symmetry

MDPI

Article

Attribute Network Representation Learning with Dual Autoencoders

Jinghong Wang [1,2,3], Zhixia Zhou [1], Bi Li [4] and Mancai Wu [5,*]

[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 & Security, Shijiazhuang 050024, China
[3] Hebei Key Laboratory of Network and Information Security, Hebei Normal University, Shijiazhuang 050024, China
[4] Business School, Hebei Normal University, Shijiazhuang 050024, China
[5] Hebei Polytechnic Institute, Shijiazhuang 050020, China
* Correspondence: xiaozhang@hbgcjsxy.com

Abstract: The purpose of attribute network representation learning is to learn the low-dimensional dense vector representation of nodes by combining structure and attribute information. The current network representation learning methods have insufficient interaction with structure when learning attribute information, and the structure and attribute information cannot be well integrated. In this paper, we propose an attribute network representation learning method for dual-channel autoencoder. One channel is for the network structure,and adopting the multi-hop attention mechanism is used to capture the node's high-order neighborhood information and calculate the neighborhood weight; The other channel is for the node attribute information, and a low-pass Laplace filter is designed to iteratively obtain the attribute information in the neighborhood of the node. The dual-channel autoencoder ensures the learning of structure and attribute information respectively. The adaptive fusion module is constructed in this method to increase the acquisition of important information through the consistency and difference constraints of two kinds of information. The method trains encoders by supervising the joint reconstruction of loss functions of two autoencoders. Based on the node clustering task on four authentic open data sets, and compared with eight network representation learning algorithms in clustering accuracy, standardized mutual information and running time of some algorithms, the experimental results show that the proposed method is superior and reasonable.

Keywords: attribute networks; network representation learning; autoencoders; interactive learning; attention mechanisms

Citation: Wang, J.; Zhou, Z.; Li, B.; Wu, M. Attribute Network Representation Learning with Dual Autoencoders. *Symmetry* **2022**, *14*, 1840. https://doi.org/10.3390/sym14091840

Academic Editors: Jun Wu and José Carlos R. Alcantud

Received: 26 June 2022
Accepted: 31 August 2022
Published: 5 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

The goal of network representation learning, also known as network embedding or graph embedding, is to learn a data representation that represents the nodes in a network as a low-dimensional, dense vector form while maximizing the preservation of information in the network [1–3]. The learned vectors are used in machine learning algorithms to accomplish downstream network analysis tasks such as link prediction, node clustering, and recommendation systems [4–7].

In recent years, with the rapid development of the Big Data era, the number of data in various fields has increased and the form of data has become more complex [8–10]. In addition to text, audio, image, and video data, information networks represent another natural and complex data structure representing several entities and relationships. The network formed by these data exists in the modality of a graph in computers, and a wide variety of real-world data in business, science, and engineering are captured in the form of information networks [11], for example, social networks constituted by users and user relationships in social platforms. citation networks constituted by papers and cross-citations of papers in

academic websites; urban traffic networks [5], protein interaction networks [6], etc. These graph data have a complex structure and attribute information that can be adapted to learning tasks in multiple domains [12–16], and in the network, the key aspect of mining the potential information data is network representation learning. Network representation learning learns the vector representation of each node from the original graph structure data so that the obtained representation vectors have representation and inference capabilities in low-dimensional space [17–20].

Traditional network representation learning methods are dedicated to retaining the topological structure information of the network [1–3], while the vast majority of real networks contain rich attribute information, such as hobbies, addresses, and ages of users in social networks, authors, keywords, and research fields of articles in citation networks, and such networks are called attribute networks [12,15,21]. When the network is highly sparse, the attribute information of nodes is important auxiliary information for network representation and helps to learn better network representation; if two nodes do not have a similar network structure, but have the same attributes, they should also have similar representation in the final representation space [22,23].

The traditional network representation learning method known as the Deepwalk [24] algorithm is based on random wandering, where all vertices in the graph are first labeled, the starting vertex is randomly selected, the path length is specified, and then the random wandering starts and the node representation is obtained in the generated wandering sequence with the help of the Skip-Gram model. Since its way of selecting the next node in the random wandering sequence is uniformly randomly distributed, the sampling process will repeat the sampling of the central node. Node2vec [25] puts restrictions on the neighborhood of vertices based on random wandering by adding a biased random wandering strategy to capture the structure of the context, i.e., defining a transfer probability between wandering strategies so that different neighborhoods can be explored effectively. The struc2vec [26] approach considers that nodes with similar network space structure also have high similarity and thus perform random wandering based on hierarchical weighted graphs. LINE [27] preserves the first-order and second-order nearest neighbors of the network by marginalized random wandering while preserving the local and global network structure of the network. SDNE [28] uses deep learning techniques in representation learning, combining autoencoders and Laplacian feature mapping to preserve the first-order and second-order similarity of the network structure.

