mathematics

MDPI

Article

Community Detection Fusing Graph Attention Network

Ruiqiang Guo [1,2,3], Juan Zou [4], Qianqian Bai [1,2,3], Wei Wang [1,2,3,*] and Xiaomeng Chang [1,2,3]

[1] College of Computer and Cyber Security, Hebei Normal University, Shijiazhuang 050024, China
[2] Hebei Provincial Engineering Research Center for Supply Chain Big Data Analytics & Data Security, Hebei Normal University, Shijiazhuang 050024, China
[3] Hebei Provincial Key Laboratory of Network & Information Security, Hebei Normal University, Shijiazhuang 050024, China
[4] Key Laboratory of Intelligent Computing and Information Processing, Ministry of Education, School of Computer Science and School of Cyberspace Science, Xiangtan University, Xiangtan 411105, China
* Correspondence: wangwei2021@hebtu.edu.cn

Abstract: It has become a tendency to use a combination of autoencoders and graph neural networks for attribute graph clustering to solve the community detection problem. However, the existing methods do not consider the influence differences between node neighborhood information and high-order neighborhood information, and the fusion of structural and attribute features is insufficient. In order to make better use of structural information and attribute information, we propose a model named community detection fusing graph attention network (CDFG). Specifically, we firstly use an autoencoder to learn attribute features. Then the graph attention network not only calculates the influence weight of the neighborhood node on the target node but also adds the high-order neighborhood information to learn the structural features. After that, the two features are initially fused by the balance parameter. The feature fusion module extracts the hidden layer representation of the graph attention layer to calculate the self-correlation matrix, which is multiplied by the node representation obtained by the preliminary fusion to achieve secondary fusion. Finally, the self-supervision mechanism makes it face the community detection task. Experiments are conducted on six real datasets. Using four evaluation metrics, the CDFG model performs better on most datasets, especially for the networks with longer average paths and diameters and smaller clustering coefficients.

Keywords: graph attention network; high-order neighborhood; attribute network; community detection

MSC: 68T01

Citation: Guo, R.; Zou, J.; Bai, Q.; Wang, W.; Chang, X. Community Detection Fusing Graph Attention Network. *Mathematics* **2022**, *10*, 4155. https://doi.org/10.3390/math10214155

Academic Editor: Jakub Nalepa

Received: 30 September 2022
Accepted: 3 November 2022
Published: 7 November 2022

1. Introduction

Community detection is a fundamental task in complex network analysis, which aims to partition a network into multiple substructures (communities). Usually, a community is defined as a set of nodes with a different affiliation from the rest of the network [1]. Community detection has been extensively studied and applied in many real-world network problems, such as recommendation [2], anomaly detection [3], and terrorist organization identification [4]. Classical community detection methods usually utilize probabilistic models and statistical inference methods. These methods employ different varieties of prior knowledge to infer community structure. For example, classical community detection methods, spectral clustering [5], GN algorithm [6], etc. However, traditional community detection algorithms usually focus only on the network structure and often ignore the attributes of nodes, resulting in a lack of semantic community division. In the real world, the attributes of nodes are becoming more affluent and more prosperous, and a more reasonable solution is to consider both the relationships between nodes and semantic information. For community detection in attribute networks, a balance should be achieved between the following two properties: (1) structural closeness, i.e., nodes within a community are structurally close to each other, while nodes in different communities are not,

and (2) attribute homogeneity, i.e., nodes in a community have similar attributes, while nodes in different ones are different [7].

Community detection is a typical application of graph clustering. For attributed graph clustering, capturing the network topology and utilizing the content information of nodes is a crucial problem. The method based on graph embedding obtains the node low-dimensional vector representation by learning the network topology and node content [8]. On this basis, the current research focus is the application of clustering methods such as K-means to solve the problem of community detection. The autoencoder is the mainstream solution for graph embedding-based methods [9], because the autoencoder-based representation can be applied to unsupervised scenarios. Inspired by the above methods, we use an autoencoder and a graph neural network as the basic framework for attribute graph clustering.

In this paper, we propose a community detection fusing graph attention network (CDFG) model. The main contributions are: (1) we fuse the autoencoder and the graph attention network with high-order neighborhood information for the first time. (2) We design the feature fusion module. Specifically, the graph attention layer in the graph attention network [10] aggregates node feature information in the neighborhood by trainable weights and considers the different extent of influence of neighborhood nodes on the target node. Then we obtain the high-order neighborhood information of the target node by calculating the topological correlation matrix. The correlation between nodes is fully utilized. The feature fusion module calculates the self-correlation matrix by taking the hidden layer representation obtained from the graph attention network and then multiplies the obtained autocorrelation matrix with the matrix obtained from the graph attention module to obtain the final node representation using the principle of jump connection. The results of conducted experiments on six real datasets and evaluating the model using four evaluation metrics show that the model performs better than other methods.

2. Related Work

2.1. Traditional Methods

Traditional methods are based on network topology for community detection, which can be divided into graph partitioning, statistical inference, hierarchical clustering, dynamic methods, spectral clustering, density-based methods, and optimization methods according to the principles applied [11]. These methods only capture the shallow structure of the network and have a high computational complexity for large-scale network data. With the increasing richness of information in the real world, traditional community detection methods can no longer meet demands.

2.2. Graph Embedding Methods

Graph embedding methods can map high-dimensional sparse vectors into low-dimensional dense vectors with the advantage of using high-dimensional nonlinear features (e.g., network topology information) and high-dimensional relational features (e.g., node attribute information) represented by nodes, neighbors, edges, subgraphs (e.g., communities), and encoded features [12]. In this kind of method, the nodes in a complex network are represented by low-dimensional real-value vectors, and the traditional clustering method can be used to solve the community detection problem. At present, deep clustering approaches focuses on the graph convolutional network-based approaches and autoencoder-based approaches.

Clustering methods based on graph convolutional networks (GCN) [13] to learn graph structure and node attributes have been widely studied [14–23]. For attribute graphs, neighboring nodes and nodes with similar characteristics may gather in the same community. Graph autoencoders (GAE) and variational graph autoencoders (VGAE) [14] integrate graph structures into node attributes by iteratively aggregating the neighborhood representation around each central node. Deep attentional embedded graph clustering (DAEGC) [15] uses a graph structure and node attributes at the same time. It captures the importance of the neighborhood nodes through a graph attention network as an

encoder and uses KL-divergence loss to supervise the training process of graph clustering. On the basis of DAEGC, the deep-neighbor-aware embedding for node clustering in attributed graphs (DNENC) [16] uses GCN as the encoder, which complements the contrast experiments. The experimental results show the effectiveness of the proposed framework. According to the setting of DAEGC, the adversarially regularized graph autoencoder (ARGA) [17] further developed an adversarial regulator to guide the learning of potential representations. Structural deep clustering network (SDCN) [18] integrates an information transfer operator, a dual self-supervised learning mechanism, an autoencoder, and a graph convolution network into a unified framework for better representation learning better. Experiments show that the autoencoder can alleviate the over-fitting phenomenon. The hierarchical attention network (HiAN) [23] designs the hierarchical attentive aggregator to fuse rich interpretable interactive information. These GCN-based methods still have the problem of smoothing. Meanwhile, GCN only aggregates neighborhood information equally when learning structural representation.

