Y = \sigma(\sigma(\tilde{X}W_1)W_2) \quad (11)$$

Where Y represents the output of the overall downstream model, σ represents the activation function ReLU, and W_i represents the weight matrix of i -th layer. For each node v_i , the result of its corresponding prediction label is:

$$\hat{y}_i^s = \arg \min_u Y_{iu} \quad (12)$$

The loss function of the downstream model is

$$L = - \sum_{i \in sample} \hat{y}_i^c \ln \hat{y}_i^s \quad (13)$$

Where we use only the samples from the reliable sample set *sample* as the training set. The pseudo-labels \hat{y}^c corresponding to these samples generated by the upstream model are used as supervisory information, and the prediction labels \hat{y}^s generated by the downstream model are used as training targets.

The proposed self-supervised attributed graph clustering and pretext task algorithms are described in Algorithm 2 and Algorithm 1, respectively.

Algorithm 1 Self-supervised Symmetric Graph Auto-Encoder (SSGAE)

Input: Graph G with n nodes; Number of cluster k ; Auxiliary distribution update interval T ; Clustering Coefficient γ ; Number of iterations $Iter_{max}$.

Output: Pseudo-Labels $\hat{y}_{cluster}$.

- 1: Update AutoEncoder’s weight by minimizing Eq. 1 to get hidden embedding Z .
- 2: Construct cosine distance matrix S and k-nearest neighborhood graph G_1 by Z .
- 3: Use G_1 for spectral clustering to get initial $\hat{y}_{cluster}$.
- 4: **while** $i < Iter_{max}$ **do**
- 5: Update S and G_1 by Z and update $\hat{y}_{cluster}$ by spectral clustering of G_1 .
- 6: **if** $i \% T == 0$ **then**
- 7: Make Auxiliary distribution S' by Eq. 3 and Eq. 4.
- 8: **else**
- 9: Calculate clustering loss according to Eq. 5 and update the pretext framework by minimizing Eq. 6.
- 10: **end if**
- 11: **end while**

Algorithm 2 Deep Self-supervised Attributed Graph Clustering (DSAGC)

Input: graph G with n nodes; Number of cluster k ; Sample select threshold k_1, k_2 ; Number of iterations $Iter_{max}$.

Output: Pseudo-Labels $\hat{y}_{self_supervised}$.

- 1: Use SSGAE to get embedding Z , pseudo-label $\hat{y}_{cluster}$ and construct cosine distance matrix S by Z .
- 2: Under the guidance of $\hat{y}_{cluster}$, divide the node of S into different clusters.
- 3: Use the sample select threshold for selecting the reliable sample.
- 4: Use a α -order general Laplacian filter to deal with the input samples X of the downstream model and get \tilde{X} .
- 5: **while** $iter \leq Iter_{max}$ **do**
- 6: Train Deep Neural Network using processed samples \tilde{X} .
- 7: **end while**

4 Experiments

4.1 Datasets

To validate the effectiveness of our proposed DSAGC method on the node clustering task, we conduct extensive experiments on four benchmark datasets. These four datasets are all widely used citation network datasets, including Cora, Citeseer, Wiki, and Pubmed. The details of each dataset are summarized in Table 1.

4.2 Experiment Settings

In the pretext task, we adopt a symmetric graph auto-encoder as the basis and use a high-confidence distribution based on spectral clustering as the constraint term to make the model learn a representation that is more favorable to clustering. We follow GALA to construct the

Table 1 Summary of the datasets

Datasets	Nodes	Features	Edge	Classes
Cora	2708	1433	5429	7
Citeseer	3312	3703	4732	6
Pubmed	19,717	500	44,338	3
Wiki	2405	4973	17,981	17

similarity matrix using KNN and embedding in the embedding space and use it later in the spectral clustering task, where the values of KNN are set to 20, 100, 20, 800, and the learning rates are set to 0.005, 0.005, 0.001, 0.001, respectively. We also adopt the Adam optimizer and use ReLU as the activation function at the input and output layers of the symmetric graph auto-encoder. In the training process of the pretext task, we first obtain the model parameters corresponding to the optimal accuracy of the pre-trained model and then add the loss function of the self-supervised term during the training process. Among them, for the convenience of training, we uniformly set the values of the coefficient of the self-supervised term γ in the overall loss function, t in the auxiliary distribution S' , and the update interval T in the four datasets to 5, 0.5, and 5, respectively.