Given that most real networks are rich in attribute information, combining node attribute information to learn vector representations of nodes can solve the problem of sparse network structure while better preserving the information of the original network. TADW [29] first proposed combining textual information of nodes into representation learning through matrix decomposition, reflecting a better performance than the Deepwalk [24] algorithm. The traditional Deepwalk method based on random wandering is proved and extended, and it is proved that its essence is an equivalent matrix decomposition method, so by adding the textual feature information matrix in the decomposition process, the method has both structure and textual information for representation learning, but the problem is that the computation and storage of the two correlation matrices for structure and textual information are less efficient and not suitable for large networks. The model of AANE [30] also integrates topological and attribute information in the network based on matrix decomposition, but the model decomposes the optimization process into multiple subproblems working in parallel, which improves the efficiency of the algorithm. ASNE [31] separates the structure and attributes of the topological nodes perform a layer of embedding, and then weighted stitching is inputted to a deep neural network to achieve network representation learning. DeepEmLAN [32] smoothly projects different types of attribute information into the same semantic space by a deep attention model while maintaining its topology. Deep autoencoder-based GAE and variational autoencoder-based VGAE [33] integrate and map the topology and attribute information into the same semantic space, using the middle layer as a vector representation of the nodes. GraphSAGE [34] extends the traditional graph con-

volutional neural network to generate a vector representation by aggregating information of multi-order neighbor nodes with node attribute information. GraphRNA [35] generates random wandering sequences on node attribute information, based on which a recurrent neural network framework is designed to learn node representations. ARNL [36] combines encoder and Skip-Gram models to jointly learn structurally and attribute representations. GCN [37] uses first-order information based on local spectral convolution filters to aggregate neighbor attribute information. GAT [38] adds node attribute information to neighbors to assign different weights and learn neighbor weights based on importance.

Existing methods for learning network representations combining structural and node attribute information continue to suffer from several problems:

- Non-linearity: Most representation learning methods are shallow, but the structure and attribute information in the network are highly non-linear, and it is difficult to capture the highly non-linear topology and node attribute information in the network.
- Interactivity: Topological structure and attribute information learning are complementary to network representation, and both should be ensured to learn interactively when performing learning, and the consistency and interactivity of structure and attribute information cannot be learned well.
- Multimodality: structure and node attribute information are two different kinds of information; one should know how to efficiently fuse the two kinds of information and learn important information in structure and attributes adaptively in downstream tasks.

To address the above issues, this paper proposes a dual autoencoders' attribute network representation learning method that can fuse local and global structure information and node attribute information in network representation learning to obtain a better node representation. Specifically, the multi-hop attention mechanism is used to capture the higher-order neighborhood information of the nodes [39], and the attribute autoencoder part is designed with a low-pass Laplace filter to process the attribute matrix because the low-frequency signals are smoother and the signal values of neighboring nodes are more correlated and have stronger similarity, while the high-frequency graph signals have much more drastic changes and the differences between the signal values of neighboring nodes are more significant. Therefore, a low-pass Laplace filter is designed to process the attribute information.

The main contributions of this paper can be summarized as follows:

- In this paper, we propose the dual autoencoders network representation learning (DANRL) method, which uses the neighbor weight analysis strategy of a multi-hop attention mechanism to assign different weights to nodes based on node neighbor attribute information, capture node high-order neighborhood information, and obtain node structure embedding representation.
- A low-pass Laplace smoothing filter is designed to process the attribute matrix, remove the high-frequency signals, make the nodes close to each other in the neighborhood closer, iteratively obtain the attribute information of important neighbor nodes, realize the complementarity and mutual constraint of two kinds of information, and obtain the attribute embedding representation after adaptive decoding.
- The adaptive learning strategy is proposed to design common parameter sharing and the importance of adaptive learning structure and properties by optimizing the joint reconstruction loss of two autoencoders.
- We experiment on four real-world datasets and compare them with eight network representation learning methods, and the experimental results show the advantages and rationality of DANRL for the node-clustering task.

2. Related Work

2.1. Network Representation Learning

Real-world networks are essentially graph-structured, in which the vertices in the network are the nodes in the graph, and the relational links between the vertices in the network are the edges connecting nodes to nodes in the graph structure; therefore, network representation learning is also called graph representation learning, network embedding, or graph embedding.

Traditional network representation learning methods focus on capturing linear and shallow topological information in the network, while realistic complex networks exist with high sparsity and nodes that contain rich attribute information, and combining attribute information to learn low-dimensional vector representations of nodes makes the learned vectors better represent the information of the original network and gives them representation and inference capabilities in low-dimensional space [40–42]. In the introduction, we mentioned traditional topology-based representation learning methods and representation learning methods that incorporate node attribute information. In this paper, we classify network representation learning methods into two major categories: one that is not based on graph neural networks and retains shallow network information and the other that is based on graph neural networks and captures network structure and attribute information. The difference between the graph neural network-based approach and the non-graph neural network-based approach is that the graph neural network-based approach can be generalized to invisible nodes, can learn vector representations using the attribute information of the nodes, and can also capture highly nonlinear information about the network structure and attributes [43]. Although the graph neural network-based approach can exploit the node attribute information in the network, it also introduces problems such as over-smoothing, which reduces the robustness of the approach. In addition, graph neural networks lack interpretation in the fusion of the structure and attribute information and do not guarantee interactivity and consistency when performing the fusion of the two types of information, which keeps the two types of information mutually constrained when learning.

2.2. Auto-Encoder

The autoencoder consists of an artificial neural network composed of an encoder and a decoder, where the encoder maps the graph structure data in the input space to the potential representation space, and the decoder then maps the data in the potential representation space to the reconstruction space, making the graph structure in the reconstruction space similar to the graph structure in the original input space. Encoder–decoder architecture combined with deep neural networks completes the reconstruction of complex networks, as shown in Figure 1 [44]. The structure of the graph autoencoder facilitates multimodal information fusion for efficient joint representation learning, and the basic encoder contains three layers, the input layer, the hidden layer, and the output layer:

$$h_i = \sigma(W_i^{(1)} + b^{(1)}) \tag{1}$$

$$\hat{x}_i = \sigma(W^{(2)} h_i + b^{(2)}) \tag{2}$$

Here, x_i denotes the first i input data, h_i is the hidden layer representation of the encoder, $\widehat{x}_i$ is the data reconstructed by the decoder, and $\theta = \left\{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\right\}$ is the parameter of the autoencoder model and is the nonlinear activation function. The encoder and decoder can be set up with different dimensionality-reduction methods depending on the task.

