Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **7**, Number **4**, November **2022**

Similarity-based Attention Embedding Approach for Attributed Graph Clustering

Wei Weng

College of Computer and Information Engineering
Xiamen University of Technology
Fujian Key Laboratory of Pattern Recognition and Image Understanding
Xiamen, 361024, P. R. China
xmutwei@163.com

Tong Li

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
2022031405@stu.xmut.edu.cn

Jian-Chao Liao

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
2022031482@stu.xmut.edu.cn

Feng Guo

Fujian Newland Auto-ID Tech. Co., Ltd.
Fuzhou, 350015, P. R. China
guo.feng@nlscan.com

Fen Chen

Xiamen Fuyun Information Tech. Co., Ltd.
Xiamen, 361008, P. R. China
chenf@safedog.cn

Bo-Wen Wei*

College of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, P. R. China
wbwjohn@163.com

*Corresponding author: Bo-Wen Wei

Received June 7, 2022, revised July 11, 2022, accepted September 26, 2022.

Abstract

Graph clustering is a fundamental method for studying complex networks. Some existing approaches focus on the graph data without attributed information. However, graph data in the real world generally have attribute information, which can better describe complex networks. Recently, deep learning-based methods are widely used for attributed graph clustering. DAEGC is a goal-directed method, which utilizes graph attention encoders for latent information learning and performs competitively in attributed graph clustering. To better aggregate the structure and node attribute information in DAEGC, we employ a lightweight attention mechanism to capture the relationship between nodes and their neighbors, learn hidden representations, and use a self-training module to train the model for optimal results. Extensive

experiments show that our proposed method improves DAEGC and achieves superior performance compared with state-of-the-art algorithms in attributed graph clustering.
Keywords: Attributed graph clustering, Graph embedding, Attention mechanism

1. **Introduction.** Graph data is a kind of data representation which contains a set of nodes and edges, and it exists widely in real-world applications, such as social networks [1], academic citation networks [2], biological protein networks [3] and so on. In real-world graph data, nodes represent basic units, and edges represent the relationship between them. The distribution of fog node data is similar to the graph data structure [4]. Attributed graph is a kind of graph where each node is associated with a set of features, and the additional information is helpful to represent the graph. There are many downstream tasks to which graph data can be applied, such as node classification [5, 6], clustering [7, 8] and link prediction [9].

The goal of clustering task is to group the nodes into disjoint sets. Traditional machine learning methods address attributed graph clustering by random walks [10], nonnegative matrix factorization [11], and so on. Benefit from the breakthroughs in Convolutional Neural Networks (CNNs) [12], computer vision [13], Transportation Mode Detection(TMD) [14] and other fields have achieved remarkable progress in recent years. For example, in human motion recognition technology, CNNs can be used to process human motion data with spatio-temporal continuity [15]. The model based on CNNs can also be applied to the stock price trend prediction task [16, 17]. Although traditional deep learning methods have achieved great success in extracting features from Euclidean data (e.g., images and texts), they cannot be directly applied to graph data. The reason is that graph data is a kind of non-Euclidean data, which is irregular, and each node has a different number of neighbors. Inspired by convolutional neural networks, recurrent neural networks and deep autoencoders, researchers propose a neural network structure for graph data, namely graph neural networks (GNNs) [18].

GNNs achieves great success on graph data. Graph clustering methods based on GNNs can be roughly divided into two categories: one is to learn the node embeddings, the other is the graph pooling method [19]. For example, the Graph Autoencoder (GAE) [20] model is an unsupervised GCN-based framework suitable for learning graph node embeddings. It uses the encoder to learn a latent representation, and then reconstructs the original graph. However, the GAE model does not consider the importance of neighbors in learning node representation. To better integrate graph structure and node attribute information, DAEGC [21] adds a self-training module and attention mechanism to the GAE model. It combines graph topology and node attribute information to learn more representative node representations, and uses an attention mechanism to calculate the attention coefficient of neighbors for aggregation. The learned latent representation has a great influence on the accuracy of clustering results.

To better aggregate the structure and node attribute information, we propose an improvement version of DAEGC for attributed graph clustering in this paper. Motivated by the method AGNN [22], we use a two-layer attention network to calculate the attention coefficients. Specifically, it adopts a similarity-based attention mechanism, which uses the cosine function to calculate the similarity between node and its neighbors. The calculation is lightweight and has fewer parameters than GAT. Firstly, less parameters make the model understandable. Secondly, using the similarity between nodes and their neighbors as the attention coefficient is closer to the group status in the real world. The intuition is that nodes within the same cluster with similar feature representations are easier to cluster [23]. To this end, using this method to calculate attention coefficient can make nodes with similar attribute information tend to cluster into the same cluster, because the attention coefficient value between similar nodes is larger. For better clustering, a self-training component combined with KL divergence is used to supervise the embeddings learning. We conduct experiments on three publicly-available datasets, namely

Cora, Citeseer and Pubmed to evaluate the performance of our proposed method. The experimental results show that our model improves DAEGC and achieves a competitive performance compared with state-of-the-art algorithms on three evaluation metrics.

2. **Related Works.**

In this section, we introduce graph clustering, graph autoencoders and attention mechanisms commonly used in graph clustering tasks.

