Attributed Graph Clustering: A Deep Attentional Embedding Approach

Chun Wang[1] , **Shirui Pan**[2] , **Ruiqi Hu**[1] , **Guodong Long**[1] , **Jing Jiang**[1] and **Chengqi Zhang**[1]

[1]Centre for Artificial Intelligence, University of Technology Sydney, Australia
[2]Faculty of IT, Monash University, Australia

{chun.wang-1, ruiqi.hu}@student.uts.edu.au, shirui.pan@monash.edu,
{guodong.long, jing.jiang, chengqi.zhang}@uts.edu.au

Abstract

Graph clustering is a fundamental task which discovers communities or groups in networks. Recent studies have mostly focused on developing deep learning approaches to learn a compact graph embedding, upon which classic clustering methods like k-means or spectral clustering algorithms are applied. These two-step frameworks are difficult to manipulate and usually lead to suboptimal performance, mainly because the graph embedding is not goal-directed, i.e., designed for the specific clustering task. In this paper, we propose a goal-directed deep learning approach, Deep Attentional Embedded Graph Clustering (`DAEGC` for short). Our method focuses on attributed graphs to sufficiently explore the two sides of information in graphs. By employing an attention network to capture the importance of the neighboring nodes to a target node, our DAEGC algorithm encodes the topological structure and node content in a graph to a compact representation, on which an inner product decoder is trained to reconstruct the graph structure. Furthermore, soft labels from the graph embedding itself are generated to supervise a self-training graph clustering process, which iteratively refines the clustering results. The self-training process is jointly learned and optimized with the graph embedding in a unified framework, to mutually benefit both components. Experimental results compared with state-of-the-art algorithms demonstrate the superiority of our method.

1 Introduction

The development of networked applications has resulted in an overwhelming number of scenarios in which data is naturally represented in graph format rather than flat-table or vector format. Graph-based representation characterizes individual properties through node attributes, and at the same time captures the pairwise relationship through the graph structure. Many real-world tasks, such as the analysis of citation networks, social networks, and protein-protein interaction, all rely on graph-data mining skills. However, the complexity of graph structure has imposed significant challenges on these graph-related learning tasks, including graph clustering, which is one of the most popular topics.

Graph clustering aims to partition the nodes in the graph into disjoint groups. Typical applications include community detection [Hastings, 2006], group segmentation [Kim *et al.*, 2006], and functional group discovery in enterprise social networks [Hu *et al.*, 2016]. Further for attributed graph clustering, a key problem is how to capture the structural relationship and exploit the node content information.

To solve this problem, more recent studies have resorted to deep learning techniques to learn compact representation to exploit the rich information of both the content and structure data [Wu *et al.*, 2019]. Based on the learned graph embedding, simple clustering algorithms such as k-means are applied. Autoencoder is a mainstream solution for this kind of embedding-based approach [Cao *et al.*, 2016; Tian *et al.*, 2014], as the autoencoder based hidden representation learning approach can be applied to purely unsupervised environments.

Nevertheless, all these embedding-based methods are two-step approaches. The drawback is that the learned embedding may not be the best fit for the subsequent graph clustering task, and the graph clustering task is not beneficial to the graph embedding learning. To achieve mutual benefit for these two steps, a goal-directed training framework is highly desirable. However, traditional goal-directed training models are mostly applied to the classification task. For instance, [Kipf and Welling, 2016] proposed graph convolutional networks for networked data. Fewer studies on goal-directed embedding methods for graph clustering exist, to the best of our knowledge.

Motivated by the above observations, we propose a goal-directed graph attentional autoencoder based attributed graph clustering framework in this paper. To exploit the interrelationship of various-typed graph data, we develop a graph attentional autoencoder to learn latent representation. The encoder exploits both graph structure and node content with a graph attention network, and multiple layers of encoders are stacked to build a deep architecture for embedding learning. The decoder on the other side, reconstruct the topological graph information and manipulates the latent graph representation. We further employ a self-training module, which takes the "*confident*" clustering assignments as soft labels to guide the optimizing procedure. By forcing the current clustering