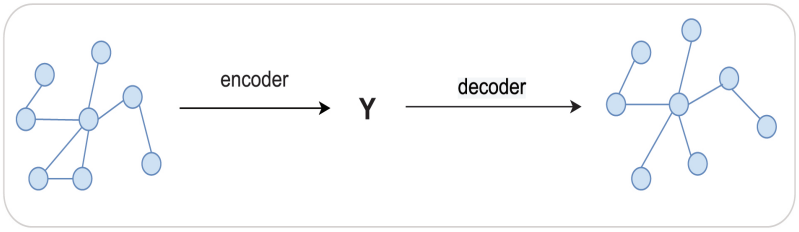

Figure 1. Encoder architecture.

In the autoencoder, the loss function is usually taken as the squared error loss function or the cross-entropy loss function. For the input sample and reconstructed sample, the squared error loss function is:

$$J\left(X, \widehat{X}\right) = \frac{1}{2}\sum_{i=1}^{n} \|\widehat{x}_i - x_i\|_2^2 \tag{3}$$

The cross-entropy loss function is:

$$J\left(X, \widehat{X}\right) = -\sum_{i=1}^{n} [x_i \log(\widehat{x}_i) + (1 - x_i)\log(1 - \widehat{x}_i)] \tag{4}$$

3. Attribute Network Representation with Dual Autoencoders

3.1. Related Concepts

In order to better describe the proposed model and its specific algorithm, this paper first gives a description of relevant concepts in the problem and the main symbolic representation involved in the algorithm model, and the main symbols are shown in Table 1.

Table 1. Symbol definition.

Symbol	Meaning
V	Node collection
E	Collection of edges between nodes
A	Node attribute collection
n	Number of network nodes, $\|V\|$
m	Number of node attributes, $\|A\|$
X	Attribute matrix, $n \times m$
M	Adjacency matrix
e_{ij}	Weight between nodes v_i and v_j
v_i	Node labeled i, $v_i \in V$
d	The node ultimately represents the dimension of the vector, $d >> t$
y_i	Representation vector of node v_i
Y	Node representation vector matrix

In the real world, networks can be represented as graph structures, such as social networks, citation networks, etc. In real networks, nodes usually have a series of attribute information, which forms vectors related to nodes. For example, in social networks, a node represents a user, and the user's relevant attribute information may include gender, age, address, friends, etc. Therefore, in the research on network representation learning, an network with rich attribute information in nodes is called an attribute network. An attribute network can be interpreted in a symbolic way as follows: given a network $G = (V, E, A)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes in the network, the number of nodes is n; $E = \{e_{ij}\}$ is the set of neighboring edges between nodes in the network; $A = \{a_1, a_2, \cdots, a_m\}$ is the set of node attributes; and the number is m. The proximity of attributes between each node pair(v_i, v_j) is determined by the similarity between the attribute vector x_i of the node v_i and the attribute vector x_j of the node v_j .

The purpose of attribute network representation learning is to use the network structure and node attribute information to learn a mapping function on a given attribute network and map the nodes in the network to a low-dimensional vector space so that the nodes with similar structures and attributes are close to each other in the low-dimensional space, and then use machine learning methods to solve the downstream tasks in network analysis, such as node clustering and link prediction. The attribute network represents the learning process, as shown in Figure 2. The adjacency matrix and attribute matrix of the attribute network are combined to learn the final representation vector of nodes. The low-dimensional representation vector is applied in the machine learning algorithm to solve the downstream tasks of network analysis, such as node clustering and link prediction. To use a symbolic expression to explain attribute network representation learning, for a given attribute network, learn the mapping function $f : v_i \rightarrow v_j \in Y^d$ and $i \in V$, where the mapping is associated with the structure and attribute information of each node, where d is the dimension of the final representation vector of nodes.

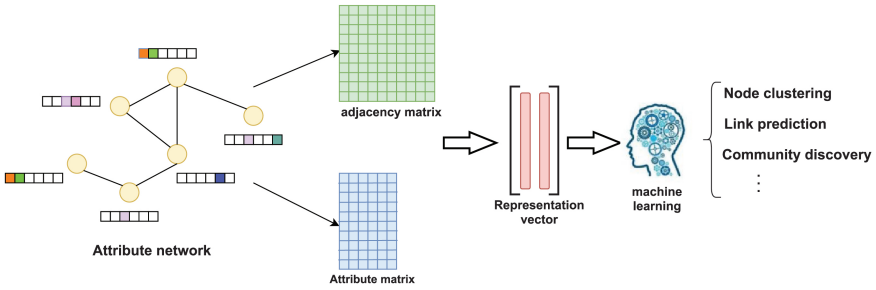

Figure 2. Attribute network represents learning process. Combined with the network structure and attribute information, the low dimensional representation vector of nodes is learned, and the machine learning algorithm is used to solve the downstream tasks of network analysis. In the figure, circular dots represent network nodes, and square lattices of different colors beside nodes represent different attribute information.