Many deep clustering methods based on the autoencoder also have been proposed [24–27]. The autoencoder (AE) [28] is the most commonly used solution for unsupervised community detection. Deep embedded clustering (DEC) [24] first trains the encoder and then uses a pretrained network to iteratively optimize the KL divergence-based clustering loss with the help of self-learning-assisted target distributions so that the representation learned by the autoencoder is closer to the center of the clusters and improves the cohesiveness of the clusters. To improve the accuracy of the target distribution, the improved deep embedded clustering (IDEC) [25] jointly optimizes the clustering assignment and learns features suitable for clustering while maintaining local structure. These methods help the autoencoder learn data representations with higher relevance to clustering by computing clustering loss, exploiting the features of the data itself but not incorporating structural information.

3. The Proposed Model

In this section, we present the community detection fusing graph attention network (CDFG) shown in Figure 1. We first introduce the notation and the problem definition. In the following subsections, we describe the CDFG model in detail in four modules, i.e., the autoencoder module (AE), the graph attention network module (GAT), the feature fusion module, and the self-supervision learning module.

A complex network whose nodes have attributes is an attribute network. Given an attribute network $G = \{V,\ E, X\}$, where $V = \{v_1, v_2, \ldots v_N\}$ is the set of nodes, E is the set of edges. N is the number of nodes. $X = \{x_i\}_1^N$ is the feature matrix, where $x_i \subseteq \mathbb{R}^d$ denotes the attribute vector of node v_i. Here, d is the attribute dimension. The topology of the graph G can be represented by the adjacency matrix $A = \left(a_{ij}\right)_{N\times N} \in \mathbb{R}^{N\times N}$ and if $\left(v_i, v_j\right) \in E$, $a_{ij} = 1$, otherwise $a_{ij} = 0$.

Problem Definition. Given an attribute network G, it is divided into several disjoint groups, i.e., $\{G_1, G_2, \ldots G_k\}$. Each community G_i is a partition of the network G, and k represents the number of communities divided from the original network G with common properties of clustering. Nodes in the same community should satisfy to be structurally tightly connected and have more similar attributes, while nodes in different communities are sparsely connected and have different attributes. $G_i \cap G_j = \text{ø}$ means that there is no intersection between communities, indicating non-overlapping community detection.

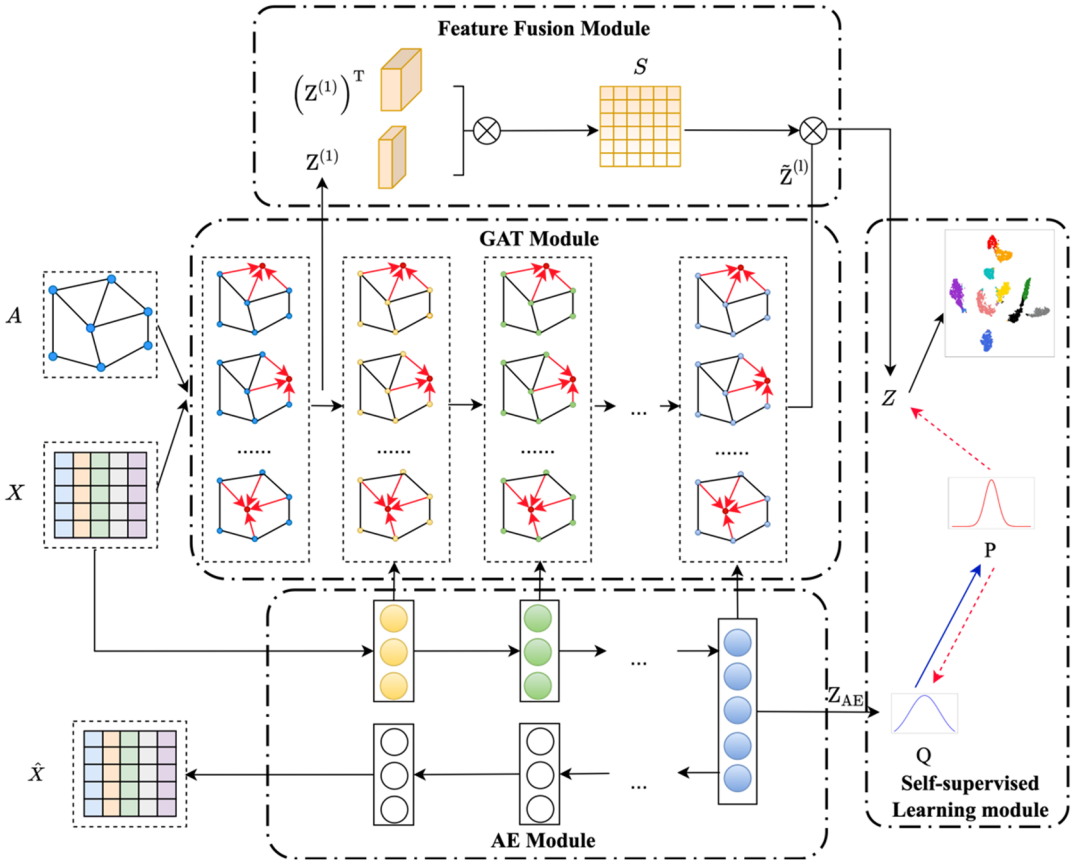


Figure 1. The architecture of the CDFG. A denotes the adjacency matrix of the graph. X is the feature matrix of the node. A and X are both used as input to the GAT module. X is used as input to the AE module. Z_{AE} is the hidden layer representation obtained by the AE module. $Z^{(1)}$ denotes the first layer representation obtained by the graph attention module. $\left(Z^{(1)}\right)^T$ is the transpose of $Z^{(1)}$. S is the self-correlation matrix. $\widetilde{Z}^{(l)}$ is the representation obtained after the fusion of the GAT module and the AE module. $\otimes$ denotes the computation of matrix multiplication, and Z is the final representation obtained after fusion. Q is obtained based on the Student's t-distribution and denotes the distance relationship between Z_{AE} and the clustering centre. P is the target distribution calculated from Q.