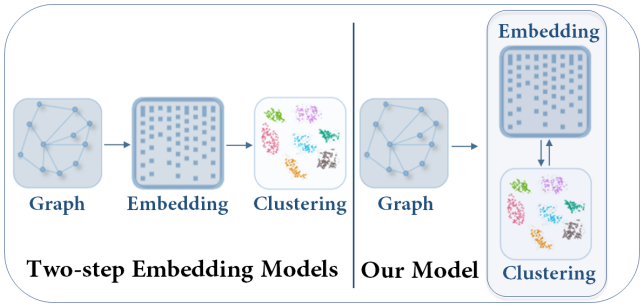


Figure 1: The difference between two-step embedding learning models and our model

distribution approaching a hypothetical better distribution, in contrast to the two-step embedding learning-based methods (shown in Fig 1), this specialized clustering component simultaneously learns the embedding and performs clustering in a unified framework, thereby achieving better clustering performance. Our contributions can be summarized as follows:

- We develop the first graph attention-based autoencoder to effectively integrate both structure and content information for deep latent representation learning.
- We propose a new goal-directed framework for attributed graph clustering. The framework jointly optimizes the embedding learning and graph clustering, to the mutual benefit of both components.
- The experimental results show that our algorithm outperforms state-of-the-art graph clustering methods.

2 Related Work

2.1 Graph Clustering

Graph clustering has been a long-standing research topic. Early methods have taken various shallow approaches to graph clustering. [Girvan and Newman, 2002] used centrality indices to find community boundaries and detect social communities. [Hastings, 2006] applied belief propagation to community detection and determined the most likely arrangement of communities. Many embedding learning based approaches apply an existing clustering algorithm on the learned embedding [Wang *et al.*, 2017b]. To handle both content and structure information, relational topic models [Sun *et al.*, 2009; Chang and Blei, 2009], co-clustering method [Guo *et al.*, 2019], and content propagation [Liu *et al.*, 2015] have also been widely used.

The limitations of these methods are that (1) they only capture either parts of the network information or shallow relationships between the content and structure data, and (2) they are directly applied on sparse original graphs. As a result, these methods cannot effectively exploit the graph structure or the interplay between the graph structure and the node content information.

In recent years, benefiting from the development of deep learning, graph clustering has progressed significantly. Many deep graph clustering algorithms employ autoencoders, adopting either the variational autoencoder [Kipf and Welling, 2016], sparse autoencoder [Tian *et al.*, 2014; Hu *et al.*, 2017], adversarially regularized method [Pan *et al.*, 2019] or denoising autoencoder [Cao *et al.*, 2016] to learn deep representation for clustering. However, these methods are two-step methods, whereas the algorithm presented in this paper is a unified approach.

2.2 Deep Clustering Algorithms

Autoencoders have been a widely used tool in the deep learning area, especially for unsupervised learning tasks such as clustering [Wang *et al.*, 2017a] and anomaly detection [Zhou and Paffenroth, 2017].

Deep Embedded Clustering (DEC) is a specialized clustering technique [Xie *et al.*, 2016]. This method employs a stacked denoising autoencoder learning approach. After obtaining the hidden representation of the autoencoder by pre-train, the encoder pathway is fine-tuned by a defined Kullback-Leibler divergence clustering loss. [Guo *et al.*, 2017a] considered that the defined clustering loss could corrupt the feature space and lead to non-representative features, so they added back the decoder and optimized the reconstruction error together with the clustering loss.

There have since then been increasing algorithms based on such deep clustering framework [Dizaji *et al.*, 2017; Guo *et al.*, 2017b]. However, as far as we know, they are only designed for data with flat-table representation. For graph data, complex structure and content information need to be carefully exploited, and goal-directed clustering for graph data is still an open problem in this area.

3 Problem Definition and Overall Framework

We consider clustering task on attributed graphs in this paper. A graph is represented as $G = (V, E, X)$, where $V = \{v_i\}_{i=1,\cdots,n}$ consists of a set of nodes, $E = \{e_{ij}\}$ is a set of edges between nodes. The topological structure of graph G can be represented by an adjacency matrix A, where $A_{i,j} = 1$ if $(v_i, v_j) \in E$; otherwise $A_{i,j} = 0$. $X = \{x_1; \ldots; x_n\}$ are the attribute values where $x_i \in R^m$ is a real-value attribute vector associated with vertex v_i.