The first-order proximity in a network is the local pairwise proximity between two vertices. For two vertices v_i and v_j , if there is a directly connected edge between the two vertices, there is first-order proximity between the vertices v_i and v_j , and the weight e_{ij} of the edge is the first-order proximity of the two vertices; otherwise, there is no first-order proximity between the two vertices.

In the real network, the direct connection between the two vertices observed only accounts for a small proportion. Even though the two vertices are very similar in nature, they are not directly connected, then the first-order proximity is zero, these vertices will be lost in the similarity measurement. Therefore, single first-order proximity is not enough to preserve the network structure. High-order proximity is used to describe the similarity of neighborhood network structure between vertices, which complements the first-order proximity and retains the global structure of the network. If $N_i = (e_{i,1}, e_{i,2}, \cdots, e_{i,n})$ denotes the first-order proximity between vertex v_i and v_j all other vertices, then the higher-order proximity of vertices v_i and v_j is determined by the similarity of N_i and N_j.

3.2. Representation Learning Method for Dual Autoencoders

The method proposed in this paper is based on dual autoencoders for attribute network representation learning (DANRL), which deeply excavates the internal relationship between structure and node attributes and uses a dual-channel autoencoder to learn the network structure and attribute information (the autoencoder of structure and the autoencoder of attribute). One channel autoencoder uses the adjacency matrix of the network to calculate the edge weight between nodes, adopts the multi-hop attention mechanism to capture the high-order neighbor information of nodes, and learns the local and global structure of the network, and obtains the structure representation vector. The other channel autoencoder uses the node attribute matrix and adjacency matrix to design a low-pass

Laplace filter. Based on the network structure, the attribute information of neighboring nodes in the neighborhood of the target node is iteratively aggregated to obtain the attribute representation vector. The dual autoencoders fully learn the network structure and attribute information and then, through adaptive fusion, input the decoder to reconstruct and obtain the reconstruction matrix. This method constructs a training set by selecting highly similar or dissimilar node pairs and monitors the joint loss function training encoder of the structural autoencoder and attribute autoencoder. Its overall framework is shown in Figure 3.

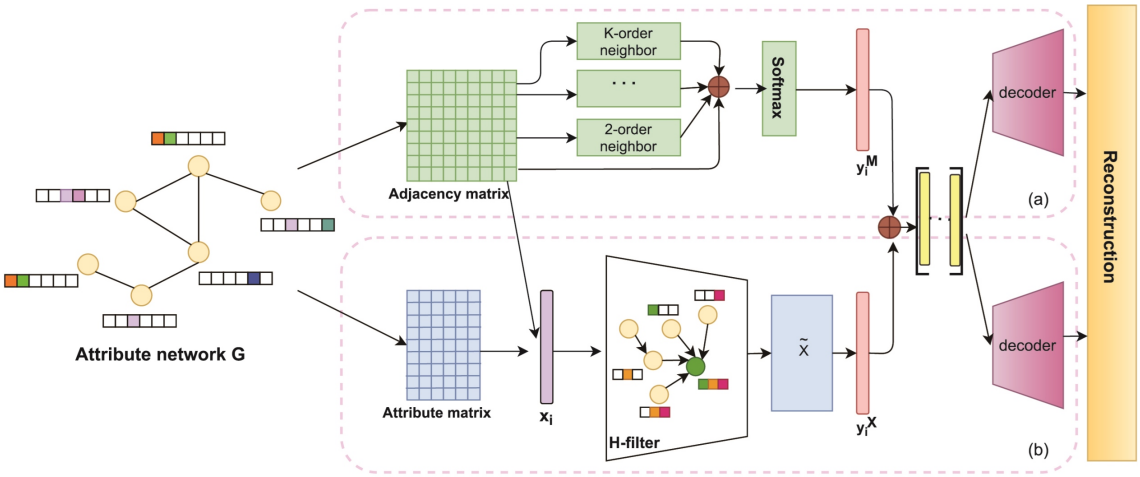

Figure 3. The DANRL framework. (**a**) represents the autoencoder of the structure, and the structure embedding representation vector is obtained. (**b**) represents the attribute autoencoder part, and the attribute embedding representation vector is obtained. The vectors learned by the dual autoencoders are adaptively fused, and the reconstructed matrix is output after reconstruction by the decoder.

3.2.1. Data Processing

The network structure is complex; some nodes do not have first-order proximity, but they have similar network structures, and the attribute information of nodes involves many types of data. Moreover, there are some nodes with missing and incomplete attribute information, etc. Therefore, this paper first preprocesses the node structure and attribute information.

For attribute networks, the adjacency matrix records the first-order proximity of the nodes in the network, while the first-order proximity can only reflect the local structure of the network, and some nodes may have similar neighborhood structures but do not have directly connected edges. For example, in social-network communities where people have common neighbors, they are not necessarily connected. In other words, the feature information obtained by first-order proximity alone is not enough. In this paper, the column vector of the adjacency matrix is used to encode the structural information of the nodes as the local structure of the network, and the second-order neighbor information of the network is encoded using a multi-hop attention-weighted summation.