3.1. AE Module

A basic autoencoder [28] is used for unsupervised representation learning of the data from the perspective of generality to extract valid information from the attribute features of the data itself. The autoencoder consists of an encoder and a decoder. The encoder maps the input data to a particular feature space to obtain the hidden layer representation, and the decoder maps the hidden layer representation to the input space. It makes the hidden layer representation retain the features of the input data through the reconstruction of the input data. We suppose there are L layers in the autoencoder and l denotes the l-th layer of the autoencoder, then the learning representation $H^{(l)}$ of the l-th layer of the encoder part is formulated as:

$$H^{(l)} = \sigma\left(W_e{}^{(l)} H^{(l-1)} + b_e{}^{(l)}\right) \tag{1}$$

where σ is the activation function, $W_e{}^{(l)}$ is the l-th layer weight matrix, and $b_e{}^{(l)}$ is the bias of the l-th layer in the encoder.

The decoder and the encoder are symmetric, and the corresponding learning representation $H^{(l)}$ is calculated as:

$$H^{(l)} = \sigma\left(W_d{}^{(l)} H^{(l-1)} + b_d{}^{(l)}\right) \tag{2}$$

where ${W_d}^{(l)}$ is the l-th layer weight matrix and ${b_d}^{(l)}$ is the bias of the l-th layer in the decoder.

The objective function can be obtained by minimizing the loss between the original data and the reconstruction:

$$\mathcal{L}_{res} = \frac{1}{2N} \parallel X - \hat{X} \parallel_F^2 \tag{3}$$

where $\hat{X}$ is the reconstruction of the original data X and $\| \cdot \|_F$ denotes the Frobenius norm. The reconstruction loss of the autoencoder is used as part of the global loss of the model.

3.2. GAT Module

After obtaining the attribute features of the data from the autoencoder, it lacks the structural information of the data. We use the graph attention network to encode the structural features and fuse the representation obtained from the two sub-module complexes by balancing a parameter with the autoencoder. The representation of each layer learned by the autoencoder is transferred to the graph attention layer by the balancing parameter, which realizes the fusion of structure information and attribute information.

The essential component of the graph attention network is the graph attention layer, which is based on the principle of learning the implicit representation of nodes by aggregating their neighbors. Each neighbor is given a different weight in the attention mechanism to measure the importance of different neighbors. Furthermore, the high-order neighborhood information of the nodes is considered when calculating the topology. We use both the attribute values and the adjacency matrix as the input of the graph attention module. Then, the learned representation of the autoencoder is combined with the learned representation of the graph attention neural network to obtain more comprehensive information.

The learning representation $z_i^{(l)}$ of the l-th layer of node v_i in the attention layer of the graph is calculated as:

$$z_i^{(l)} = \sigma\left(\sum_{j \in N_i} a_{ij} W^{(l-1)} z_j^{(l-1)} \right) \tag{4}$$

where $z_i^{(l)}$ is the hidden layer representation of node v_i, N_i denotes the set of neighbors of node v_i, a_{ij} is the attention coefficient of node v_j to node v_i, and $W^{(l-1)}$ denotes the learnable parameter matrix. The attention coefficients are calculated from the attribute values and topological distances, respectively. In terms of attributes, it can be regarded as a single-layer neural network with weight vector attribute value concatenation.

$$c_{ij} = a^T \left[W x_i \parallel W x_j \right] \tag{5}$$

From a topological point of view, neighboring nodes influence the target node through connected edges. While the classical GAT considers only first-order neighborhoods, the graph attention layer used in this paper considers high-order neighborhoods of the graph.

$$R = \left(B + B^2 + \ldots + B^t \right) / t \tag{6}$$

where B is the transition matrix and $B_{ij} = 1/m_i$ if edges exist at nodes v_i and v_j, otherwise $B_{ij} = 0$ and m_i is the degree of node v_i. R_{ij} denotes the topological correlation of node v_i and node v_j up to t orders. Here, t can be chosen flexibly according to different datasets.

The attention coefficients are usually normalized for comparison between nodes using the softmax function. The final attention coefficient after adding the topological weights and activation functions can be expressed as:

$$a_{ij} = \frac{\exp\left(\sigma R_{ij} c_{ij}\right)}{\sum_{r \epsilon N_i} \exp(\sigma R_{ir} c_{ir})} \tag{7}$$

The representation obtained by the autoencoder is passed to the graph attention layer for each layer by balancing the parameters ε. The fused representation $\widetilde{Z}^{(l)}$ of the autoencoder, and the graph attention layer is obtained as follows:

$$\widetilde{Z}^{(l)} = (1-\varepsilon)Z^{(l)} + \varepsilon Z_{AE}^{(l-1)} \tag{8}$$

where $Z^{(l)}$ is the output of the graph attention layer and $Z_{AE}^{(l-1)}$ is the output of the autoencoder $l-1$ layer, and the final representation is obtained by fusing the two learned representations layer by layer with a balancing parameter. In this way, the hidden layer representation inherits more attributes from the attribute space of the original graph, preserving features that can be better clustered.

3.3. Feature Fusion Module

This module performs feature fusion of the node representations obtained from the graph attention module using the principle of skip connection.

Firstly, a self-correlated learning mechanism is introduced. The latent representation $Z^{(1)}$ obtained by encoding the first layer of the graph attention module is transposed and then multiplied with itself. The normalized self-correlated matrix S is obtained using the softmax function:

$$S_{ij} = \frac{e^{(Z^{(1)}(Z^{(1)})^T)_{ij}}}{\sum_{k=1}^{N} e^{(Z^{(1)}Z^{(1)T})_{ik}}} \tag{9}$$

Then, S is used as the correlation coefficient and multiplied with the node representation $\widetilde{Z}^{(l)}$ obtained from the graph attention module to calculate the node representation Z_F:

$$Z_F = S\widetilde{Z}^{(l)} \tag{10}$$

Finally, we use the softmax function for multiple classifications:

$$Z = softmax(Z_F) \tag{11}$$

The result z_{ik} denotes that the probability of the node v_i belongs to the k-th clustering centre. Z is considered as a probability distribution.

3.4. Feature Fusion Module

After unifying the above three components in a framework, it is necessary to make it oriented to the clustering task. We adopt the self-supervised learning mechanism for model optimization.

For the i-th node and the k-th clustering centre, the similarity between the embedding representation and the clustering centre is measured using the Student's t-distribution as a kernel:

$$q_{ik} = \frac{\left(1+ \parallel z_{(AE)(i)} - \mu_k \parallel / v\right)^{-(v+1)/2}}{\sum_{k'}\left(1+ \parallel z_{(AE)(i)} - \mu_{k'} \parallel / v\right)^{-(v+1)/2}} \tag{12}$$

where $z_{(AE)(i)}$ is the i-th row of $Z_{AE}^{(L)}$, μ_k is obtained by the autoencoder initialized by K-means pretraining, v is the degree of freedom of the Student's t-distribution, q_{ik} is viewed as the probability of assigning the i-th sample to the k-th clustering centre, and $Q = \left[q_{ij}\right]$ is the distribution of all samples.