Given the graph G, graph clustering aims to partition the nodes in G into k disjoint groups $\{G_1, G_2, \cdots, G_k\}$, so that nodes within the same cluster are generally: (1) close to each other in terms of graph structure while distant otherwise; and (2) more likely to have similar attribute values.

Our framework is shown in Fig 2 and consists of two parts: a graph attentional autoencoder and a self-training clustering module.

- **Graph attentional autoencoder.** Our autoencoder takes the attribute values and graph structure as input, and learns the latent embedding by minimizing the reconstruction loss.
- **Self-training clustering.** The self-training module performs clustering based on the learned representation, and in return, manipulates the latent representation according to the current clustering result.

We jointly learn the graph embedding and perform clustering in a unified framework, so that each component benefits the other.

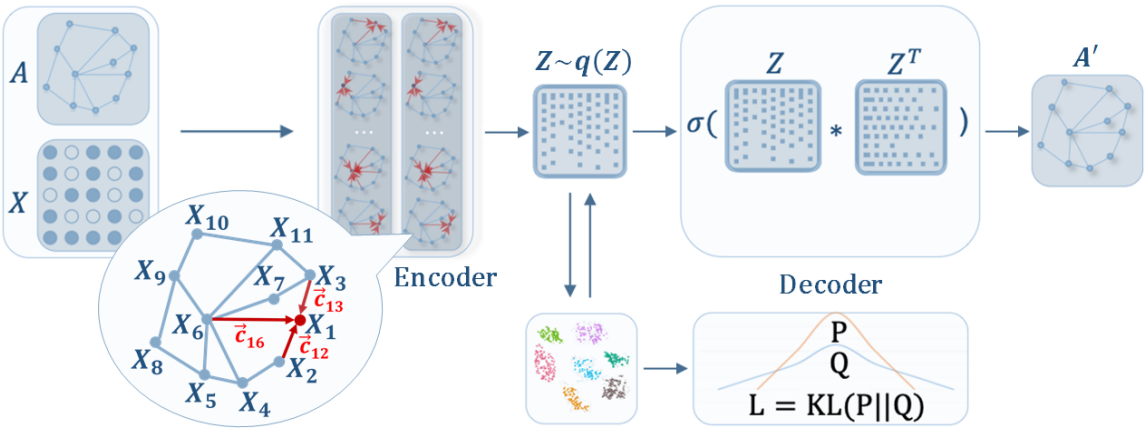


Figure 2: The conceptual framework of Deep Attentional Embedded Graph Clustering (DAEGC). Given a graph $G = (V, E, X)$, DAEGC learns a hidden representation Z through a graph attention-based autoencoder, and manipulates it with a self-training clustering module, which is optimized together with the autoencoder and perform clustering during training.

4 Proposed Method

In this section, we present our proposed DAEGC approach. We first develop a graph attentional autoencoder which effectively integrates both structure and content information to learn a latent representation. Based on the representation, a self-training module is proposed to guide the clustering algorithm towards better performance.

4.1 Graph Attentional Autoencoder

Graph Attentional Encoder

To represent both graph structure A and node content X in a unified framework, we develop a variant of the graph attention network [Velickovic *et al.*, 2017] as a graph encoder. The idea is to learn hidden representations of each node by attending over its neighbors, to combine the attribute values with the graph structure in the latent representation. The most straightforward strategy to attend the neighbors of a node is to integrate its representation equally with all its neighbors. However, in order to measure the importance of various neighbors, different weights are given to the neighbor representations in our layer-wise graph attention strategy:

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

Here, z_i^{l+1} denotes the output representation of node i, and N_i denotes the neighbors of i. α_{ij} is the attention coefficient that indicates the importance of neighbor node j to node i, and σ is a nonlinerity function. To calculate the attention coefficient α_{ij}, we measure the importance of neighbor node j from both the aspects of the attribute value and the topological distance.