Similarly, we save the model parameters corresponding to the clustering accuracy of the upstream model at convergence and later use the pseudo-labels and embeddings obtained in this condition to select reliable samples. The threshold k_1 is set to 0.3, and the threshold k_2 is set to 0.4 for reliable sample selection. In the downstream task, we input all samples into a 2-layer DNN model. Where the reliable samples selected by the upstream model are used as the training set, and their corresponding pseudo-labels are used as supervisory information. At the same time, before the samples are input into the model, the α -order general Laplacian matrix is used to process the samples. The specific parameter settings are consistent with those in [24]. The Adam optimizer is chosen for the model, the learning rate is set to 0.005 for all datasets, and the activation function is ReLU. All codes are implemented by pytorch—1.7.0 on Windows 10.

4.3 Baselines and Evaluation Metrics

In the node clustering task, we mainly use three performance metrics to measure the model's performance, i.e., ACC, NMI, and ARI. We compare the proposed method DSAGC with two methods: the clustering algorithm that only uses node features or graph structure information and the graph clustering algorithm based on deep learning. Algorithms that use only features or structures for clustering include: k-means [1], spectral clustering (SC) [2], Graph-Encoder [42], DeepWalk [3], DNGR [43], DEC [7], and TADW [44]. The deep graph clustering algorithms include: Graph Autoencoder (GAE), Variational Graph Autoencoder (VGAE) [17], Adversarial Regularized Graph Autoencoder (ARGE), Adversarial Regularized Variational Graph Autoencoder (ARVGE) [18], Deep Attentional Embedded Graph Clustering (DAEGC) [39], Embedding Graph Autoencoder (EGAE) [25], Adaptive Graph Convolution (AGC) [23], Graph convolutional Autoencoder using LAplacian smoothing and sharpening (GALA) [20], Graph Clustering via Variational Graph Embedding (GC-VGE) [45], Structural Deep Clustering Network (SDCN) [41].

The specific performance of each comparison experiment on the four datasets is shown in Table 2. The data in bold in the table indicates the best performance tested under this metric. The source of the data is mainly from the experimental results declared in the paper of the corresponding method. Some methods do not test these datasets in their original text, and we will cite results tested in other papers. The upstream model adopts a symmetric graph auto-encoder similar to GALA, but its performance lags behind the latter by 1% to 2%, but with the addition of a self-supervised term, its effect is reversed in some datasets. We select reliable samples based on the optimal performance of the upstream model so that they serve as the only supervised information to guide the downstream model. In this case, we run the downstream model 10 times and take the average value as the final result. Compared to our upstream and downstream task models, the accuracy is essentially improved in all three

Table 2 Experimental results of node clustering on Cora Dataset

Metric	ACC	NMI	ARI
k-means	50.0	31.7	23.9
Spectral clustering	39.8	29.7	17.4
Graph encoder	30.1	5.9	4.6
DeepWalk	52.9	38.4	29.1
DNGR	41.9	31.8	14.2
DEC	46.5	23.5	15.1
TADW	53.6	36.6	24.0
GAE	53.0	39.7	29.3
VGAE	59.2	40.8	34.7
ARGE	64.0	44.9	35.2
ARVGE	63.8	45.0	37.4
DAEGC	70.4	52.8	49.6
GC-VGE	70.7	53.6	48.2
SDCN	35.6	14.3	7.8
AGC	68.9	53.7	48.6
EGAE	72.4	54.0	47.2
GALA	74.6	57.7	53.2
SSGAE	75.2	56.6	54.8
DSAGC	77.1	58.7	57.2

performance metrics. This implies the feasibility of relying solely on reliable samples to guide the semi-supervised training of the downstream model. Also, our method can achieve very competitive results by comparing it with other methods. In particular, the ACC of the Wiki dataset has increased by nearly 6%, and its ARI index has also increased by 7%, while the other three datasets have also improved by 1.5% to 2%. At the same time, we also see that not all performance metrics achieve optimal results on the Pubmed dataset. One of the main reasons is that spectral clustering can have difficulty guaranteeing its speed and accuracy when dealing with graph datasets with a very large number of nodes, which is also an aspect worth studying next.