In order to make the obtained embedding representation closer to the cluster center, the target distribution is calculated as:

$$p_{ik} = \frac{{q_{ik}}^2 / \sum_i q_{ik}}{\sum_{k'}({q_{ik'}}^2 / \sum_i q_{ik'})} \tag{13}$$

The values of Q in the objective distribution P are normalised by the sum of squares so that the results obtained have a high confidence level and the target function is obtained in the following form:

$$\mathcal{L}_{clu} = KL(P \parallel Q) = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}} \tag{14}$$

By minimizing the KL-divergence loss between the Q and P distributions, the target distribution P can help the autoencoder module learn a better representation of the clustering task by bringing the data closer to the cluster centres. Similarly, in order to train the graph attention neural network, the KL-divergence loss is as follows:

$$\mathcal{L}_{GAT} = KL(P \parallel Z) = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{z_{ik}} \tag{15}$$

By this way, GAT and AE optimize on the same objective, making their results converge during the training process. Since the objective of the AE module and the GAT module is to approximate the target distribution P, these two modules can supervise each other's learning.

The reconstruction loss, cluster learning loss, and graph attention neural network classification loss obtained from the autoencoder are jointly optimized, and the final loss function is:

$$\mathcal{L} = \mathcal{L}_{res} + \alpha \mathcal{L}_{clu} + \beta \mathcal{L}_{GAT} \tag{16}$$

where α is a hyperparameter to balance the cluster optimisation and local structure preservation, and β is a coefficient to control the interference of GAT on the embedding space.

The final clustering result of the nodes is the soft distribution of the distribution Z, i.e., the clustering result of i-th sample:

$$r_i = arg\max_k z_{ik} \tag{17}$$

4. Experiments

4.1. Datasets

We conduct experiments on six public datasets with the statistical information shown in Table 1, including three non-graph datasets and three graph datasets. USPS [29], HHAR, [30] and REUT [31] are non-graph datasets lacking graph structure information. For non-graph datasets, the method of constructing a KNN graph in SDCN is used for graph construction with the values of K being 3, 5, and 1, respectively. ACM, DBLP, and CITE are classical graph datasets. There are significant differences among the six datasets in average path length, clustering coefficient, and diameter.

Table 1. The details of the datasets.

Dataset	Type	Samples	Dimension	Average Path Length	Clustering Coefficient	Diameter	Classes
USPS	Image	9298	256	8.3	0.2306	43	10
HHAR	Record	10,299	561	11.1	0.2084	71	6
REUT	Text	10,000	2000	3.7	0.0005	9	4
ACM	Graph	3025	1870	5.8	0.6886	20	3
DBLP	Graph	4058	334	9.3	0.1301	28	4
CITE	Graph	3327	3703	7.0	0.1941	24	6

4.2. Experiments Setup

Baselines. We compare our method with two types of methods: AE-based clustering and GCN-based graph clustering.

- AE [28] is a deep clustering approach that performs a K-means algorithm on the representation learned by the autoencoder.

- DEC [24] designs a clustering target to guide the embedding process.
- IDEC [25] adds reconstruction loss to DEC to learn better embedding results.
- GAE and VGAE [14] are unsupervised graph embedding methods that use GCN to learn data representations.
- DAEGC [15] uses graph attention networks as encoders to learn node representations and uses clustering losses to supervise the graph clustering process.
- SDCN [18] integrates autoencoder and GCN to learn the data representations.

Parameters Setting. First, we pretrain the autoencoder to initialize the clustering center using all data with 30 iterations, and the learning rate is 0.001. For fair comparisons, the number of neural units for GAT and autoencoder is set to d-500-500-2000-10 as in the SDCN, with d being the feature dimension of the input data. The structure of nodes with two-hop neighbors is more common in graph data, and t is set to 2 for the generality of the model. α is 0.1 and β is set to 0.01 in the loss function. The value of the balance parameter ε is kept the same as that of the SDCN. To ensure the convergence of the clustering results, we trained uniformly for all datasets for 500 iterations. To prevent extreme cases, we run 10 times for each dataset. The average and standard deviation are calculated as the final results.

Evaluation Metrics. We use the four common metrics to evaluate the effect of clustering, including accuracy (ACC), normalized mutual information (NMI), adjusted rand index (ARI), and F1 score (F1). The higher value of each metric indicates a better result of clustering.

4.3. Clustering Results

The clustering results of our proposed method on the six datasets are shown in Table 2, with the bolded numbers indicating the optimal results.

Table 2. Clustering results on six datasets (mean ± std).