From the perspective of attribute values, the attention coefficient α_{ij} can be represented as a single-layer feedforward neural network on the concatenation of x_i and x_j with weight vector $\overrightarrow{a} \in R^{2m'}$:

$$c_{ij} = \overrightarrow{a}^T [Wx_i || Wx_j]. \tag{2}$$

Topologically, neighbor nodes contribute to the representation of a target node through edges. GAT considers only the 1-hop neighboring nodes (first-order) for graph attention [Velickovic *et al.*, 2017]. As graphs have complex structure relationships, we propose to exploit high-order neighbors in our encoder. We obtain a proximity matrix by considering t-order neighbor nodes in the graph:

$$M = (B + B^2 + \cdots + B^t)/t, \tag{3}$$

here B is the transition matrix where $B_{ij} = 1/d_i$ if $e_{ij} \in E$ and $B_{ij} = 0$ otherwise. d_i is the degree of node i. Therefore M_{ij} denotes the topological relevance of node j to node i up to t orders. In this case, N_i means the neighboring nodes of i in M. i.e., j is a neighbor of i if $M_{ij} > 0$. t could be chosen flexibly for different datasets to balance the precision and efficiency of the model.

The attention coefficients are usually normalized across all neighborhoods $j \in N_i$ with a softmax function to make them easily comparable across nodes:

$$\alpha_{ij} = \text{softmax}_j(c_{ij}) = \frac{\exp(c_{ij})}{\sum_{r \in N_i} \exp(c_{ir})}. \tag{4}$$

Adding the topological weights M and an activation function δ (here LeakyReLU is used), the coefficients can be expressed as:

$$\alpha_{ij} = \frac{\exp(\delta M_{ij}(\overrightarrow{a}^T [Wx_i || Wx_j]))}{\sum_{r \in N_i} \exp(\delta M_{ir}(\overrightarrow{a}^T [Wx_i || Wx_r]))}. \tag{5}$$

We have $x_i = z_i^0$ as the input for our problem, and stack two graph attention layers:

$$z_i^{(1)} = \sigma(\sum_{j \in N_i} \alpha_{ij} W^{(0)} x_j), \tag{6}$$

$$z_i^{(2)} = \sigma(\sum_{j \in N_i} \alpha_{ij} W^{(1)} z_j^{(1)}), \tag{7}$$

in this way, our encoder encodes both the structure and the node attributes into a hidden representation, i.e., we will have $z_i = z_i^{(2)}$.

Inner Product Decoder

There are various kinds of decoders, which reconstruct either the graph structure, the attribute value, or both. As our latent embedding already contains both content and structure information, we choose to adopt a simple inner product decoder to predict the links between nodes, which would be efficient and flexible:

$$\hat{A}_{ij} = \text{sigmoid}({z_i}^\top z_j), \tag{8}$$

where $\hat{A}$ is the reconstructed structure matrix of the graph.

Reconstruction Loss

We minimize the reconstruction error by measuring the difference between A and $\hat{A}$:

$$L_r = \sum_{i=1}^{n} loss(A_{i,j}, \hat{A}_{ij}). \tag{9}$$

4.2 Self-optimizing Embedding

One of the main challenges for graph clustering methods is the nonexistence of label guidance. The graph clustering task is naturally unsupervised and feedback during training as to whether the learned embedding is well optimized cannot therefore be obtained. To confront this challenge, we develop a self-optimizing embedding algorithm as a solution.

Apart from optimizing the reconstruction error, we input our hidden embedding into a self-optimizing clustering module which minimizes the following objective:

$$L_c = KL(P||Q) = \sum_i \sum_u p_{iu} log \frac{p_{iu}}{q_{iu}}, \tag{10}$$

Where q_{iu} measures the similarity between node embedding z_i and cluster center embedding μ_u. We measure it with a Student's t-distribution so that it could handle different scaled clusters and is computationally convenient [Maaten and Hinton, 2008]:

$$q_{iu} = \frac{(1 + ||z_i - \mu_u||^2)^{-1}}{\sum_k (1 + ||z_i - \mu_k||^2)^{-1}}, \tag{11}$$

it can be seen as a soft clustering assignment distribution of each node. On the other hand, p_{iu} is the target distribution defined as:

$$p_{iu} = \frac{q_{iu}^2 / \sum_i q_{iu}}{\sum_k (q_{ik}^2 / \sum_i q_{ik})}. \tag{12}$$

Soft assignments with high probability (nodes close to the cluster center) are considered to be trustworthy in Q. So the target distribution P raises Q to the second power to emphasize the role of those "confident assignments". The clustering loss then force the current distribution Q to approach the target distribution P, so as to set these "confident assignments" as soft labels to supervise Q's embedding learning.