Node attribute information usually involves many data types, and these data do not have size and order differences, and there is no direct connection between each attribute, so this paper encodes the attribute information uniquely and then splices the encoded representation of each attribute into the attribute vector representation of the node. For example, for any node v_i, its attribute representation vector is a_i, a_{ij} represents the attribute encoded vector corresponding to the node v_i, and $\oplus$ represents the splicing. Then:

$$a_i = a_{i1} \oplus a_{i2} \oplus a_{i3} \oplus \ldots \oplus a_{im} \tag{5}$$

For the problem of missing or incomplete attribute information, traditional methods include using statistics to fill in the missing data with the mean or plural or adding a random perturbation mechanism to add perturbations to the input samples with the probability to

randomly set the missing attribute information of some nodes to zero as the input vector. These methods are simple and intuitive, but they do not consider the structural information of the nodes, and there are biases in network information fusion. Therefore, in this paper, we combine the first-order proximity of the node structure and the node attribute matrix and fill the nodes with missing attribute information according to the first-order neighbor nodes of the target nodes.

3.2.2. Structural Autoencoder

To capture the highly nonlinear structural information, an unsupervised network representation learning module designed by the structural autoencoder is based on the reconstruction of the adjacency matrix task. Since it is necessary to capture both local and global highly nonlinear structural information of the network, this paper uses a graph attention mechanism to learn the importance weights between nodes and their neighbors to achieve the aggregation of weighted message-passing mechanisms.

An unsupervised network representation learning module designed by a structural autoencoder based on the task of reconstructing the adjacency matrix. Since it is necessary to capture both local and global highly nonlinear structural information of the network, this paper uses a multi-hop attention mechanism to learn the importance of weights among its neighbors to achieve the aggregation of weighted message-passing mechanisms. The geometric distances of nodes in the peripheral Euclidean space of the embedding space are also calculated, and the geometric distances are sorted to add the information of nodes that are geometrically close to each other to the aggregation operation.

First, the importance of learning neighboring nodes using the graph attention layer:

$$\mathrm{e_{ij}} = attn\left(\mathrm{y_i^M}, \mathrm{y_j^M}\right) = \sigma\left(\mu \cdot \left[W^{(1)}\mathrm{x_i^M} \oplus W^{(1)}\mathrm{x_j^M}\right]\right) \tag{6}$$

where $attn(\cdot)$ is the attention layer, μ and $W^{(1)}$ are the parameters to be learned, $\oplus$ denotes vector splicing, and $\mathrm{e_{ij}}$ denotes the importance of the features of node v_j to node v_i. To make the importance weight coefficients easily comparable across nodes, we normalize $\mathrm{e_{ij}}$ with the softmax function:

$$\gamma_{ij} = Soft\max_j\left(\mathrm{e_{ij}}\right) = \frac{\exp\left(\mathrm{e_{ij}}\right)}{\sum_{k\in N_i}\exp\left(\mathrm{e_{ik}}\right)} = \frac{\exp\left(\mathrm{ReLU}\left(\mu \cdot \left[W^{(1)}\mathrm{x_i^M}||W^{(1)}\mathrm{x_j^M}\right]\right)\right)}{\sum_{k\in N_i}\exp\left(\mathrm{ReLU}\left(\mu \cdot \left[W^{(1)}\mathrm{x_i^M}||W^{(1)}\mathrm{x_k^M}\right]\right)\right)} \tag{7}$$

In addition to capturing information about the proximity nodes of the target node, to also capture information about nodes that are not directly connected to an edge, we diffuse the graph using multi-hop attention, a process that computes the attention scores of multi-hop neighbors based on the matrix M:

$$\mathrm{M} = \sum_{i=0}^{k}\theta_i \mathrm{M}^i \text{ where } \sum_{i=0}^{k}\theta_i = 1 \text{ and } \theta_i > 0 \tag{8}$$

where θ_i is the attention weight decay factor and M^i describes the path length from one node to another and increases the acceptance domain of nodes, i.e., nodes that are geometrically close to each other in the peripheral Euclidean space. We define a differential twist function for node distance and node proximity "distance" in the peripheral Euclidean space as follows:

$$\rho = \frac{1}{\mathrm{n}(n-1)}\sum_{i\neq j}\frac{d_{\mathrm{M}}\left(\mathrm{x_i}, \mathrm{x_j}\right)}{d_{\varepsilon}\left(\mathrm{x_i}, \mathrm{x_j}\right)} = \frac{1}{\mathrm{n}(n-1)}\sum_{i\neq j}\frac{d_{\mathrm{M}}\left(\mathrm{x_i}, \mathrm{x_j}\right)}{\sqrt{\sum_{i=1}^{n}\left(\mathrm{x_i} - \mathrm{x_j}\right)^2}} \tag{9}$$

where $d_{\mathrm{M}}\left(\mathrm{x_i}, \mathrm{x_j}\right)$ represents the distance measure of proximity in non-Euclidean space, and $d_{\varepsilon}\left(\mathrm{x_i}, \mathrm{x_j}\right)$ represents the distance of nodes in peripheral Euclidean space. ρ represents

the "trade-off" between the two distances, and the smaller ρ means the "trade-off" between non-Euclidean and Euclidean is small, which better preserves the node proximity information and geometrically similar node information.

Finally, the neighboring features in the acceptance domain of the nodes are weighted and summed:

$$\mathrm{Y_i^M} = \sum_{k \in \mathrm{M}} \left(\gamma_{ik} \cdot \mathrm{Y_k^M} + \gamma_{i\varepsilon} \cdot \mathrm{Y_\varepsilon^M} \right) \tag{10}$$

after adaptive decoding, the structural embedding representation is obtained.