Dataset	Metric	AE	DEC	IDEC	GAE	VGAE	DAEGC	SDCN	CDFG
USPS	ACC	71.0 ± 0.0	73.3 ± 0.2	76.2 ± 0.1	63.1 ± 0.3	56.2 ± 0.7	73.6 ± 0.4	78.1 ± 0.2	78.3 ± 0.1
	NMI	67.5 ± 0.0	70.6 ± 0.3	75.6 ± 0.1	60.7 ± 0.6	51.1 ± 0.4	71.1 ± 0.2	79.5 ± 0.3	79.7 ± 0.1
	ARI	58.8 ± 0.1	63.7 ± 0.3	67.9 ± 0.1	50.3 ± 0.6	41.0 ± 0.6	63.3 ± 0.3	71.8 ± 0.2	72.0 ± 0.2
	F1	69.7 ± 0.0	71.8 ± 0.2	74.6 ± 0.1	61.8 ± 0.4	53.6 ± 1.1	72.5 ± 0.5	77.0 ± 0.2	77.2 ± 0.1
HHAR	ACC	68.7 ± 0.3	69.4 ± 0.3	71.1 ± 0.4	62.3 ± 1.0	71.3 ± 0.4	76.5 ± 2.2	84.3 ± 0.2	88.5 ± 0.6
	NMI	71.4 ± 1.0	72.9 ± 0.4	74.2 ± 0.4	55.1 ± 1.4	63.0 ± 0.4	69.1 ± 2.3	79.9 ± 0.1	82.7 ± 0.5
	ARI	60.4 ± 0.9	61.3 ± 0.5	62.8 ± 0.5	42.6 ± 1.6	51.5 ± 0.7	60.4 ± 2.2	72.8 ± 0.1	77.5 ± 0.5
	F1	66.4 ± 0.3	67.3 ± 0.3	68.6 ± 0.3	62.6 ± 1.0	71.6 ± 0.3	76.9 ± 2.2	82.6 ± 0.1	88.3 ± 0.8
REUT	ACC	74.9 ± 0.2	73.6 ± 0.1	75.4 ± 0.1	54.4 ± 0.3	60.9 ± 0.2	65.6 ± 0.1	77.2 ± 0.2	78.0 ± 0.3
	NMI	49.7 ± 0.3	47.5 ± 0.3	50.3 ± 0.2	25.9 ± 0.4	25.5 ± 0.2	30.6 ± 0.3	50.8 ± 0.2	53.5 ± 0.5
	ARI	49.6 ± 0.4	48.4 ± 0.1	51.3 ± 0.2	19.6 ± 0.2	26.2 ± 0.4	31.1 ± 0.2	55.4 ± 0.4	56.8 ± 0.6
	F1	61.0 ± 0.2	64.3 ± 0.2	63.2 ± 0.1	43.5 ± 0.4	57.1 ± 0.2	61.8 ± 0.1	65.5 ± 0.1	65.6 ± 0.0
ACM	ACC	81.8 ± 0.1	84.3 ± 0.8	85.1 ± 0.5	84.5 ± 1.4	84.1 ± 0.2	86.9 ± 2.8	90.5 ± 0.2	90.2 ± 0.3
	NMI	49.3 ± 0.2	54.5 ± 1.5	56.6 ± 1.2	55.4 ± 1.9	53.2 ± 0.5	56.2 ± 4.2	68.3 ± 0.3	67.6 ± 0.4
	ARI	54.6 ± 0.2	60.6 ± 1.9	62.2 ± 1.5	59.5 ± 3.1	57.7 ± 0.7	59.4 ± 3.9	73.9 ± 0.4	73.2 ± 0.6
	F1	82.0 ± 0.1	84.5 ± 0.7	85.1 ± 0.5	84.7 ± 1.3	84.2 ± 0.2	87.1 ± 2.8	90.4 ± 0.2	90.1 ± 0.3
DBLP	ACC	51.4 ± 0.4	58.2 ± 0.6	60.3 ± 0.6	61.2 ± 1.2	58.6 ± 0.1	62.1 ± 0.5	68.1 ± 1.8	74.8 ± 1.1
	NMI	25.4 ± 0.2	29.5 ± 0.3	31.2 ± 0.5	30.8 ± 0.9	26.9 ± 0.1	32.5 ± 0.5	39.5 ± 1.3	41.6 ± 1.6
	ARI	12.2 ± 0.4	23.9 ± 0.4	25.4 ± 0.6	22.0 ± 1.4	17.9 ± 0.1	21.0 ± 0.5	39.2 ± 2.0	45.7 ± 1.8
	F1	52.5 ± 0.4	59.4 ± 0.5	61.3 ± 0.6	61.4 ± 2.2	58.7 ± 0.1	61.8 ± 0.7	67.7 ± 1.5	73.7 ± 1.3
CITE	ACC	57.1 ± 0.1	55.9 ± 0.2	60.5 ± 1.4	61.4 ± 0.8	61.0 ± 0.4	64.5 ± 1.4	66.0 ± 0.3	69.2 ± 0.9
	NMI	27.6 ± 0.1	28.3 ± 0.3	27.2 ± 2.4	34.6 ± 0.7	32.7 ± 0.3	36.4 ± 0.9	38.7 ± 0.3	42.2 ± 0.9
	ARI	29.3 ± 0.1	28.1 ± 0.4	25.7 ± 2.7	33.6 ± 1.2	33.1 ± 0.5	37.8 ± 1.2	40.2 ± 0.4	44.4 ± 1.1
	F1	53.8 ± 0.1	52.6 ± 0.2	61.6 ± 1.4	57.4 ± 0.8	57.7 ± 0.5	62.2 ± 1.3	63.6 ± 0.2	62.8 ± 0.6

From the clustering results in Table 2, we notice that the model with the addition of the graph attention layer and feature fusion module performs well on most of the datasets. It is

due to GAT's better representation learning ability compared to GCN in the learning graph structure. In addition, the feature fusion module further enhances the features. The model can learn better about neighborhood information after adding the graph attention layer and also considers the high-order neighborhood information of the nodes. The feature fusion module makes a secondary fusion of structural and attribute features.

For the three non-graph datasets, USPS, HHAR, and REUT, all metrics perform well. As far as the average is concerned, compared to the SDCN method, there is a more significant improvement in HHAR. Our approach improves 4.2% on ACC, 2.8% on NMI, 4.7% on ARI, and 5.7% on F1. The HHAR dataset has a longer average path length and network diameter compared with the other two non-graph datasets. We consider the high-order neighborhood information and learn more informative node representations. For REUT, there is a more significant improvement on ACC, NMI, and ARI than USPS. From the viewpoint of network features, REUT has a smaller clustering coefficient and higher feature dimension than USPS.

For DBLP, the improvement is 6.7% on ACC, 2.1% on NMI, 6.5% on ARI, and 6% on F1. For CITE, the improvement is 3.2% on ACC, 3.5% on NMI, and 4.2% on ARI. For DBLP, there is a greater improvement on ACC and ARI than CITE because DBLP has a longer average path length and network diameter than CITE, and a smaller clustering coefficient. The effect is not improved for the ACM dataset because the network clustering coefficient is higher and aggregation is higher than the other two graph datasets. There will not be much difference in learning between GCN and GAT. The average path length and network diameter are smaller for the ACM dataset, and there will not be much difference considering high-order information.

In all metrics, our method significantly improved the graph dataset DBLP compared to the non-graph dataset HHAR. In other words, the model performs better for the data with graph structure than the data constructing the KNN graph. Since the edges in the KNN graph are not real and there is some noise, it is necessary to construct an effective KNN graph to improve the model's effectiveness. In terms of the characteristics of the network, the model proposed in this paper is more suitable for networks with longer average path length and network diameter and smaller clustering coefficient.

4.4. Ablation Study

We conduct an ablation study to evaluate the effectiveness of the GAT module and the feature fusion module. The results are reported in Figure 2.

Analysis of the GAT module. From Figure 2, we can see that CDFG-w has a 0.1% to 4.6% improvement over the SDCN method, which shows the effectiveness of the graph attention network module. For non-graph datasets, the CDFG-w method performs better on HHAR than the other two non-graph datasets. For graph datasets, the CDFG-w method performs better on DBLP and CITE. The improvement is greater for networks with longer average path lengths and network diameters and smaller clustering coefficients, i.e., HHAR, DBLP, and CITE, which means that the graph attention layer considering high-order neighborhood information is more effective for this type of network.

Analysis of the feature fusion module. The CDFG has a 0.1% to 4.3% improvement over the CDFG-w method, which demonstrates the effectiveness of the feature fusion module. We can find that CDFG method performs better on graph datasets than non-graph datasets. For graph datasets, the CDFG method performs better on DBLP than on CITE. Similarly, the feature fusion module is more effective for networks with long average path length and network diameter and a small clustering coefficient. In addition, the feature fusion module improves the dataset with actual graph structure to a greater extent than the KNN graph because the dataset with actual graph structure reflects the characteristics of the data more accurately. Moreover, the model learns better after further enhancement of the features.