To this end, we first train the autoencoder without the self-optimize clustering part to obtain a meaningful embedding z as described in Eq.(7). Self-optimizing clustering is then performed to improve this embedding. To obtain the soft clustering assignment distributions of all the nodes Q through Eq.(11), the k-means clustering is performed once and for all on the embedding z before training the entire model, to obtain the initial cluster centers μ.

Then in the following training, the cluster centers μ are updated together with the embedding z using Stochastic Gradient Descent (SGD) based on the gradients of L_c with respect to μ and z.

We calculate the target distribution P according to Eq.(12), and the clustering loss L_c according to Eq.(10).

The target distribution P works as "ground-truth labels" in the training procedure, but also depends on the current soft assignment Q which updates at every iteration. It would be hazardous to update P at every iteration with Q as the constant change of target would obstruct learning and convergence. To avoid instability in the self-optimizing process, we update P every 5 iterations in our experiment.

In summary, we minimize the clustering loss to help the autoencoder manipulate the embedding space using the embedding's own characteristics and scatter embedding points to obtain better clustering performance.

Dataset	Nodes	Features	Clusters	Links	Words
Cora	2,708	1,433	7	5,429	3,880,564
Citeseer	3,327	3,703	6	4,732	12,274,336
Pubmed	19,717	500	3	44,338	9,858,500

Table 1: Benchmark Graph Datasets

4.3 Joint Embedding and Clustering Optimization

We jointly optimize the autoencoder embedding and clustering learning, and define our total objective function as:

$$L = L_r + \gamma L_c, \tag{13}$$

where L_r and L_c are the reconstruction loss and clustering loss respectively, $\gamma \geq 0$ is a coefficient that controls the balance in between. It is worth mentioning that we could gain our clustering result directly from the last optimized Q, and the label estimated for node v_i could be obtained as:

$$s_i = \arg\max_u q_{iu}, \tag{14}$$

which is the most likely assignment from the last soft assignment distribution Q.

Our method is summarized in Algorithm 1. Our algorithm has the following advantages:

- **Interplay exploitation.** The graph attention network-based autoencoder efficiently exploits the interplay between both the structure and content information.
- **Clustering specialized embedding.** The proposed self-training clustering component manipulates the embedding to improve the clustering performance.
- **Joint learning.** The framework jointly optimizes the two parts of the loss functions, learns the embedding and performs clustering in a unified framework.

5 Experiments

5.1 Benchmark Datasets

We used three standard citation networks widely-used for assessment of attributed graph analysis in our experiments, summarized in Table 1. Publications in the datasets are categorized by the research sub-fields.

Algorithm 1 Deep Attentional Embedded Graph Clustering

Require:
Graph G with n nodes; Number of clusters k; Number of iterations $Iter$; Target distribution update interval T; Clustering Coefficient γ.
Ensure:
Final clustering results.

Update the autoencoder by minimizing Eq.(9) to get the autoencoder hidden embedding Z;
Compute the initial cluster centers μ based on Z;
for $l = 0$ to $Iter - 1$ **do**
  Calculate soft assignment distribution Q with Z and μ according to Eq.(11);
  if $l\%T == 0$ **then**
    Calculate target distribution P with Q by Eq.(12);
  end if
  Calculate clustering loss L_c according to Eq.(10);
  Update the whole framework by minimizing Eq.(13);
end for
Get the clustering results with final Q by Eq.(14)

5.2 Baseline Methods

We compared a total of ten algorithms with our method in our experiments. The graph clustering algorithms include approaches that use only node attributes or network structure information, and also approaches that combine both. Deep representation learning-based graph clustering algorithms were also compared.