4.4 Ablation Study

In this section, we test three main aspects: (1) the effect of using activation functions in different layers on symmetric graph auto-encoder. (2) the effect of different self-optimizing methods on the performance of symmetric graph auto-encoder. (3) the effect of the selection of the downstream model on the final accuracy.

We conduct experiments on the use of activation functions in the upstream model and find that using activation functions only in the first and last layers in the symmetric graph auto-encoder can achieve better results. The experimental results are shown in Table 6. We hypothesize that using activation functions in the hidden and embedded layers with lower dimensionality of the graph symmetric auto-encoder leads to loss of some information, which affects the results of the feature matrix X reconstruction.

We compare the impact of using the currently popular Student T distribution and our proposed method as the self-optimizing term on the performance of the upstream symmetric

Table 3 Experimental results of node clustering on Citeseer Dataset

Metric	ACC	NMI	ARI
k-means	54.4	31.2	28.5
Spectral clustering	30.8	9.0	8.2
Graph encoder	29.3	5.7	4.3
DeepWalk	33.7	8.9	9.2
DNGR	32.6	18.0	4.3
DEC	55.9	28.3	28.1
TADW	45.5	29.1	22.8
GAE	45.6	22.1	19.1
VGAE	46.7	26.1	20.6
ARGE	57.3	35.0	34.1
ARVGE	54.4	26.1	24.5
DAEGC	67.2	39.7	41.0
GC-VGE	66.6	40.9	41.5
SDCN	66.0	38.7	40.2
AGC	67.0	41.1	41.9
EGAE	67.4	41.2	43.2
GALA	69.3	44.1	44.6
SSGAE	71.2	45.2	46.9
DSAGC	72.7	44.9	47.3

Table 4 Experimental results of node clustering on Wiki Dataset

Metric	ACC	NMI	ARI
k-means	40.4	42.9	15.0
Spectral clustering	22.0	18.2	1.5
Graph encoder	20.7	12.1	0.5
DeepWalk	38.5	32.4	17.3
DNGR	37.6	35.9	18.0
DEC	40.0	41.1	25.6
TADW	31.0	27.1	4.5
GAE	37.9	34.5	18.9
VGAE	45.1	46.8	26.3
ARGE	38.1	34.5	26.3
ARVGE	38.7	33.9	10.7
DAEGC	48.2	44.8	33.1
GC-VGE	48.8	47.6	28.4
SDCN	44.3	42.0	28.8
AGC	47.7	45.3	34.3
EGAE	51.5	48.0	33.1
GALA	54.5	50.4	38.9
SSGAE	57.1	52.2	35.3
DSAGC	63.4	55.7	42.5

Table 5 Experimental results of node clustering on Pubmed Dataset

Metric	ACC	NMI	ARI
k-means	59.5	31.5	28.1
Spectral clustering	52.8	9.7	6.2
Graph encoder	53.1	21.0	18.4
DeepWalk	54.3	10.2	8.8
DNGR	46.8	15.3	5.9
DEC	60.1	22.4	19.6
TADW	51.1	24.4	21.7
GAE	63.2	24.9	24.6
VGAE	61.9	21.6	20.1
ARGE	68.1	27.6	29.1
ARVGE	51.3	11.7	7.8
DAEGC	67.1	26.6	27.8
GC-VGE	68.2	29.7	29.8
SDCN	64.2	22.9	22.3
AGC	69.8	31.6	31.9
EGAE	70.6	32.0	33.0
GALA	69.4	32.7	32.1
SSGAE	69.0	28.8	29.5
DSAGC	70.7	31.2	32.6

Table 6 The effect of using activation functions in different layers on symmetric graph auto-encoder