2.1. **Graph clustering.**

Graph clustering methods can be roughly divided into two branches: graph structure-based clustering (PNE), or considering graph structure and node attribute information (ANE). PNE clustering methods only consider edge information between nodes. For example, [24] computes edge betweenness. Rahavan et al. [25] proposed a semi-supervised algorithm based on label propagation. The method mentioned in study [26] applies graph-based Laplacian feature mapping. Methods based on matrix factorization use non-negative matrix factorization to learn network structure features, such as [27]. The method proposed by Grover et al. [28] generates training sequences based on random walks, and it can explore different neighborhoods of nodes. Gaussian mixture model [29] realizes clustering by learning prior distribution and deriving posterior distribution. ANE algorithms consider graph structure and node features. Zhang et al. [23] proposed a method for obtaining smooth feature representation by using k-order graph convolution, and performed spectral clustering on learned features. GCN [30] is a representative graph convolution based method. The algorithm determines the node and its neighbors according to the structure of the graph, and then continuously update the feature vector of the node according to the attribute information of its neighbors. ASNE [31] is a network representation learning method combining any type of topology and attributes. AANE [32] employs matrix factorization and takes attribute information as one of the decomposed information.

2.2. **Graph autoencoder.**

Kipf and Welling presented the graph autoencoder (GAE) and graph variational autoencoder (VGAE) [20] in 2016. Since then, graph autoencoders have been widely used in many fields due to their simple structure and efficient encode ability. These methods first capture latent variables and reconstruct the original graph using graph autoencoder or graph variational autoencoder. Compared with VGAE using two-layer GCN learning node representation, MGAE [33] uses three-layer GCN, and employs marginalized denoising autoencoder when reconstructing the original graph. ANRL [34] uses a single encoder to encode node attribute information, but multiple decoders are used in the decoder part, local and global structure information can be reconstructed simultaneously. DANE [35] model has two encoders, one encoder encodes and reconstructs structural information, and the other encodes and reconstructs attribute information. ARGE [36] combines graph encoder and adversarial network. The graph autoencoder takes the graph structure and node features to learn a latent representation, and then reconstructs the graph structure from the latent representation.

2.3. **Attention mechanism.**

The attention mechanism is a brain signal processing mechanism unique to human vision. Human vision obtains the target area that needs to be focused on by quickly scanning the global image, which is commonly referred to as the focus of attention, and ignoring useless information. Attention models were first used in machine translation [37] and are now widely used in various types of tasks, such as natural language processing [38], image recognition [39], and recommender systems [40]. For instance, regions of interest can be quickly discovered from images by using the principles of visual attention mechanisms [41]. The attention mechanism is also applied in the image title technology. DenseAtt [42] applies a multilayer dense attention mechanism, which has achieved good results.

In graph clustering, GAT [43] aggregates neighbor node information through the self-attention mechanism, improves the expression ability of node features, and realizes adaptive matching of

weights of different neighbors. GCAGC [44] is an adaptive graph convolution network with attention graph clustering. CATs [45] uses a joint attention strategy to learn node representations. AGCN [46] is a novel deep clustering method, which uses a heterogeneous fusion module to dynamically fuse node attributes and topology features, and uses the attention mechanism to dynamically fuse corresponding features.

3. The Proposed Algorithm.

3.1. **Problem definition.** A non-directed graph $G = (V, E, X)$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of nodes with $|V| = n$. $E = \{e_{ij}\}$ is a set of edges, it can be represented as an adjacency matrix $A = \{a_{ij}\} \in \mathbb{R}^{n \times n}$, where $A_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $A_{ij} = 0$. X is a feature matrix of nodes. $X = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^{n \times d}$, where $x_i \in \mathbb{R}^d$ is a real-value feature vector affiliated with node v_i. For a given graph G, our goal is to divide the nodes of the graph G into *m* clusters $C = \{C_1, C_2, ..., C_m\}$. The nodes in the same clusters are similar to each other, and there is a high probability that they have the same feature values.

3.2. **Proposed method.** As discussed in section 1, we introduce the proposed method. Our method is adapted from DAEGC [21]. DAEGC develops a graph attention encoder to combine the structure and attribute information of the graph, and uses an attention mechanism to learn the hidden representation of each node. Motivated by AGNN [22], we adopt a lightweight attention mechanism to better aggregate the structure and node attribute information. Our model consists of three parts, a graph attention encoder, a self-training module and an inner product decoder, as shown in Figure 1. The graph attention encoder aggregates the information of neighbor nodes, and attention is based on similarity, which can learn similar hidden embeddings for objects related to each other. The self-training module utilizes KL divergence to generate soft labels to supervise the embedding learning. The inner product decoder reconstructs the original graph by computing the inner product of hidden representations.

3.2.1. *Graph attentional encoder.* We use a graph attention network [22] as the graph encoder. Assuming that neighbors similar to the node are more important, the relative importance of nodes can be obtained directly by calculating the similarity between nodes [47]. For example, in social networks, a user and his friends generally have common hobbies or common topics. The similarity-based attention mechanism is based on this assumption. The similarity between nodes can be calculated as the attention coefficient. Given the corresponding features X and the graph structure A, the attention coefficient can be learned by:

$$\alpha_{i,j} = \frac{e_{i,j}}{\sum_{k \in \Gamma_{v_0}} e_{i,k}}, \tag{1}$$

where $e_{i,j}$ means the correlation between node i and node j. It can be calculated as follows:

$$e_{ij} = \beta \cdot \cos(Wx_i, Wx_j). \tag{2}$$

Among them, x_i and x_j are the feature representations of node i and node j, and W is a trainable weight matrix that maps attributes of the node to the hidden space. When aggregating neighbor information, it is necessary to normalize the attention of all neighbors, and the normalized attention weight α_{ij} is the aggregation coefficient. It can be formulated as:

$$\alpha_{i,j} = \frac{exp(\beta \cdot \cos(Wx_i, Wx_j))}{\sum_{k \in \Gamma v_0} exp(\beta \cdot \cos(Wx_i, Wx_k))}, \tag{3}$$

where β is a single trainable parameter and cos represents cosine-similarity. The similarity-based attention mechanism requires few parameters for calculating the attention coefficient.