Methods Using Structure or Content Only

- K**-means** is the basis of many clustering methods.
- **Spectral clustering** uses the eigenvalues to perform dimensionality reduction before clustering.
- **GraphEncoder** [Tian *et al.*, 2014] trains a stacked sparse autoencoder to obtain representation.
- **DeepWalk** [Perozzi *et al.*, 2014] is a structure-only representation learning method.
- **DNGR** [Cao *et al.*, 2016] uses stacked denoising autoencoders and encodes each vertex into a low dimensional vector representation.
- **M-NMF** [Wang *et al.*, 2017b] is a Nonnegative Matrix Factorization model targeted at community-preserved embedding.

Methods Using Both Structure and Content

- **RMSC** [Xia *et al.*, 2014] is a robust multi-view spectral clustering method. We regard structure and content data as two views of information.
- **TADW** [Yang *et al.*, 2015] regards DeepWalk as a matrix factorization method and adds the features of vertices for representation learning.
- **VGAE & GAE** [Kipf and Welling, 2016] combine graph convolutional network with the (variational) autoencoder to learn representations.

	Info.	ACC(↑)	NMI(↑)	F-score(↑)	ARI(↑)
K-means	C	0.500	0.317	0.376	0.239
Spectral	S	0.398	0.297	0.332	0.174
GraphEncoder	S	0.301	0.059	0.230	0.046
DeepWalk	S	0.529	0.384	0.435	0.291
DNGR	S	0.419	0.318	0.340	0.142
M-NMF	S	0.423	0.256	0.320	0.161
RMSC	C&S	0.466	0.320	0.347	0.203
TADW	C&S	0.536	0.366	0.401	0.240
GAE	C&S	0.530	0.397	0.415	0.293
VGAE	C&S	0.592	0.408	0.456	0.347
DAEGC	C&S	**0.704**	**0.528**	**0.682**	**0.496**

Table 2: Experimental Results on **Cora** Dataset

	Info.	ACC(↑)	NMI(↑)	F-score(↑)	ARI(↑)
K-means	C	0.544	0.312	0.413	0.285
Spectral	S	0.308	0.090	0.257	0.082
GraphEncoder	S	0.293	0.057	0.213	0.043
DeepWalk	S	0.390	0.131	0.305	0.137
DNGR	S	0.326	0.180	0.300	0.043
M-NMF	S	0.336	0.099	0.255	0.070
RMSC	C&S	0.516	0.308	0.404	0.266
TADW	C&S	0.529	0.320	0.436	0.286
GAE	C&S	0.380	0.174	0.297	0.141
VGAE	C&S	0.392	0.163	0.278	0.101
DAEGC	C&S	**0.672**	**0.397**	**0.636**	**0.410**

Table 3: Experimental Results on **Citeseer** Dataset

- **DAEGC** is our proposed unsupervised deep attentional embedded graph clustering.

For representation learning algorithms such as DeepWalk, TADW and DNGR which do not specify the clustering algorithm, we learned the representation from these algorithms, and then applied the k-means algorithm on their respective representations, but for algorithms like RMSC which require an alternative algorithm as its clustering method, we followed their preference and used the specified algorithms.

5.3 Evaluation Metrics & Parameter Settings

Metrics We use four metrics [Xia *et al.*, 2014] to evaluate the clustering result: Accuracy (ACC), Normalized Mutual Information (NMI), F-score, and Adjusted Rand Index (ARI). A better clustering result should lead to a higher values for all the metrics.

Baseline settings For the baseline algorithms, we carefully select the parameters for each algorithm, following the procedures in the original papers. In TADW, for instance, we set the dimension of the factorized matrix to 80 and the regularization parameter to 0.2; For the RMSC algorithm, we regard graph structure and node content as two different views of the data and construct a Gaussian kernel on them. We run the k-means algorithm 50 times to get an average score for all embedding learning methods for fair comparison.