3.2.3. Attribute Autoencoder

The attribute autoencoder is a network representation learning module that captures highly nonlinear attribute information of nodes. In the attribute learning process, the encoder performs feature mapping of the original node attributes of the network and uses a Laplace smoothing filter to mitigate the high-frequency noise in the node attributes and obtains the embedded representation of the node attributes. The structure representation and attribute representation are adaptively fused to achieve interactive learning and consistency between them, and the reconstruction of the node attribute matrix is completed.

To measure the smoothness of the attribute vector x in the graph, firstly, calculate the Rayleigh entropy of the graph Laplacian matrix $\mathrm{L}(\mathrm{L} = \mathrm{D} - \mathrm{M})$ and the attribute vector x:

$$R(\mathrm{L}, \mathrm{x}) = \frac{\mathrm{x^T L x}}{\mathrm{x^T x}} \tag{11}$$

and

$$\begin{aligned} &\mathrm{x^T L x} = \mathrm{x^T D x} - \mathrm{x^T M x} \\ &= \sum_i \mathrm{x}^2(v_i) d_i - \sum_i \sum_j \mathrm{M_{ij} x}(v_i)\mathrm{x}(v_i) \\ &= \frac{1}{2}\left(\sum_i \mathrm{x}^2(v_i) d_i - 2 \sum_i \sum_j \mathrm{M}_{ij} \mathrm{x}(v_i)\mathrm{x}(v_j) + \sum_j \mathrm{x}^2(v_j) d_j \right) \\ &= \frac{1}{2} \sum_i \sum_j \mathrm{M_{ij}} \left(\mathrm{x_i} - \mathrm{x_j} \right)^2 \end{aligned} \tag{12}$$

It follows that neighboring nodes should have similar values, and the more similar they are, the smoother they are. The result of Rayleigh entropy is the eigenvalue L. The solution to x in $R(\mathrm{L}, \mathrm{x})$ corresponds to the eigenvector of L.

The conventional Laplace smoothing filter is defined as:

$$\mathrm{H} = \mathrm{I} - k\mathrm{L} \tag{13}$$

The filtered attribute vector $\widetilde{\mathrm{x}}$ is:

$$\widetilde{\mathrm{x}} = \mathrm{Hx} = (\mathrm{I} - k\mathrm{L})\mathrm{x} = \sum_{i=1}^{n} (1 - k\lambda_i) p_i \mu_i = \sum_{i=1}^{n} p_i^{'} \mu_i^{'} \tag{14}$$

where μ_i is the eigenvector of L and p_i is the coefficient of the eigenvector. The attribute vector matrix after t-layer Laplacian filtering is as follows: $\widetilde{\mathrm{X}} = \mathrm{H}^t \mathrm{X}$.

In the actual network analysis task, a symmetric normalized graph Laplace matrix is used, where $\widetilde{\mathrm{D}}$ and $\widetilde{\mathrm{L}}$ are the degree matrix and the Laplace matrix concerning the matrix $\widetilde{\mathrm{M}}$.

$$\widetilde{\mathrm{M}} = \mathrm{I} + \mathrm{M} \tag{15}$$

$$\widetilde{\mathrm{L}} = \widetilde{\mathrm{D}}^{-\frac{1}{2}} \mathrm{L} \widetilde{\mathrm{D}}^{-\frac{1}{2}} \tag{16}$$

Thus, the Laplace matrix is:

$$\mathrm{H} = \mathrm{I} - k\widetilde{\mathrm{L}} \tag{17}$$

For the choice of k values, let the $\widetilde{\mathrm{L}}$ maximum eigenvalues be λ_m, $k = 1/\lambda_m$. In the evaluation task and result analysis section, we show the effect of different k values on the experimental results.

In this paper, the similarity of attribute information between each pair of nodes in the attribute matrix after smoothing and filtering is calculated by cosine similarity, and then the similarity information between the nodes is stored as follows:

$$\mathrm{S}_{ij}^{\mathrm{X}} = \mathrm{Cos}Sim\left(\widetilde{\mathrm{X}}\right) = \frac{\mathrm{x_i x_j^T}}{|\widetilde{\mathrm{x_i}}||\widetilde{\mathrm{x_j}}|} \tag{18}$$

To analyze the distribution of common attributes between different nodes, we first determine whether there are directly connected edges between node pairs based on the adjacency matrix obtained from the original network, and then we multiply the corresponding attribute encoding vectors in the attribute matrix to determine the common attributes between two nodes. This part of the encoder, like the structural encoder, consists of multilayer nonlinear functions.

3.3. Model Optimization

In this paper, the optimization objective function of the model is defined as the joint optimization of the reconstruction error of the structural autoencoder and the attribute autoencoder, and the optimization loss function is as follows:

$$L_{loss} = L_{str} + L_{attr} = \min\left(\sum_{1}^{n}\left\|\mathrm{y_i^M} - \mathrm{y_i}\right\|_2^2 + \sum_{1}^{m}\left\|\mathrm{y_i^X} - \mathrm{y_i}\right\|_2^2\right) \tag{19}$$

Based on the above interpretation of the model components, a description of the DANRL algorithm is obtained as shown in Algorithm 1.

Algorithm 1: The algorithm of DANRL.