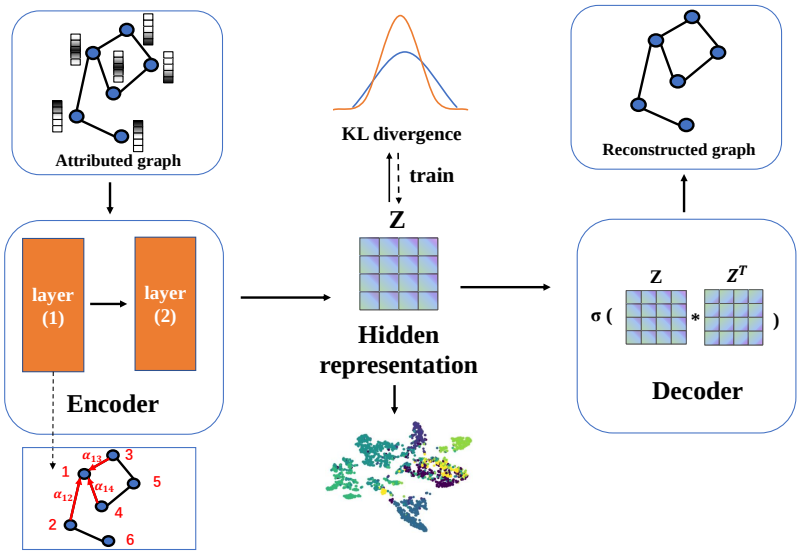


FIGURE 1. The architecture of the similarity-based attention embedding method is shown in the figure. The framework of the model is adapted from the structure of DAEGC [21]. Given the graph structure information A and the graph node attribute information X, the graph attention encoder with two layers of attention network is used to encode the graph data and learn the hidden representation Z. The method to calculate the attention coefficient here is to calculate the similarity between adjacent nodes as the attention coefficient. Next, the self-training module is performed by minimizing the KL divergence. The module reduces the loss of the objective function together with the autoencoder and performs clustering in training process.

The DAEGC model uses GAT as the attention mechanism, with an extra parameter $\vec{a}^T$. In DAEGC, to capture the graph topology information, the graph encoder adds a coefficient $M = (B + B^2 + + B^t)/t$, and B is the transition matrix of the graph. In this formula, as t increases, the computation will be increased while capturing information, and more memory will be occupied. Different from the attention mechanism in the GAT model, the similarity-based attention mechanism uses the cos function to calculate the similarity between nodes. This lightweight attention mechanism saves memory and improves computing efficiency. Similarity-based graph attentional encoder is described in Algorithm 1.

Algorithm 1 Similarity-based graph attentional encoder

Input: The feature matrix X and the structure matrix A of graph G.
Output: The learned hidden representation z of graph G.

1: Initializes the parameter matrix W.
2: Calculate e_{ij} according to Eq. 2.
3: The aggregation coefficient $\alpha_{i,j}$ is calculated by Eq. 3.
4: The hidden representation z is updated by Eq. 4.

The graph attentional encoder can learn hidden representations of nodes by assigning different weights to different neighbors:

$$z_i^{l+1} = \sigma(\sum_{j \in N_i} \alpha_{ij} \mathrm{W} z_j^l). \tag{4}$$

As shown in Eq. 4, z_i^{l+1} represents the learned information of the node i. Node j is a neighbor of the node i, and N_i denotes the neighbors of node i. α_{ij} is the attention coefficient between node i and node j. The attention coefficient α describes the importance of neighbor nodes.

We uses two graph attention layers to capture the hidden representation of node information, and let x be the initial input feature, $x_i = z_i^0$:

$$z_i^{(1)} = \sigma(\sum_{j \in N_i} \alpha_{ij} W^{(0)} x_j), \tag{5}$$

$$z_i^{(2)} = \sigma(\sum_{j \in N_i} \alpha_{ij} W^{(1)} z_j^{(1)}). \tag{6}$$

After obtaining the hidden representation of the node information, the decoder decodes the learned information, here we let $z_i = z_i^{(2)}$.

3.2.2. *Clustering via self-training embedding.* The data used in the graph clustering task is unlabeled. In other words, graph clustering tasks belong to unsupervised learning. Unsupervised tasks cannot obtain clear feedback during model training, thus self-training method is employed to solve this problem. We use hidden embedding as input data for the self-training clustering module, the objective function is formulated as:

$$L_c = KL(P \parallel Q) = \sum_i \sum_u p_{iu} \log \frac{p_{iu}}{q_{iu}}, \tag{7}$$

where q_{iu} denotes the similarity between cluster center embedding μ_μ and node embedding z_i. We use Student's t-distribution [50] to measure it, which can handle clusters of different sizes, and it is very convenient in calculation:

$$q_{iu} = \frac{(1 + \|z_i - \mu_\mu\|^2)^{-1}}{\sum_k (1 + \|z_i - \mu_\mu\|^2)^{-1}}. \tag{8}$$