Activation layers	Citeseer	Cora	Pubmed	Wiki
$L_1 + L_2 + L_3 + L_4$	66.09 ± 0.85	64.08 ± 3.05	61.23 ± 2.07	52.72 ± 1.00
$L_1 + L_4$	67.98 ± 0.71	71.92 ± 1.25	65.55 ± 2.04	55.94 ± 1.05

graph auto-encoder. The former uses a combination of symmetric graph auto-encoder + k-means + self-optimizing method based on the Student T distribution (collectively referred to as combination I later). The latter uses a combination of symmetric graph auto-encoder + spectral clustering with KNN composition + the proposed method (collectively referred to as combination II later). The results are shown in Fig. 2, where the cherry red line shows combination II, the orange line shows combination I, the dashed line shows the result without adding the self-optimizing term, and the solid line shows the result with adding the self-optimizing term. We compare the performance of the two combinations with self-optimizing term coefficients from 1 to 10. Since the gradient of the model under the Pubmed dataset is smaller for combination I, we use the coefficients with a larger span. From the results in the figure, it can be seen that Combination II has outperformed Combination I without the addition of the self-optimizing term. In addition to the Citeseer dataset, the performance of combination I is sensitive to the coefficients of the self-optimizing terms, which does not completely ensure that the performance will be improved, especially for the Pubmed dataset, but the performance will be decreased. We conjecture that the dataset has many data samples, and it is difficult to obtain effective soft-label information just by using the Euclidean distance between the embedding points and the clustering centers when the clustering information is

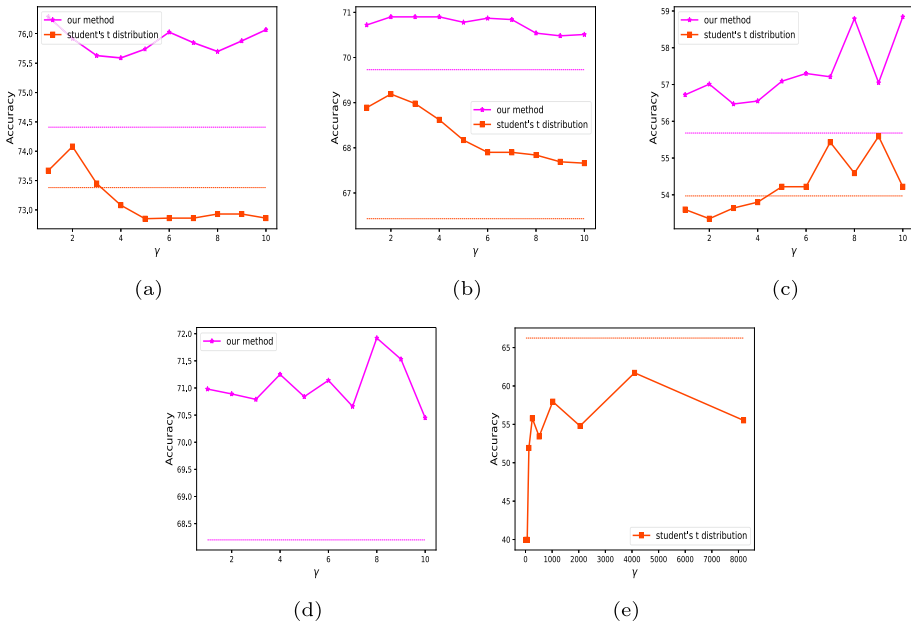


Fig. 2 The effect of different self-optimization methods on the performance of symmetric graph auto-encoder. **a** Cora. **b** Citeseer. **c** Wiki. **d, e** Pubmed

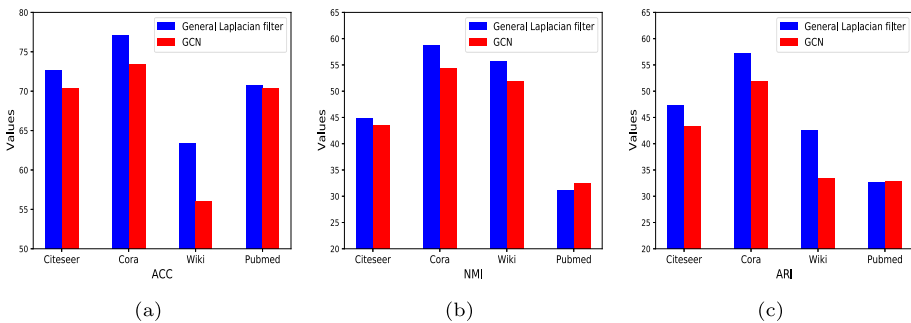


Fig. 3 The overall model accuracy of different downstream models. **a** ACC. **b** NMI. **c** ARI

not particularly reliable. The effectiveness of the method is demonstrated by the effective performance improvement on the upstream models for all four datasets.