Input: Attribute network $G = (V, E, A)$,adjacency matrix M,attribute matrix X, filter layers t;
Output: Node representation matrix Y;
1. Calculate neighbor node importance weight $\mathrm{e_{ij}}$ from (6);
2. Normalized from (7);
3. for n = 2, 3, . . . , n
4. Calculate Euclidean distance and twist ρ from (9);
5. end for;
6. Obtain node structure embedded representation $\mathrm{y_i^M}$;
7. Obtain Laplacian $\widetilde{L}$ from (16);
8. $k \leftarrow 1/\lambda_m$;
9. Get filter matrix H from (17);
10. Get the smoothed attribute matrix $\widetilde{\mathrm{X}}$ from (14);
11. Calculate node attribute similarity matrix $\mathrm{S_{ij}^X}$ from (18);
12. Obtain node attribute embedded representation $\mathrm{y_i^X}$;
13. for epoch = 1, 2, . . . , custom do
14. Update encoder parameters;
15. Calculate the joint optimization loss function;
16. end for;

4. Experiments and Results Analysis

In this paper, extensive experiments were conducted to verify the superiority of DANRL by experimenting on four real network datasets and comparing them with traditional network representation learning methods and methods that incorporate node attribute information.

Experimental environment: Intel(R)Core(TM)i7-7700 CPU @ 3.6 GHz 3.6GH, GeForce GTX 1060Ti; Python 3.7.3, PyTorch 1.3.1.

4.1. Experimental Dataset

We performed statistics on the four datasets involved in this paper, and the results are shown in Table 2. The datasets are described in detail as follows.

Citation networks: Citeseer, Pubmed, and Cora all belong to citation networks. In citation networks, network nodes represent papers, and connected edges represent citation relationships between papers. Node labels are the research topics of papers, i.e., classification results, and node attributes represent the attribute features of each paper, such as keywords, year of publication, and research keywords. The Citeseer dataset classifies papers into six categories: Agents, AI, DB, IR, ML, and HCI; The Pubmed dataset comes from 19,717 papers on diabetes in the Pubmed database, and this dataset classifies papers into three categories; the Cora dataset consists of papers related to machine learning, and in this dataset, papers are classified into seven categories.

The Wiki dataset is a network of nodes as web pages, the links between different nodes are hyperlinks in the web pages, and the textual information on the web pages is processed similarly to the textual information in other datasets to extract attributes.

Table 2. Statistics of datasets.

Dataset	Node	Edge	Attribute	Lable
Citeseer	3312	4714	3703	6
Pubmed	19,717	44,338	500	3
Cora	2708	5429	1433	7
Wiki	2405	17,981	4973	17

4.2. Comparison Algorithm and Parameter Setting

In this paper, the DANRL algorithm is compared with eight representative network representation learning methods, including three traditional algorithms (Deepwalk, Node2vec, LINE) and five algorithms that combine attribute information (TADW, DANE, AANE, GAE, VGAE). 10% of the dataset is taken as the test set, 10% as the validation set, and the remaining 80% as the training set, and the training samples are adaptively reselected every time the threshold is updated. The comparison algorithm is divided into two groups:

(1) Traditional structure-based network representation learning methods. These algorithms only consider the structural information of the network and do not combine network attribute information for node representation learning. Deepwalk and Node2vec algorithms, use random wandering to generate sequences of nodes and then input the sequences to the skip-gram model to learn the potential node representation vectors. LINE is a representation learning method that uses first-order similarity and second-order similarity to preserve the local and global structure information of the representation learning method.

(2) Attribute network representation learning methods. The difference between these methods and the traditional structure-based representation learning methods is that the attribute information of the nodes is also considered, and the similarity between the structure of the original network and the attribute information is maintained when learning the vector representation. Examples include the TADW method, which was the first to combine external information (textual information), the DANE method, which focuses on nonlinear information about structure and attributes, and the VGAE method, which uses the combined attribute similarity of a variational autoencoder to reconstruct the network structure.

In this paper, the node representation vectors learned by the two major classes of methods described above are used for the node clustering task, thus evaluating the effectiveness of the methods. In the algorithm that requires random wandering, the number of wandering nodes is set to 80, the step size to 10, and the window size to 10. The regularization term coefficient of TADW is 0.1, and the embedding dimension of all algorithms is set to 256. The parameters of the DANRL algorithm for different data sets are set in Table 3.

Table 3. Parameter settings of different datasets.

Dataset	t	lr
Cora	8	1×10^{-3}
Citeseer	3	3×10^{-3}
Pubmed	35	1×10^{-4}
Wiki	1	1×10^{-3}

4.3. Evaluation Tasks and Analysis of Results

In this paper, we measure the DANRL algorithm performance through a common downstream task of network analysis, the node clustering task. Node clustering is an unsupervised method in which nodes are grouped into clusters, and we calculate the accuracy of node clustering by comparing the obtained label results with the true labels, and normalized mutual information (NMI) is used to measure the similarity of the clustering results, which takes values in the range of [0, 1], with larger accuracy and NMI values indicating better clustering results. The node-clustering task is used to evaluate the effect of the results of node embedding on subsequent, e.g., network analysis tasks such as community detection, to obtain a comparison between the labeled results and the true labels to calculate the accuracy of node clustering. Specifically:

$$ACC = \frac{\sum_{i=1}^{n} \delta(s_i, map(r_i))}{n} \tag{20}$$

where r_i is the label after clustering, s_i is the true label of the data, n is the total number of data, and δ is the comparison indicator function.

$$\delta(x, y) = \begin{cases} 1 \cdots if\ x = y \\ 0 \cdots otherwise \end{cases} \tag{21}$$

The standardized mutual information (NMI) is used to measure the similarity of clustering results, and is one of the important indicators for community detection. Its value is in teh range of [0, 1], and a larger value indicates that the clustering results are more similar, as defined below:

$$NMI(X, Y) = -2\frac{\sum_{x}\sum_{y} p(x, y) \log \frac{p(x,y)}{p(x)p(y)}}{\sum_{i} p(x_i) \log p(x_i) + \sum_{j} p(x_j) \log p(x_j)} \tag{22}$$

where X denotes the true label of the data, Y denotes the label after the clustering algorithm, $NMI(X, Y)$ denotes the similarity between the computed result and the actual label; $p(x, y)$ denotes the joint probability distribution between X and Y, and $p(x)$, $p(y)$ denote the edge distribution.