The soft cluster distribution of each node can be expressed by q_{iu}, and the target distribution p_{iu} can be defined as follows:

$$p_{iu} = \frac{q_{iu}^2 / \sum_i q_{iu}}{\sum_k (q_{ik}^2 / \sum_i q_{ik})}. \tag{9}$$

Q is the probability distribution of nodes. Nodes close to the center of the cluster are soft assignments with high probability, and this part of the node distribution is considered credible. To reduce the clustering loss, the current distribution must be close to the true distribution P. So the "confidence distribution" can be used as a soft label to supervise the learning of Q. During training, we first use the graph attention encoder to get the node embedding z which is defined in Eq.7. Before training the whole model, K-means is used to cluster on the node embedding z to obtain the initial cluster centers. Then the confidence distribution Q is obtained by Eq.11.

3.2.3. *Inner product decoder.* The proposed model is trained with gradient descent using the Adam optimizer, updating the cluster centers μ and embeddings z. We use Eq.7 and Eq.9 to update the clustering loss L_c and calculate the target distribution P. There are many kinds of decoders, each of which is suitable for capturing information. Some decoders reconstruct graph structure information, and some decoders are suitable for reconstructing attribute information.

For the embedding z containing node attribute information, the inner product decoder is selected, which is more convenient and flexible when predicting the relationship between nodes. Therefore, the reconstruction information $\hat{A}_{ij}$ of the graph is obtained by the following formula:

$$\hat{A}_{ij} = sigmoid(z_i^\top z_j). \tag{10}$$

3.2.4. *Clustering optimization.* The reconstruction loss of the data after encoding and decoding is calculated by the following formula:

$$L_r = \sum_{i=1}^{n} loss(A_{ij}, \hat{A}_{ij}). \tag{11}$$

The loss function consists of two parts, the clustering loss L_c and the reconstruction loss L_r. We optimize the two modules of autoencoder embedding and clustering at the same time, and the final objective function can be formulated as follows:

$$L = L_r + \gamma L_c, \tag{12}$$

where γ is a nonnegative parameter to balance the reconstruction loss and clustering loss. The clustering result can be obtained directly from the optimized Q, and the result can be used as the prediction of node v_i:

$$s_i = arg \max_u q_{iu}. \tag{13}$$

In conclusion, we use a similarity-based attention graph encoder to capture hidden representations of nodes, and self-training clustering embedding is used to improve the clustering performance. The final objective function considers the reconstruction loss and clustering loss and carries out learning in a unified framework. The pseudocode of the method is summarized in Algorithm 2.

Algorithm 2 Similarity-based attention embedded attributed graph clustering

Input: Graph structure information matrix A, attributed information matrix X, label matrix L, the number of clusters *k*, the number of iterations ***Iter***, target distribution update interval T.
Output: Final clustering results.

```
Get the autoencoder hidden representation Z according to Algorithm 1.
Calculate the initial cluster centers μ based on Z by K-means.
for ℓ = 0 to Iter − 1 do
    if ℓ%T == 0 do
        Calculate soft assignment distribution Q by Eq.8.
        Compute target distribution P according to Eq.9.
    end if
    Calculate clustering loss L_c by Eq.7.
    Minimizing the loss L that defined by Eq.12 to update the model.
end for
Get the final clustering results according to Eq.13.
```

4. **Experiments.**

4.1. **Datasets.** We evaluated our method on three attributed networks: Cora, Citeseer, and Pubmed [2]. The above three datasets are all citation network datasets. In these datasets, the nodes correspond to publications. If there is a citation relationship between each other, the two corresponding nodes are connected by an edge. Table 1 summarizes the statistics of the datasets.

Cora: The Cora dataset contains 2708 nodes, which represents 2708 papers about machine learning and there are 5429 edges between these nodes. The labels contain seven areas: case based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning, and theory. Each node is characterized by a 1433 dimensional binary vector.

Citeseer: The Citeseer dataset includes 3327 publications with six classes, including agents, artificial intelligence, database, information retrieval, machine language, and human-computer interaction. There are 4732 edges between them, which represent the reference relationship between publications. The features of each node are represented by a 3703-dimensional binary vector.

Pubmed: The Pubmed dataset contains 19,717 scientific publications, including Diabetes Mellitus Experimental, Diabetes Mellitus Type 1, and Diabetes Mellitus Type 2. There are 44338 edges between them. Node is described by a 500-dimensional binary vector.

TABLE 1. Datasets statistics

Datasets	*Nodes*	*Edges*	*Features*	*Classes*
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19717	44338	500	3

4.2. **Comparison methods.** In the experiments, we compared our method with some typical algorithms. These comparison algorithms can be roughly divided into two types. Some methods only use node attribute information or graph structure information, and the other methods use node attribute information and graph structure information simultaneously.

K-means K-means algorithm is a basic unsupervised clustering algorithm, where k is the number of categories after clustering.

DeepWalk [48]**:** Deepwalk[1] is a representative and successful work of early network representation learning. DeepWalk learns a social representation of a network from truncated random walks. It is also a method that only considers graph structure information when learning graph representation.

GraRep [27]**:** GraRep[2] only utilizes graph structure information. The model represents the vertices in the graph by learning low-dimensional vectors and integrates the global structural information of the graph.

TADW [10]**:** TADW[3] algorithm considers the graph structure and node attribute information, which learnes hidden representations comprehensively.