In terms of downstream model selection, we compare two different forms of graph convolution schemes: one is a traditional two-layer GCN structure, and the other uses a form of filter that is separated from the power matrix (i.e., a combined form of α -nd power of the filter and DNN). The latter uses a general form of Laplacian smoothing filter, which takes into account the spectral radius of different datasets and is able to achieve better low-pass filtering results. We control for the same model structure and input samples. As shown in Fig. 3, the performance of the latter is slightly stronger than that of the former, which also shows to some extent that the winding of the filter and weight matrix does affect the performance of the model.

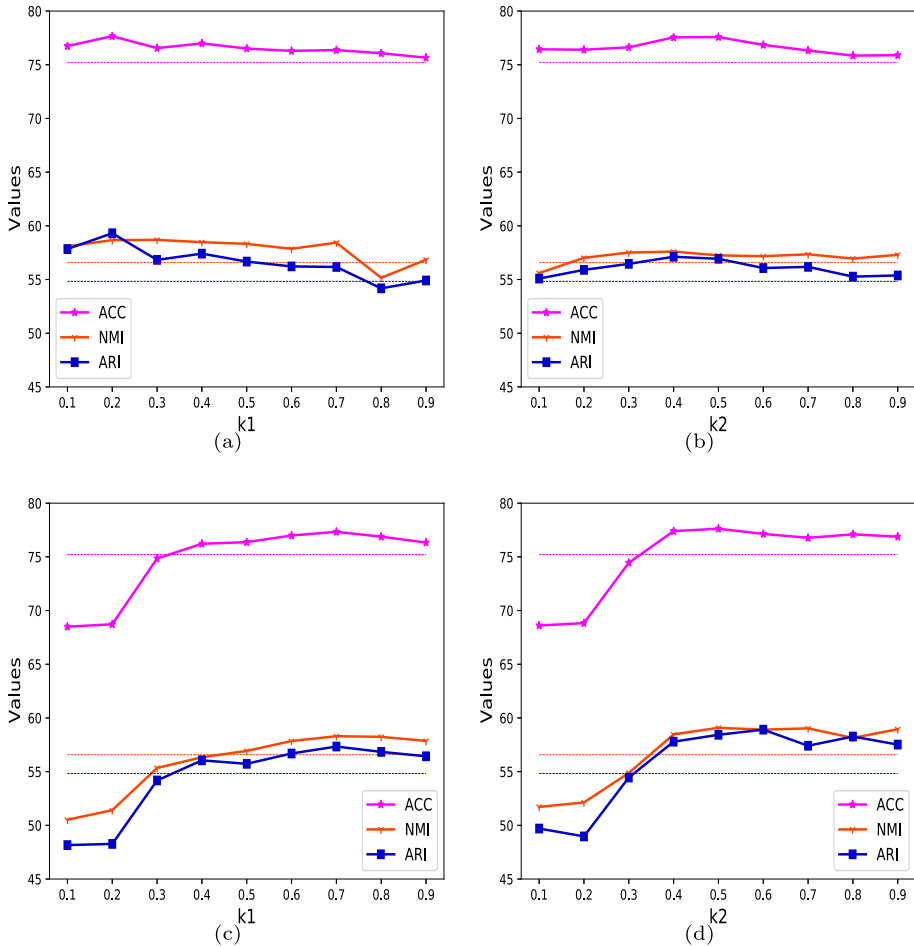


Fig. 4 The effect of sample selection coefficients k_1 and k_2 in the downstream model. a Only k_1 . b Only k_2 . c $k_2 = 0.3$. d $k_1 = 0.3$

4.5 Parameters Analysis

In this section, we will mainly analyze the influence of some hyper-parameters in the model.