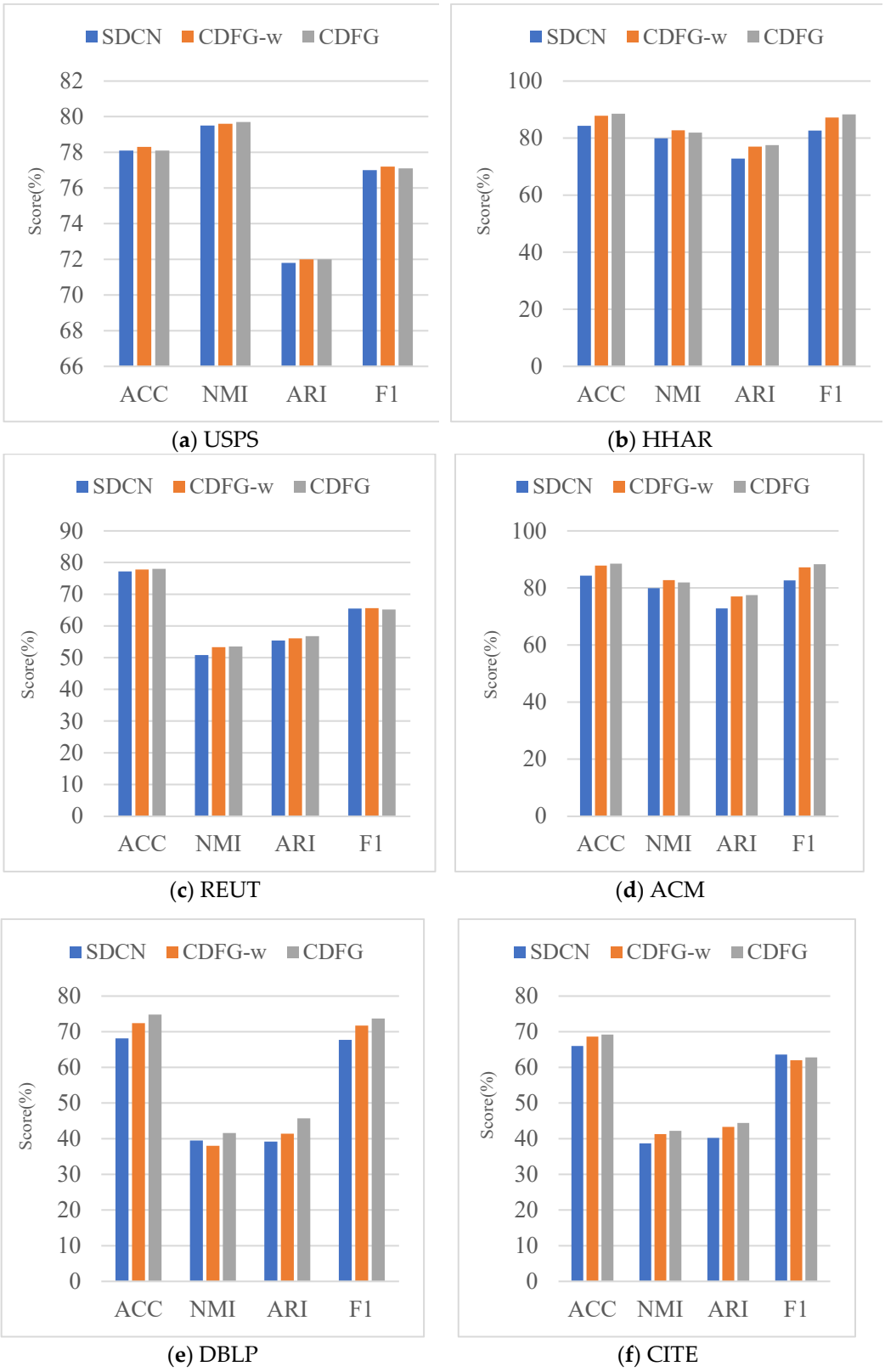
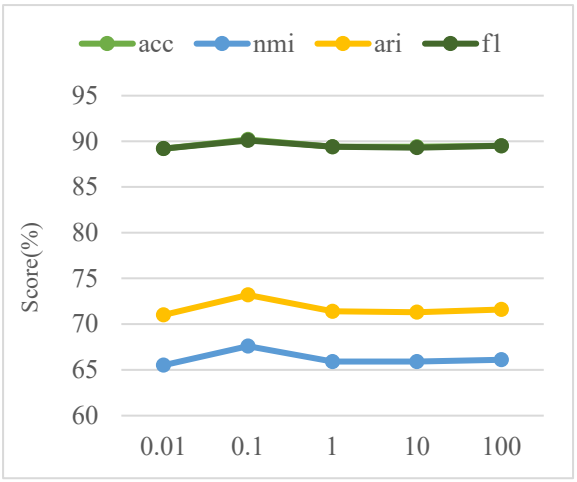


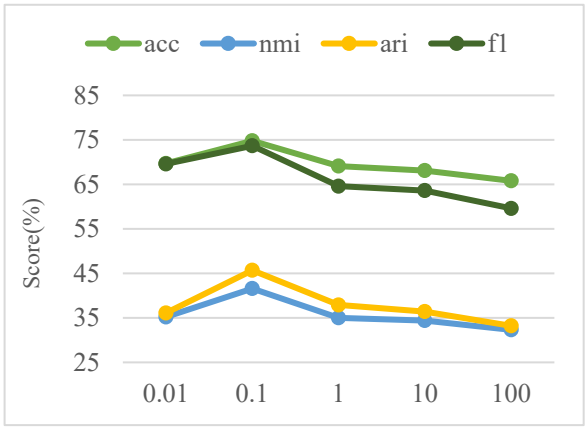
Figure 2. Comparison of experimental ablation effects of graph attention network module and feature fusion module. SDCN is the method using only graph convolution network and autoencoder. CDFG-w is the method with the addition of a graph attention network module. CDFG is the method with the addition of a feature fusion module.