AANE[4] [32]**:**This algorithm is similar to TADW, which uses matrix decomposition method for graph data. The result of matrix factorization combines the graph structure and attribute information.

ARGE & ARVGE[5] [36]**:** The method uses an adversarial graph embedding framework to process the graph data. The encoder part combines the graph structure and node attribute information. Then the decoder is used to reconstruct the graph information. The latent representation

[1] https://github.com/houchengbin/OpenANE
[2] https://github.com/houchengbin/OpenANE
[3] https://github.com/houchengbin/OpenANE
[4] https://github.com/houchengbin/OpenANE
[5] https://github.com/GRAND-Lab/ARGA

is matched with prior distribution and the reconstruction loss is reduced by adversarial training. Depending on whether additional input of variational is considered, the methods are divided into adversarially regularized graph autoencoder (ARGE) and adversarially regularized variational graph autoencoder (ARVGE).

DAEGC[6] [21]**:** The DAEGC algorithm considers graph structure and node attribute information simultaneously, combines two kinds of information in the encoder, uses the attention network to learn the importance of graph nodes, and reconstructs the graph structure in the decoder.

4.3. **Evaluation metrics and parameter settings.**

4.3.1. *Evaluation metrics.* To evaluate the proposed model, we use three commonly used metrics, including accuracy (ACC), normalized mutual information (NMI) and $F-score$ [49]. NMI measures the similarity of two clustering results. It is a normalization of the Mutual Information (MI) score to scale the results between 0 and 1. The NMI result of 0 means no mutual information, and the NMI result of 1 means complete correlation. Assume two labels U and V, U and V have the same N objects, the MI between U and V is calculated by:

$$MI(U,V)=\sum_{i=1}^{|U|}\sum_{j=1}^{|V|}\frac{|U_i\cap V_j|}{N}\log\frac{N\,|U_i\cap V_j|}{|U_i|\,|V_j|}. \tag{14}$$

The NMI is defined as:

$$NMI(U,V)=\frac{MI(U,V)}{mean(H(U)H(V))}, \tag{15}$$

where $H(U(V))=-\sum_{i=1}^{|U(V)|}P(i)\log P(i)$, $P(i)=\frac{|U(V)_i|}{N}$. For ACC and $F-score$, the larger value indicates better performance.

4.3.2. *Parameter settings.* Our method set 256 neurons of hidden layers and 16 neurons of embedding layers, the learning rate $lr=0.001$ for the Cora and Pubmed dataset. For Citeseer, the lr was 0.0001. We set the maximum number of training $max-epoch=100$. The Adam optimizer was used for optimization. For the comparison methods, including Deepwalk, TADW, GraRep, and AANE, the K-means algorithm was used to conduct the clustering task. We selected the parameters for each method according to the original paper. For Deepwalk, we set walks of each node $r=10$, walk length $l=80$, window size $w=10$ and the number of node embedding dimension is 128. For TADW, we set the parameter of balancing factor $\lambda=0.2$. For AANE, the balancing factors were set to 0.2 and 0.05 respectively, the penalty parameter was set to 5. In GraRep, we set the transition probability matrix parameter $kstep=4$. For the models ARGE and ARVGE, we constructed encoders with 32-units of hidden layers and 16-units of embedding layers. For DAEGC, we set 256 neurons for hidden layers and 16 neurons for embedding layers, the alpha of the leaky-relu was 0.2.

4.4. **Result Analysis.** The results of experiments on three datasets are summarized in Table 1, 2 and 3. The best results are highlighted in bold, and the rank of each algorithm is shown in parentheses. A smaller rank indicates better performance. To intuitively reflect the average rank of these algorithms, the overall average rank is depicted in Figure 2. The column "input" indicates the type of input data: "Features" means that the algorithm only considers node attribute information. "Graph" means that the algorithm only uses graph structure information. "Both" refers

[6] https://github.com/Tiger101010/DAEGC

to utilizing graph structure and node attribute information. The observations are summarized as follows:

1. Methods using graph structure information or attribute information generally perform worse than algorithms using the structure and attribute information. It indicates that combining two types of information is helpful for learning comprehensive node representation.
2. Our method using a similarity-based attention mechanism improves the value of NMI effectively on the Citeseer dataset. The NMI results of our method are 11% higher than that of ARGE and 12.9% higher than that of ARVGE. Compared with the original model DAEGC, the NMI value also increased by 1.4%.

TABLE 2. Experimental results on Cora dataset

	$Input$	ACC	NMI	$F-score$
K-means	Features	0.413(8)	0.229(9)	0.363(9)
Deepwalk	Graph	0.637(6)	0.449(7)	0.629(6)
GraRep	Graph	0.617(7)	0.463(6)	0.578(7)
TADW	Both	0.660(5)	0.480(5)	0.659(3)
AANE	Both	0.385(9)	0.234(8)	0.367(8)
ARGE	Both	0.661(4)	0.480(4)	0.645(5)
ARVGE	Both	0.697(3)	0.513(3)	0.657(4)
DAEGC	Both	0.723(2)	0.560(2)	**0.702(1)**
ours	Both	**0.733(1)**	**0.562(1)**	0.684(2)