4.5.1 Sample Selection Coefficients k_1 and k_2 .

In this section, to verify the effectiveness of the reliable sample selection mechanism, we test the effect of two reliable sample thresholds k_1 and k_2 on the downstream accuracy of the model using the Cora dataset as an example. Among them, Fig. 4a shows the effect of the value of k_1 on the downstream model when only k_1 is used as the reliable sample selection mechanism. Figure 4b shows the effect of the value of k_2 on the downstream model when only k_2 is used as the reliable sample selection mechanism. The three dashed lines in the figure indicate the performance of the upstream model under the three metrics as a reference for comparison. It is clear that both of the reliable sample selection mechanisms we employ

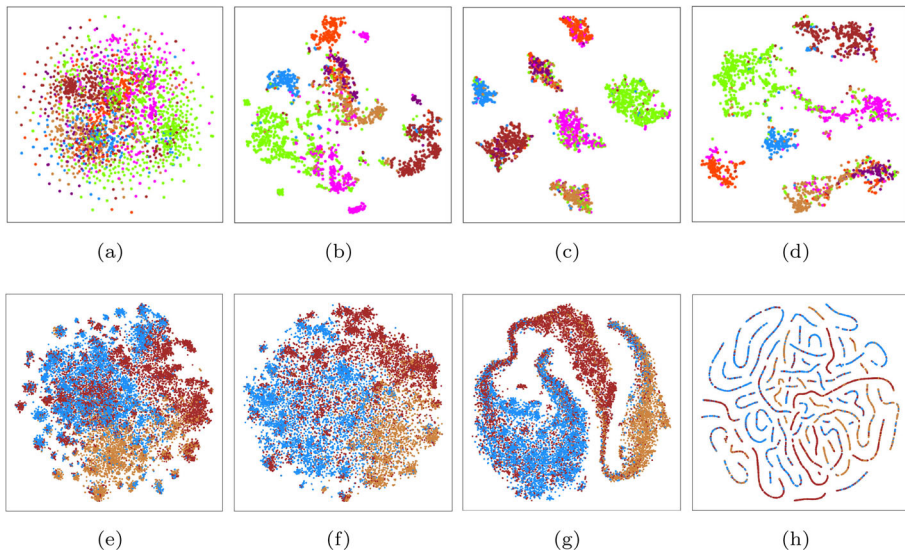


Fig. 5 2D t-SNE visualization of DSAGC on Cora and Pubmed. Where each column represents original, only reconstruction, pretext task, and downstream task respectively, i.e., the data distribution of the four phases of the model training. **a–d** Cora. **e–h** Pubmed

can greatly improve the quality of pseudo-labels as supervisory information. When the value of k_1 is small, we obtain samples with higher confidence, but the number of samples is small, so the overall generalization performance is poor. Since we select the samples according to percentages, we can ensure that the number of samples selected for each cluster class is relatively balanced. The overall model performance does not fluctuate significantly with changes of k_1 and k_2 values. After that, as the threshold value keeps increasing, the number of samples and reliability reach a balance point, achieving a better result. At the same time, we test the combination of two reliable samples. In particular, Fig. 4c shows the effect of changing the value of k_1 on the downstream model when setting $k_2 = 0.3$. Figure 4d shows the effect of varying the value of k_1 on the downstream model when setting $k_2 = 0.3$. We can see that when k_1 or k_2 is less than or equal to 0.3, the performance of the model is generally poor and even lower than the accuracy of the upstream model. The main reason for this is that there are not enough sample points to satisfy both conditions at the same time, so the performance of the model is seriously affected. As the reliable sample threshold increases, the effects of sample size and sample reliability on the model are balanced, and the downstream model outperforms the upstream model, reaching the goal of using pseudo-labels to guide the training of the downstream model.

4.5.2 Visualization Analysis

To more intuitively reflect the impact of the two self-supervised terms on the data distribution at each stage of model training, we perform t-SNE visualization on two datasets. As shown in Fig. 5, it is observed that the model can perform an initial clustering of the samples when guided only by the reconstruction loss. However, the distances between the different clusters are still very close together. In contrast, when the clustering information is used to construct the trustworthy distribution to guide the training, the distance between the clusters is

progressively expanded, and the same cluster samples move closer to each other. Through the reliable sample selection mechanism, we can further weaken the negative impact of cluster boundary samples and improve the performance of the downstream model.