Parameter settings For our method, we set the clustering coefficient γ to 10. We consider second-order neighbors and set $M = (B+B^2)/2$. The encoder is constructed with a 256-

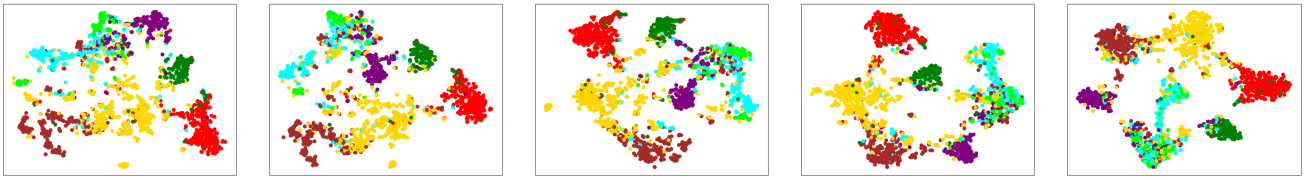
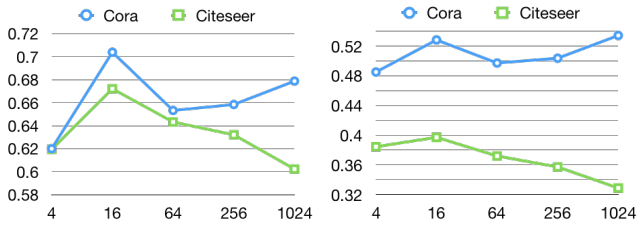


Figure 3: 2D visualization of the DAEGC algorithm on the Cora dataset during training. The first visualization illustrates the embedding training with the graph attentional autoencoder only, followed by visualizations showing subsequent equal epochs in which the self-training component is included, till the last one being the final embedding visualization.



(a) ACC (b) NMI

Figure 4: Parameter v.s. different dimensions of the embedding

neuron hidden layer and a 16-neuron embedding layer for all datasets.

5.4 Experiment Results

The experiment results on the three benchmark datasets are summarized in Table 2, 3, and 4, where the bold values indicate the best performance. C, S, and C&S indicate if the algorithm uses only content, structure, or both content and structure information, respectively. We can see that our method clearly outperforms all the baselines across most of the evaluation metrics.

We can observe from these results that methods using both the structure and content information of the graph generally perform better than those using only one side of information. In the Cora dataset, for example, TADW, GAE, VGAE and our method outperform all the baselines using one side of information. This observation demonstrates that both the graph structure and node content contain useful information for graph clustering, and illustrates the significance of capturing the interplay between two-sides information.

The results of most of the deep learning models are satisfactory. The GraphEncoder and DNGR algorithm are not necessarily an improvement although they both employ deep autoencoder for representation learning. This observation may result from their neglect at the node content information.

It is worth mentioning that our algorithm significantly outperforms GAE and VGAE. On the Cora dataset for example, our method represents a relative increase of 18.97% and 29.49% w.r.t. accuracy and NMI against VGAE, and the increase is even greater on the Citeseer dataset. The reasons for this are that (1) we employ a graph attention network that effectively integrates both content and structure information of the graph; (2) Our self-training clustering component is specialized and powerful in improving the clustering efficiency.

Parameter study We vary the dimension of embedding from 4 neurons to 1024 and report the results in Fig 4. It can be observed from both 4(a) and 4(b) that: when adding the dimension of embedding from 4-neuron to 16-neuron, the performance on clustering steadily rises; but when we further increase the neurons of the embedding layer, the performance fluctuates, though the ACC and NMI score both remain good on the whole.

	Info.	ACC(↑)	NMI(↑)	F-score(↑)	ARI(↑)
K-means	C	0.580	**0.278**	0.544	0.246
Spectral	S	0.496	0.147	0.471	0.098
GraphEncoder	S	0.531	0.210	0.506	0.184
DeepWalk	S	0.647	0.238	0.530	0.255
DNGR	S	0.468	0.153	0.445	0.059
M-NMF	S	0.470	0.084	0.443	0.058
RMSC	C&S	0.629	0.273	0.521	0.247
TADW	C&S	0.565	0.224	0.481	0.177
GAE	C&S	0.632	0.249	0.511	0.246
VGAE	C&S	0.619	0.216	0.478	0.201
DAEGC	C&S	**0.671**	0.266	**0.659**	**0.278**

Table 4: Experimental Results on **Pubmed** Dataset

Network visualization We visualize the Cora dataset in a two-dimensional space by applying the t-SNE algorithm [Van Der Maaten, 2014] on the learned embedding during training. The result in Fig 3 demonstrates that, after training with our graph attentional autoencoder, the embedding is already meaningful. However by applying self-training clustering, the embedding becomes more evident as our training progresses, with less overlapping and each group of nodes gradually gathered together.