TABLE 3. Experimental results on Citeseer dataset

	$Input$	ACC	NMI	$F-score$
K-means	Features	0.472(7)	0.231(8)	0.441(7)
Deepwalk	Graph	0.449(8)	0.239(7)	0.421(8)
GraRep	Graph	0.354(9)	0.187(9)	0.348(9)
TADW	Both	0.597(3)	0.340(3)	0.543(5)
AANE	Both	0.571(6)	0.301(5)	0.533(6)
ARGE	Both	0.581(5)	0.310(4)	0.558(3)
ARVGE	Both	0.584(4)	0.291(6)	0.556(4)
DAEGC	Both	**0.676(1)**	0.411(2)	**0.621(1)**
ours	Both	0.623(2)	**0.425(1)**	0.562(2)

4.5. **Parameter sensitivity.** Parameter sensitivity was conducted on the Citeseer dataset. We investigate the influence of different numbers of embedding dimensions on the performance. We need to choose an appropriate number of embedded dimensions. We first show how different numbers of embedding dimensions affect the performance of the algorithm. In Figure 3, we can see the value of NMI and $F-score$ increases first and then decreases with the increase of the embedding dimension. For ACC, it decreases first and then increases, and the value starts to deteriorate when the embedding dimension is large. Thus, to obtain the best performance on a certain dataset, we suggest searching parameter values first.

4.6. **Graph visualization.** The visualization tool t-SNE [50] was used to generate visualizations of the network on a two-dimensional space. We visualized the clustering results of several methods on the Citeseer dataset. For the units which belong to different categories, we used different colors to label them. Therefore, a good result requires that the points of the same

TABLE 4. Experimental results on Pubmed dataset

	Input	*ACC*	*NMI*	*F − score*
K-means	Features	0.596(6)	**0.310(1)**	0.582(8)
Deepwalk	Graph	0.660(2)	0.297(2)	0.665(2)
GraRep	Graph	0.594(7)	0.148(7)	0.592(6)
TADW	Both	0.622(5)	0.262(4)	0.616(5)
AANE	Both	0.573(8)	0.181(8)	0.589(7)
ARGE	Both	0.653(3)	0.250(5)	0.658(3)
ARVGE	Both	0.485(9)	0.068(9)	0.374(9)
DAEGC	Both	0.624(4)	0.213(6)	0.627(4)
ours	Both	**0.671(1)**	0.295(3)	**0.672(1)**

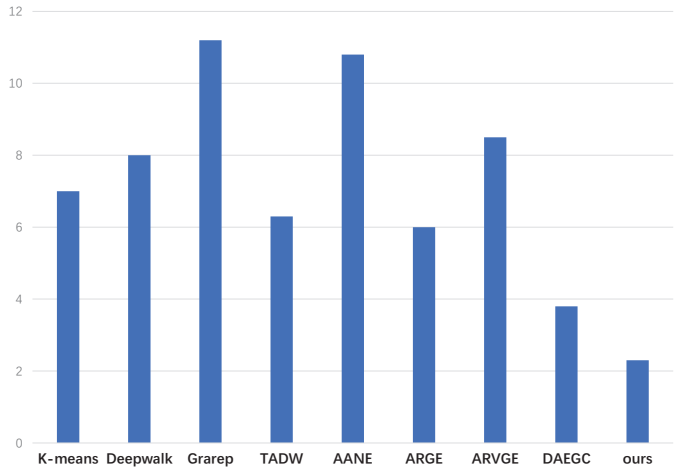
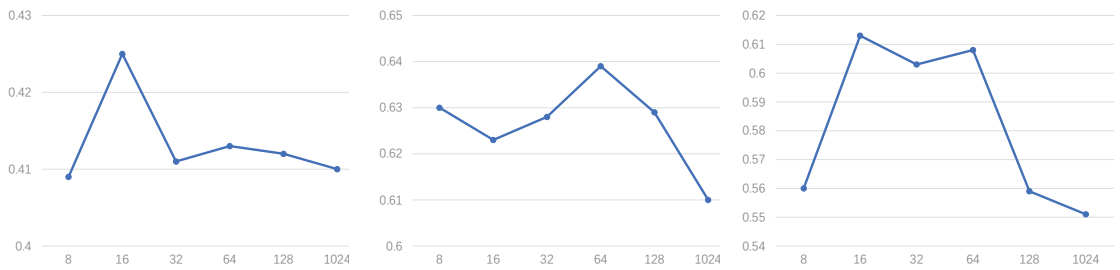


FIGURE 2. Overall average rank.



(a) *NMI* (b) *ACC* (c) *F − score*

FIGURE 3. Parameter analysis of different number of embedding dimensions.

color should be closed to each other and the boundaries of different clusters are clear. The visualization results are shown in Figure 4.

The Citeseer dataset was divided into six categories so the points in Figure 4 have six colors. It can be seen from (a) of Figure 4 that the points of different colors are mixed together