5 Conclusion

In this paper, we propose a new deep self-supervised attributed graph clustering framework for social network datasets analysis. The model uses the pseudo-label information generated by clustering to construct a high-confidence distribution based on spectral clustering in the pretext task, which guides the model to learn embeddings that satisfy the specific clustering task. We further use the pseudo-label information to select reliable samples to assist in the training of the downstream model. Under the guidance of the pseudo-label, we use the cosine similarity between embedding to select the samples most similar to the other samples in the same cluster as the reliable samples. We also add a margin between the similarity of the same cluster samples and opposite cluster samples to improve its quality. We evaluate the proposed DSAGC on four popular benchmark datasets. The experimental results show the effectiveness of the proposed model on the node clustering task.

Data Availability The datasets generated during or analyzed during the current study are available from the corresponding author on reasonable request.

Code available Code is available at <https://github.com/hulu88/NPL-DSAGC>.

Declarations

Conflict of interest The authors declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

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

References

1. Likas A, Vlassis N, Verbeek JJ (2003) The global k-means clustering algorithm. *Pattern Recogn* 36(2):451–461
2. Ng A, Jordan M, Weiss Y (2001) On spectral clustering: analysis and an algorithm. *Adv Neural Inf Process Syst* 14:849–856
3. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pp 701–710
4. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint [arXiv:1312.6114](https://arxiv.org/abs/1312.6114)
5. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A, Bottou L (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J Machine Learn Res* 11(12):3371–3408
6. Masci J, Meier U, Cireşan D, Schmidhuber J (2011) Stacked convolutional auto-encoders for hierarchical feature extraction. In: *International conference on artificial neural networks*. Springer, pp 52–59

7. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: International conference on machine learning. PMLR, pp 478–487
8. Peng X, Feng J, Zhou JT, Lei Y, Yan S (2020) Deep subspace clustering. *IEEE Trans Neural Netw Learn Syst* 99, 1–13
9. Lu H, Liu S, Wei H, Tu J (2020) Multi-kernel fuzzy clustering based on auto-encoder for fMRI functional networks. *Exp Syst Appl* 159:113513
10. Guo W, Cai J, Wang S (2020) Unsupervised discriminative feature representation via adversarial auto-encoder. *Appl Intell* 50:1155–1171
11. Wang S, Cai J, Lin Q, Guo W (2019) An overview of unsupervised deep feature representation for text categorization. *IEEE Trans Comput Social Syst* 6(3):504–517. <https://doi.org/10.1109/TCSS.2019.2910599>
12. Cai J, Wang S, Xu C, Guo W (2022) Unsupervised deep clustering via contractive feature representation and focal loss. *Pattern Recogn* 123:108386
13. Lu H, Jin T, Wei H, Nappi M, Li H, Wan S (2023) Soft-orthogonal constrained dual-stream encoder with self-supervised clustering network for brain functional connectivity data. *Expert Syst Appl* 244:122898
14. He Z, Wan S, Zappatore M, Lu H (2023) A similarity matrix low-rank approximation and inconsistency separation fusion approach for multi-view clustering. *IEEE Trans Artif Intel* 5:868–881
15. Kipf TN, Welling M (2016) semi-supervised classification with graph convolutional network. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907)
16. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv preprint [arXiv:1710.10903](https://arxiv.org/abs/1710.10903)
17. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv preprint [arXiv:1611.07308](https://arxiv.org/abs/1611.07308)
18. Pan S, Hu R, Long G, Jiang J, Yao L, Zhang C (2018) Adversarially regularized graph autoencoder for graph embedding. arXiv preprint [arXiv:1802.04407](https://arxiv.org/abs/1802.04407)
19. Chen C, Lu H, Hong H, Wang H, Wan S (2023) Deep self-supervised graph attention convolution autoencoder for networks clustering. *IEEE Trans Consum Electron*. <https://doi.org/10.1109/TCE.2023.3279836>
20. Park J, Lee M, Chang HJ, Lee K, Choi JY (2019) Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 6519–6528
21. Wang C, Pan S, Long G, Zhu X, Jiang J (2017) MGAE: Marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 889–898
22. You Y, Chen T, Sui Y, Chen T, Wang Z, Shen Y (2020) Graph contrastive learning with augmentations. *Adv Neural Inf Process Syst* 33:5812–5823
23. Zhang X, Liu H, Li Q, Wu X-M (2019) Attributed graph clustering via adaptive graph convolution. arXiv preprint [arXiv:1906.01210](https://arxiv.org/abs/1906.01210)
24. Cui G, Zhou J, Yang C, Liu Z (2020) Adaptive graph encoder for attributed graph embedding. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining, pp 976–985
25. Zhang H, Li P, Zhang R, Li X (2022) Embedding graph auto-encoder for graph clustering. *IEEE Trans Neural Netw Learn Syst* 34:9352–9362
26. Lv J, Kang Z, Lu X, Xu Z (2021) Pseudo-supervised deep subspace clustering. *IEEE Trans Image Process* 30:5252–5263
27. Guo X, Gao L, Liu X, Yin J (2017) Improved deep embedded clustering with local structure preservation. In: *Ijcai*, pp 1753–1759
28. Caron M, Bojanowski P, Joulin A, Douze M (2018) Deep clustering for unsupervised learning of visual features. In: Proceedings of the European conference on computer vision (ECCV), pp 132–149
29. Lu H, Chen C, Wei H, Ma Z, Jiang K, Wang Y (2022) Improved deep convolutional embedded clustering with re-selectable sample training. *Pattern Recogn* 127:108611
30. Chen C, Lu H, Wei H, Geng X (2022) Deep subspace image clustering network with self-expression and self-supervision. *Appl Intell* 53:4859–4873
31. Hammond DK, Vandergheynst P, Gribonval R (2011) Wavelets on graphs via spectral graph theory. *Appl Comput Harmon Anal* 30(2):129–150
32. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. *Adv Neural Inf Process Syst* 29:3844–3852
33. Cai J, Fan J, Guo W, Wang S, Zhang Y, Zhang Z (2022) Efficient deep embedded subspace clustering. In: 2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 21–30. <https://doi.org/10.1109/CVPR52688.2022.00012>
34. Cai J, Wang S, Guo W (2021) Unsupervised embedded feature learning for deep clustering with stacked sparse auto-encoder. *Expert Syst Appl* 186:115729

35. Sun K, Lin Z, Zhu Z (2020) Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In: Proceedings of the AAAI conference on artificial intelligence, 34, pp 5892–5899
36. Liu C, Wen L, Kang Z, Luo G, Tian L (2021) Self-supervised consensus representation learning for attributed graph. In: Proceedings of the 29th ACM international conference on multimedia, pp 2654–2662
37. Hui B, Zhu P, Hu Q (2020) Collaborative graph convolutional networks: unsupervised learning meets semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence, 34, pp 4215–4222
38. Zhu Y, Xu Y, Yu F, Liu Q, Wu S, Wang L (2021) Graph contrastive learning with adaptive augmentation. In: WWW '21: the Web conference 2021
39. Wang C, Pan S, Hu R, Long G, Jiang J, Zhang C (2019) Attributed graph clustering: a deep attentional embedding approach. arXiv preprint [arXiv:1906.06532](https://arxiv.org/abs/1906.06532)
40. Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: International conference on machine learning. PMLR, pp 6861–6871
41. Bo D, Wang X, Shi C, Zhu M, Lu E, Cui P (2020) Structural deep clustering network. In: Proceedings of the Web conference 2020, pp 1400–1410
42. Tian F, Gao B, Cui Q, Chen E, Liu T-Y (2014) Learning deep representations for graph clustering. In: Proceedings of the AAAI conference on artificial intelligence, 28
43. Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: Proceedings of the AAAI conference on artificial intelligence, 30
44. Yang C, Liu Z, Zhao D, Sun M, Chang E (2015) Network representation learning with rich text information. In: 24th international joint conference on artificial intelligence
45. Guo L, Dai Q (2022) Graph clustering via variational graph embedding. *Pattern Recogn* 122:108334

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