6 Conclusion

In this paper, we propose an unsupervised deep attentional embedding algorithm, DAEGC, to jointly perform graph clustering and learn graph embedding in a unified framework. The learned graph embedding integrates both the structure and content information and is specialized for clustering tasks. While the graph clustering task is naturally unsupervised, we propose a self-training clustering component that generates soft labels from "confident" assignments to supervise the embedding updating. The clustering loss and autoencoder reconstruction loss are jointly optimized to simultaneously obtain both graph embedding and graph clustering result. A comparison of the experimental results with various state-of-the-art algorithms validate DAEGC's graph clustering performance.

References

[Cao *et al.*, 2016] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proc. of AAAI*, pages 1145–1152, 2016.

[Chang and Blei, 2009] Jonathan Chang and David M Blei. Relational topic models for document networks. In *AISTATS*, volume 9, pages 81–88, 2009.

[Dizaji *et al.*, 2017] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, pages 5747–5756. IEEE, 2017.

[Girvan and Newman, 2002] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proc. of National Acad Sciences*, 99(12):7821–7826, 2002.

[Guo *et al.*, 2017a] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Proc. of IJCAI*, pages 1753–1759, 2017.

[Guo *et al.*, 2017b] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *ICONIP*, pages 373–382. Springer, 2017.

[Guo *et al.*, 2019] Ting Guo, Shirui Pan, Xingquan Zhu, and Chengqi Zhang. Cfond: consensus factorization for co-clustering networked data. *IEEE TKDE*, 31(4):706–719, 2019.

[Hastings, 2006] Matthew B Hastings. Community detection as an inference problem. *Physical Review E*, 74(3):035102, 2006.

[Hu *et al.*, 2016] Ruiqi Hu, Shirui Pan, Guodong Long, Xingquan Zhu, Jing Jiang, and Chengqi Zhang. Co-clustering enterprise social networks. In *IJCNN*, pages 107–114, 2016.

[Hu *et al.*, 2017] Pengwei Hu, Keith CC Chan, and Tiantian He. Deep graph clustering in social network. In *Proc. of WWW*, pages 1425–1426, 2017.

[Kim *et al.*, 2006] Su-Yeon Kim, Tae-Soo Jung, Eui-Ho Suh, and Hyun-Seok Hwang. Customer segmentation and strategy development based on customer lifetime value: A case study. *Expert Systems with Applications*, 31(1):101–107, 2006.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[Liu *et al.*, 2015] Liyuan Liu, Linli Xu, Zhen Wangy, and Enhong Chen. Community detection based on structure and content: A content propagation perspective. In *Proc. of ICDM*, pages 271–280. IEEE, 2015.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, pages 2579–2605, 2008.

[Pan *et al.*, 2019] Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning graph embedding with adversarial training methods. *arXiv preprint arXiv:1901.01250*, 2019.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proc. of KDD*, pages 701–710. ACM, 2014.

[Sun *et al.*, 2009] Yizhou Sun, Jiawei Han, Jing Gao, and Yintao Yu. Itopicmodel: Information network-integrated topic modeling. In *Proc. of ICDM*, pages 493–502. IEEE, 2009.

[Tian *et al.*, 2014] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *Proc. of AAAI*, pages 1293–1299, 2014.

[Van Der Maaten, 2014] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[Velickovic *et al.*, 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang *et al.*, 2017a] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proc. of CIKM*, pages 889–898. ACM, 2017.

[Wang *et al.*, 2017b] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proc. of AAAI*, 2017.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[Xia *et al.*, 2014] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *Proc. of AAAI*, pages 2149–2155, 2014.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[Yang *et al.*, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *Proc. of IJCAI*, pages 2111–2117, 2015.

[Zhou and Paffenroth, 2017] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proc. of KDD*, pages 665–674. ACM, 2017.