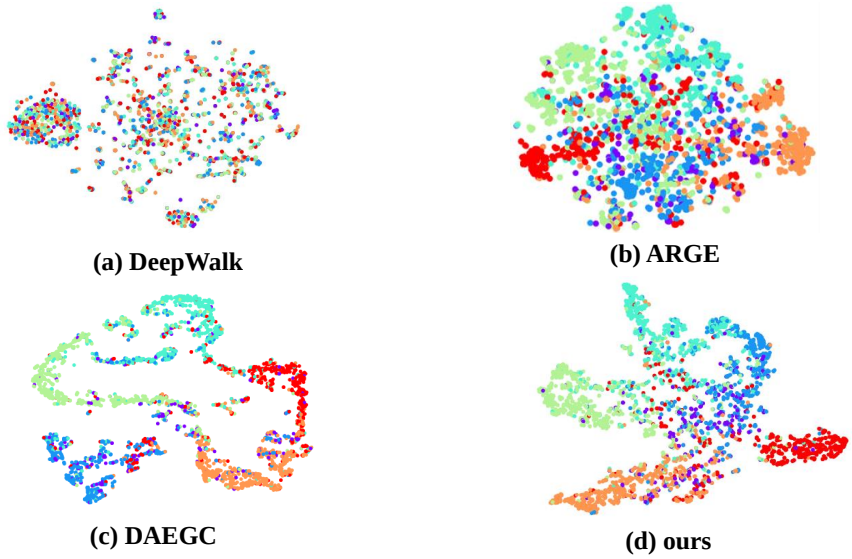


FIGURE 4. Visualization comparison of clustering results. The different colors represent different labels.

without a clear distinction. It indicates that the hidden representation learned by Deepwalk is not satisfactory. For the ARGE algorithm, points with the same color have a certain degree of aggregation, and the cluster of different colors is formed. But the center part of the units of different colors is still mixed. DAEGC results with clear boundaries, but only five distinct partitions can be seen. The Citeseer dataset has six clusters, so the points with two colors are completely mixed together with no distinction. The result of our method is better than the above three methods. The boundary between nodes with different colors is clear. Points of the same color are clustering together and it can be seen that there are six different colors clustered.

5. **Conclusions.** In this paper, we propose an improved version of DAEGC. The algorithm utilizes a graph encoder with a similarity-based attention mechanism to learn hidden representations for node clustering tasks. And during the training process, the node embeddings are updated using soft labels. The inner product decoder is used to reconstruct the graph. Using the attentional mechanism based on similarity in the encoder can make the NMI value increase stably. Experimental results show that our model performs better than other state-of-the-art methods on several evaluation metrics. In the future, we will try to better aggregate the graph structure and node attribute information.

Acknowledgment. This work is partially supported by the natural Science Foundation of Fujian Province of China (No. 2021J011187), the National Social Science Fund of China (No. 21BTJ011). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. Wei Weng and Bo-wen Wei proposed the methodology and revise the manuscript; Tong Li performed the experiment and wrote the manuscript; Jian-Chao Liao, Feng Guo and Fen Chen helped perform the analysis with constructive discussions.

REFERENCES

[1] W.-L. Hamilton, J. Zhang, and C. Danescu-Niculescu-Mizil, "Loyalty in online communities," in *International AAAI Conference on Web and Social Media*. AAAI, 2017, pp. 540–543.

[2] Z. Yang, W.-W. Cohen, and R. Salakhutdinov, "Revisiting Semi-Supervised Learning with Graph Embeddings," in *International Conference on International Conference on Machine Learning*. PMLR, 2016, pp. 40–48.
[3] M. Zitnik, and J. Leskovec Zitnik, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no.14, pp. i190–i198, 2017.
[4] T.-Y. Wu, X. Guo, L. Yang, Q. Meng, and C.-M. Chen, "A Lightweight Authenticated Key Agreement Protocol Using Fog Nodes in Social Internet of Vehicles," *Mobile Information Systems*, vol. 2021, 3277113, 2021.
[5] J.-C. Liao, T. Li, W. Weng, J.-B. Wang, and J. Wen, "Overlapping Community Detection by Motif-aware Label Propagation," *Journal of Network Intelligence*, vol. 7, no. 1, pp. 260–277, 2022.
[6] X. Shen, Q. Dai, F. Chung, W. Lu, and K.-S. Choi, "Adversarial deep network embedding for cross-network node classification," in *AAAI Conference on Artificial Intelligence*. AAAI, 2020, pp. 2991–2999.
[7] C. Wang, S. Pan, and P.-Y. Celina, "Deep neighbor-aware embedding for node clustering in attributed graphs," *Pattern Recognition* , vol. 122, 108230, 2022.
[8] H. Sun, F. He, J. Huang, Y. Sun, and Y. Li, "Network embedding for community detection in attributed networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 14, no. 3, pp. 1–25, 2020.
[9] X. Du, J. Yan, and H. Zha, "Joint Link Prediction and Network Alignment via Cross-graph Embedding," in *28th International Joint Conference on Artificial Intelligence*. IJCAI, 2019, pp. 2251–2257.
[10] C. Yang, Z. Liu, and D. Zhao, "Network representation learning with rich text information," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*. IJCAI, 2015, pp. 2111–2117.
[11] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang,"Semantic community identification in large attribute networks," in *AAAI Conference on Artificial Intelligence*. AAAI, 2016, pp. 265–271.
[12] J.-S. Ma, H.-H Xue, Y.-D. Zeng, Z.-C Zhang and Q.-C Wang, "Significant wave height forecasting using WRF-CLSF model in Taiwan strait," *Engineering Applications of Computational Fluid Mechanics*, vol. 15, no. 1, pp. 1400–1419, 2021.
[13] J. L, Tan. F, J. S, "Generative adversarial network technologies and applications in computer vision," *Computational Intelligence and Neuroscience*, vol. 2, pp. 1–17, 2020.
[14] S. Kumar, A. Damaraju, A. Kumar, S. Kumari, and C.-M. Chen, "LSTM Network for Transportation Mode Detection," *Journal of Internet Technology*, vol. 22, no. 4, pp. 891–902, 2021.
[15] F. Zhang, T.-Y. Wu, J.-S. Pan, G. Ding, and Z. Li, "Human Motion Recognition Based on SVM in VR Art Media Interaction Environment," *Human-centric Computing and Information Sciences*, vol. 9, 40, 2019.
[16] J. M.-T. Wu, Z. Li, N. Herencsar, B. Vo, and J. C.-W Lin, "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators," *Multimedia Systems*, 2021. [Online]. Available: https://doi.org/10.1007/s00530-021-00758-w
[17] J. M.-T Wu, Z. Li, G. Srivastava, J. Frnda, V. G. Diaz, and J. C.-W. Lin, "A CNN-based stock price trend prediction with futures and historical price," in *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*. IEEE, 2020, pp. 134–139.
[18] Z. Wu, S. Pan, and F. Chen, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.
[19] J. Zhou, G. Cui, and S. Hu, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
[20] T.-N. Kipf, and M. Welling, "Variational graph auto-encoders," in *Conference and Workshop on Neural Information Processing Systems* NIPS, 2016, pp.1–3.
[21] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed Graph Clustering: A Deep Attentional Embedding Approach," in *Twenty-Eighth International Joint Conference on Artificial Intelligence*. IJCAI, 2019, pp. 3670–3676.
[22] K.-K Thekumparampil, C. Wang, and S. Oh, "Attention-based graph neural network for semi-supervised learning," in *International Conference on Learning Representations* ICLR, 2018, pp.1–15.
[23] X. Zhang, H. Liu, and Q. Li, "Attributed graph clustering via adaptive graph convolution," in *28th International Joint Conference on Artificial Intelligence*. IJCAI, 2019, pp. 4327–4333.
[24] M. Girvan, and M. Newman, "Community structure in social and biological networks," *National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[25] U.-N Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, 036106, 2007.
[26] M. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, 036104, 2006.