4.5. Parameter α Sensitivity Analysis

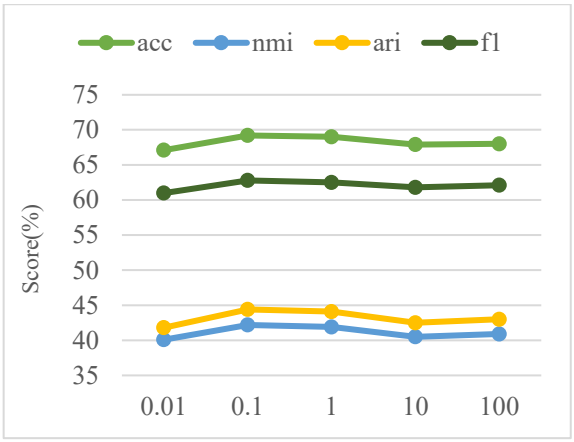
We conduct parameter sensitivity analysis of α in the loss function, which is an important parameter for balancing the clustering loss and other losses. To evaluate the effect of parameter α on model performance, the CDFG model is experimented with three graph datasets, including ACM, DBLP, and CITE, by setting $\alpha = [0.01, 0.1, 1, 10, 100]$ with fixed $\beta = 0.01$. We do not discuss β here because the CDFG method is insensitive to β. We ran our method 10 times independently for each dataset and report the average results. The results of each metric are shown in Figure 3.



(**a**) ACM



(**b**) DBLP



(**c**) CITE

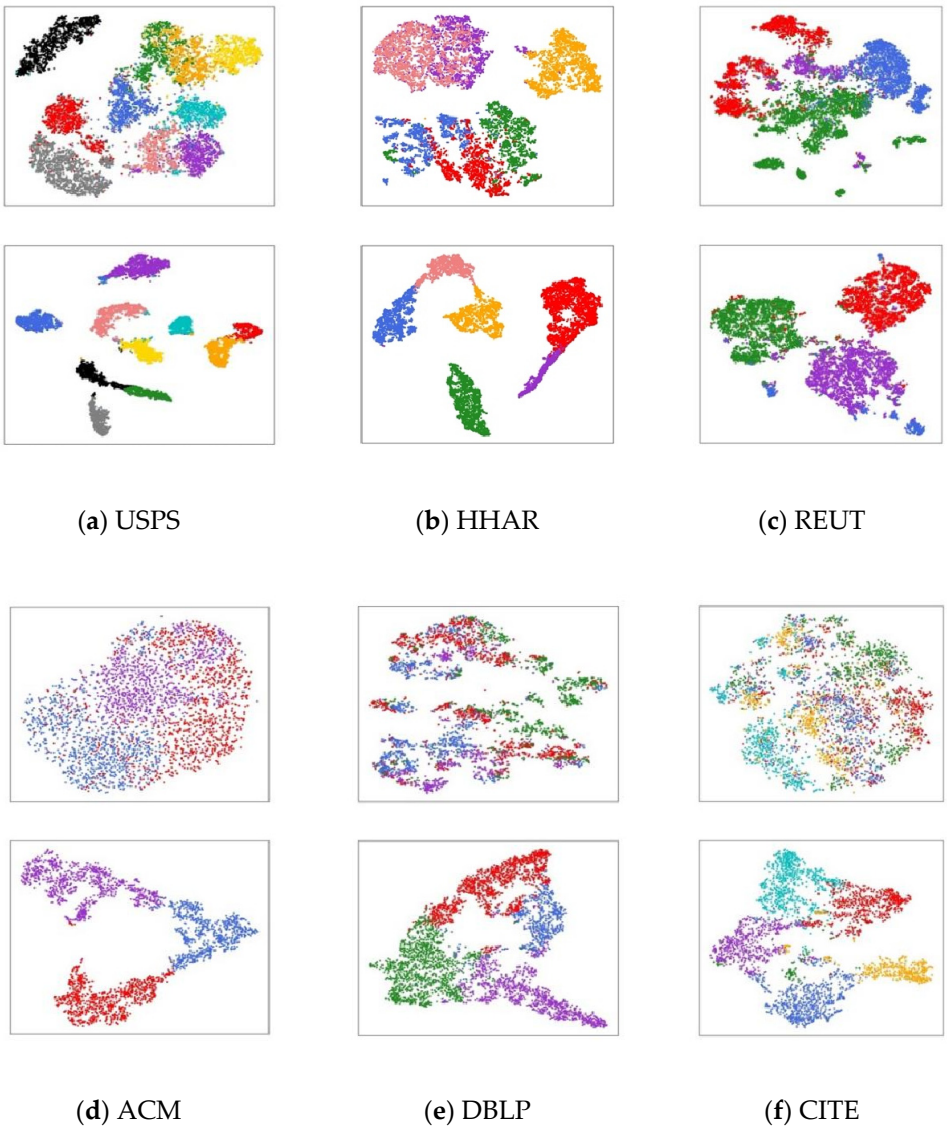
Figure 3. The influence of parameter α on the model effect.

From Figure 3, we can observe that the parameter α has a certain influence on the clustering effect, and all three datasets reach the optimal value when $\alpha = 0.1$. For ACM and CITE, the trend of changes is gentler with the parameter α. However, for DBLP, the variation is more significant compared to ACM and CITE. From the viewpoint of network characteristics, networks with longer average path length and network diameter and smaller clustering coefficient are more sensitive to the change of parameter α.

4.6. Network Visualisation

In order to verify the validity of the model more intuitively, we conduct an experiment visualizing the clustering results. For the original data, we use PCA firstly to reduce the dimension as the embedding representation obtained by CDFG, and use the t-SNE [32] method for 2D visualization. For the learned embedding results, we directly visualize the

data samples in 2D space by using the t-SNE method. The visualization results are shown in Figure 4. The data points of the same color indicate the same category. The clearer the boundary between clusters composed of sample points of different colors, the better the clustering results.



(**a**) USPS (**b**) HHAR (**c**) REUT

(**d**) ACM (**e**) DBLP (**f**) CITE

Figure 4. The visualization results on six datasets. The first and second rows correspond to the original data and CDFG, respectively.

The visualization results show that the original data distribution is more scattered, and the boundary is more confusing, while after the representation learning of the model, the same category is more aggregated, and the boundary between categories is clearer, which can verify the validity of the CDFG.

5. Conclusions

In this paper, we propose a community detection model fusing the graph attention layer and the autoencoder. The innovation of the model is that it fuses the autoencoder and the graph attention network with high-order neighborhood information for the first time. In addition, the feature fusion module is designed to achieve secondary fusion. The graph attention layer learns structural features better by considering the importance of neighborhood information. Adding high-order neighborhood information is more robust and effective for networks with longer mean paths and network diameters. The autoencoder learns

the data characteristics, and a balance parameter fuses the two parts. The feature fusion module makes a secondary fusion of structural and attribute features. The experimental results show that the proposed model performs better on network datasets with longer mean paths and network diameters and smaller clustering coefficients. Compared with various state-of-the-art methods, CDFG has better performance for community detection.

This is because the use of the graph attention network calculates the influence weight of the neighborhood node on the target node through the attention mechanism, taking into account the difference of the influence of different neighborhood nodes. At the same time, high-order neighborhood information is added to the graph attention layer. The final node representation contains more information. On the other hand, the feature fusion module further fuses the structure information and the attribute information, and enhances the feature.