[27] S.-S. Cao, W. Lu, and Q.-K. Xu, "Grarep: Learning graph representations with global structural information," in *24th ACM International Conference on Information and Knowledge Management*. CIKM, 2015, pp. 891–900.

[28] A. Grover, and J. Leskovec, "node2vec: Scalable feature learning for networks," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD, 2016, pp. 855–864.

[29] L. Kang, R.-S. Chen, N. Xiong, Y.-C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting Hyper-Parameters of Gaussian Process Regression Based on Non-Inertial Particle Swarm Optimization in Internet of Things," *IEEE Access*, vol. 7, pp. 59504–59513, 2019.

[30] T.-N. Kipf, and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, ICLR, 2017, pp.1–14.

[31] L. Liao, X. He, and H. Zhang, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[32] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics*. SDM, 2017, pp. 633–641.

[33] C. Wang, S. Pan, and G. Long, "MGAE: Marginalized Graph Autoencoder for Graph Clustering," in *ACM on Conference on Information and Knowledge Management*. CIKM, 2017, pp. 889–898.

[34] Z. Zhang, H. Yang, and J. Bu, "ANRL: Attributed Network Representation Learning via Deep Neural Networks," in *Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI, 2018, pp. 3155–3161.

[35] H. Gao, and H. Huang, "Deep attributed network embedding," in *Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI, 2018, pp. 3364–3370.

[36] S.-R. Pan, R.-Q. Hu, and G.-D. Long, "Adversarially regularized graph autoencoder for graph embedding," in *Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI, 2018, pp. 2609-2615.

[37] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *3rd International Conference on Learning Representations*, ICLR, 2015, pp.1–15.

[38] A. Vaswani, N. Shazeer, and N. Parmar, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 6000–6010, 2017.

[39] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*. CVPR, 2017, pp. 4438–4446.

[40] F. Sun, J. Liu, and J. Wu, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *28th ACM International Conference on Information and Knowledge Management*. CIKM, 2019, pp. 1441–1450.

[41] F. Zhang, T.-Y. Wu, and G. Zheng, "Video salient region detection model based on wavelet transform and feature comparison," *EURASIP Journal on Image and Video Processing* , vol. 2019, 58, 2019.

[42] E.-K. Wang, X. Zhang, F. Wang, T.-Y. Wu, and C.-M. Chen, "Multilayer Dense Attention Model for Image Caption," *IEEE Access*, vol. 7, pp. 66358–66368, 2019.

[43] P. Veličković, G. Cucurull, and A. Casanova, "Graph Attention Networks," in *International Conference on Learning Representations*, ICLR, 2018, pp. 1–12.

[44] K. Zhang, T. Li, and S. Shen, "Adaptive graph convolutional network with attention graph clustering for co-saliency detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR, 2020, pp. 9050–9059.

[45] T. He, Y.-S. Ong, and L. Bai, "Learning Conjoint Attentions for Graph Neural Nets," *Advances in Neural Information Processing Systems*, vol. 2021, no. 34, pp. 2641–2653, 2021.

[46] Z. Peng, H. Liu, and Y. Jia, "Attention-driven Graph Clustering Network," in *29th ACM International Conference on Multimedia*.ACM MM, 2021, pp. 935–943.

[47] J.-B. Lee, R.-A. Rossi, and S. Kim, "Attention models in graphs: A survey," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 6, pp. 1–25, 2019.

[48] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD, 2014, pp. 701–710.

[49] X. Su, S. Xue, and F. Liu, "A comprehensive survey on community detection with deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022. [Online]. Available: https://doi.org/10.1109/TNNLS.2021.3137396

[50] L. Van der Maaten, and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.