Currently, the model only supports non-overlapping community detection and it is a research direction to make model support overlapping community detection in the future. How to improve the efficiency of the algorithm while keeping the excellent performance of the model is also worth studying.

Author Contributions: Conceptualization, R.G. and W.W.; methodology, R.G. and J.Z.; software, Q.B.; validation, Q.B. and X.C.; formal analysis, W.W.; investigation, R.G. and J.Z.; resources, R.G. and J.Z.; data curation, W.W.; writing—original draft preparation, Q.B.; writing—review and editing, W.W.; visualization, X.C.; supervision, R.G.; project administration, W.W.; funding acquisition, R.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the People's Livelihood Project of the Key R&D Program of Hebei Province (20375701D) and the Central Guidance on Local Science and Technology Development Fund of Hebei Province (226Z1808G).

Data Availability Statement: Code and data are available at https://github.com/cuttercn/CDFG (accessed on 23 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bo, Y.; Liu, D.; Liu, J. Discovering Communities from Social Networks: Methodologies and Applications. In *Handbook of Social Network Technologies & Applications*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 331–346.
2. Saturuli, V.; Wu, Y.; Zheng, X.; Qian, Y.; Wichers, B.; Dai, Q.; Tang, G.M.; Jiang, J.; Lin, J. SimClusters: Community-Based Rep-resentations for Heterogeneous Recommendations at Twitter. In Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, 6–10 July 2020; ACM: New York, NY, USA; pp. 3183–3193.
3. Keyvanpour, M.R.; Shirzad, M.B.; Ghaderi, M. AD-C: A new node anomaly detection based on community detection in social networks. *Int. J. Electron. Bus.* **2020**, *15*, 199. [CrossRef]
4. Saidi, F.; Trabelsi, Z.; Ghazela, H.B. A novel approach for terrorist sub-communities detection based on constrained evidential clustering. In Proceedings of the 12th International Conference on Research Challenges in Information Science (RCIS), Nantes, France, 29–31 May 2018.
5. Amini, A.A.; Chen, A.; Bickel, P.J.; Levina, E. Pseudo-likelihood methods for community detection in large sparse networks. *Ann. Stats* **2013**, *41*, 2097–2122. [CrossRef]
6. Girvan, M.; Newman, M.E. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef] [PubMed]
7. Chunaev, P. Community detection in node-attributed social networks: A survey. *Comput. Sci. Rev.* **2020**, *37*, 100286. [CrossRef]
8. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [CrossRef] [PubMed]
9. Cao, S.; Lu, W.; Xu, Q. Deep neural networks for learning graph representations. *Proc. AAAI Conf. Artif. Intell.* **2016**, *30*, 1145–1152. [CrossRef]
10. Velikovi, P.; Cucurull, G.; Casanova, A.; Remero, A.; Lio, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2018**, arXiv:1710.10903.
11. Su, X.; Xue, S.; Liu, F.; Wu, J.; Zhou, C.; Hu, W.; Paris, C.; Nepal, S.; Jin, D.; Sheng, Q.Z. A Comprehensive Survey on Community Detection with Deep Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–21. [CrossRef]
12. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
13. Kip, F.T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
14. Kip, F.T.N.; Welling, M. Variational Graph Autoencoders. *arXiv* **2016**, arXiv:1611.07308.

15. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed Graph Clustering: A Deep Attentional Embedding Approach. *arXiv* **2019**, arXiv:1906.06532.
16. Wang, C.; Pan, S.; Yu Celina, P.; Hu, R.; Long, G.; Zhang, C. Deep neighbor-aware embedding for node clustering in attributed graphs. *Pattern Recognit.* **2022**, *122*, 108230. [CrossRef]
17. Pan, S.; Hu, R.; Fung, S.F.; Long, G.; Jiang, J.; Zhang, C. Learning Graph Embedding with Adversarial Training Methods. *IEEE Trans. Cybern.* **2020**, *50*, 2475–2487. [CrossRef]
18. Bo, D.; Wang, X.; Shi, C.; Zhu, M.; Lu, E.; Cui, P. Structural Deep Clustering Network. In Proceedings of the Web Conference (WWW 20), Taipei, Taiwan, 20–24 April 2020; pp. 1400–1410.
19. Tu, W.; Zhou, S.; Liu, X.; Guo, X.; Cai, Z.; Zhu, E.; Cheng, J. Deep Fusion Clustering Network. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; pp. 1–10.
20. Kim, D.; Oh, A. How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision. *arXiv* **2021**, arXiv:2204.04879.
21. Liao, H.; Hu, J.; Li, T.; Du, S.; Peng, B. Deep linear graph attention model for attributed graph clustering. *Knowl. Based Syst.* **2022**, *246*, 108665. [CrossRef]
22. Dong, Y.; Wang, Z.; Du, J.; Fang, W.; Li, L. Attention-based hierarchical denoised deep clustering network. *World Wide Web* **2022**, prepublish. [CrossRef]
23. Zhao, Q.; Ma, H.; Guo, L.; Li, Z. Hierarchical attention network for attributed community detection of joint representation. *Neural Comput. Appl.* **2022**, prepublish. [CrossRef]
24. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised Deep Embedding for Clustering Analysis. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 478–487.
25. Guo, X.; Gao, L.; Liu, X.; Yin, J. Improved Deep Embedded Clustering with Local Structure Preservation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, 19–25 August 2017; pp. 1753–1759.
26. Huang, X.; Hu, Z.; Lin, L. Deep clustering based on embedded auto-encoder. *Soft Comput.* **2021**, *prepublish*. [CrossRef]
27. Peng, Z.; Jia, Y.; Liu, H.; Hou, J.; Zhang, Q. Maximum Entropy Subspace Clustering Network. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 2199–2210. [CrossRef]
28. Hinton, G.E.; Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
29. Cui, Y.L.; Matan, O.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jacket, L.D.; Baird, H.S. Handwritten zip code recognition with multilayer networks. In Proceedings of the International Conference on Pattern Recognition, Atlantic City, NJ, USA, 16–21 June 1990; IEEE: Piscataway, NJ, USA.
30. Stisen, A.; Blunck, H.; Bhattacharya, S.; Prentow, T.S.; Kjargaard, M.B.; Den, A.K.; Sonne, T.; Jensen, M.M. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In Proceedings of the Acm Conference on Embedded Networked Sensor Systems, Seoul, Korea, 1–4 November 2015; ACM: New York, NY, USA, 2015.
31. Lewis, D.D.; Yang, Y.; Rose, T.G.; Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.* **2004**, *5*, 361–397.
32. van der Maaten, L.; Hinton, G. Visualizing data using t-sne. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.