The node clustering task accuracy and NMI experimental results are shown in Table 4, where the bolded values are the results of the algorithm DANRL in this paper, the underlined ones are the optimal results, and "−" indicates no experimental results. Because the NMI results of node2vec and LINE algorithms for clustering tasks on the Wiki dataset are poor, they are not presented, and the NMI results of the AANE algorithm on the Pubmed and Wiki datasets are unstable, so they are not compared. For the Cora dataset, the precision of the DANRL algorithm is 0.775, which is 7.3% better than the 0.702 of the suboptimal DANE algorithm. For the Wiki dataset, the precision and NMI of the optimal results are 0.473 and 0.499, respectively, which are slightly higher than the 0.468 and 0.497 of the DANRL algorithm, but the DANRL algorithm is close to the optimal results. For the Citeseer dataset, the DANRL accuracy and NMI are 0.705 and 0.458, which are significantly higher than the suboptimal results of 0.479 and 0.422 in other algorithms, with an improvement of nearly 23% and 3%. This proves that the algorithm in this paper has better performance on the clustering task.

Table 4. Experimental results of node clustering.

Method	Cora		Citeseer		Pubmed		Wiki	
	ACC	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**
Deepwalk	0.482	0.328	0.326	0.088	0.543	0.105	0.388	0.223
Node2vec	0.647	0.356	0.451	0.101	0.664	0.127	0.379	–
LINE	0.479	0.433	0.391	0.225	0.661	0.387	0.409	–
TADW	0.599	0.443	0.455	0.290	0.511	0.244	0.311	0.118
DANE	0.702	0.630	0.479	0.422	0.694	0.308	0.473	0.499
AANE	0.445	0.161	0.447	0.143	0.451	–	0.432	–
GAE	0.616	0.490	0.367	0.223	0.631	0.248	0.377	0.374
VGAE	0.554	0.407	0.377	0.281	0.627	0.333	0.444	0.299
DANRL	**0.775**	**0.695**	**0.705**	**0.458**	**0.709**	**0.326**	**0.468**	**0.497**

Figure 4 is a visualized line graph of the accuracy results of the DANRL algorithm and other algorithms for the node clustering task on different datasets. The vertical axis represents the clustering accuracy (ACC), the horizontal axis represents the different four datasets, and the dark gray dash indicates the algorithm in this paper. It can be found that on the Citeseer dataset, the clustering accuracy of the DANRL method is much higher compared to the other algorithms. The accuracy results are highest on the Cora dataset and not so high on the Wiki dataset, but compared to other representation learning algorithms, the results of this paper's method are highest on this dataset, indicating that network attribute information is of different importance for different network structures, and that is not the case that the more attributes are combined the better.

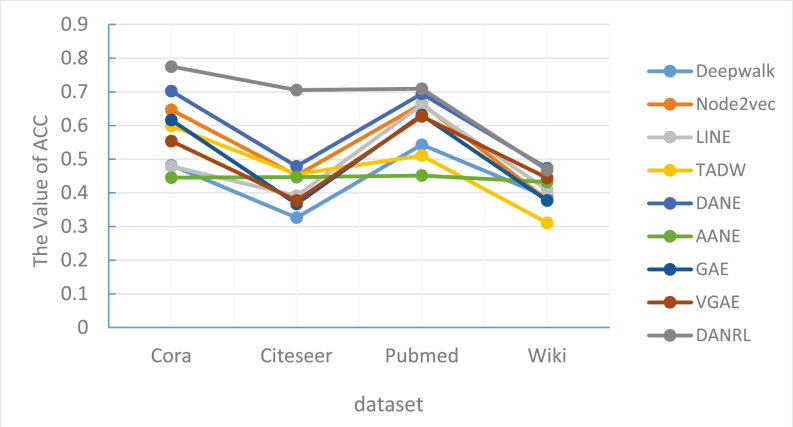

Figure 4. Node clustering comparison algorithm ACC values.

Figure 5 shows the visualization results of the node-clustering NMI evaluation metrics on all network data, where the dark red color represents the results of the DANRL algorithm proposed in this paper. The model architecture of the dual autoencoders proposed in this paper, where the structure and node attributes learn interactively and constrain each other, shows advantages in different networks, and although some of the network results are not optimal, they are close to the best results.

Figure 6 shows the experimental results for different k values. The left graph clusters ACC metrics and the right graph shows clustering NMI metrics. In the left panel, the vertical axis indicates the clustering accuracy, and the horizontal axis indicates different k values. The ACC metrics of Cora and Citeseer networks are optimal when $k = 2/3$, and the difference between the results on Pubmed and Wiki networks and $k = 4/5$ is not very obvious, but from $k = 1$ to $k = 2/3$ and $k = 4/5$, ACC is optimal at the place closest to the inverse of the maximum eigenvalue of each network (i.e., $k = 2/3$). In the right panel, the vertical axis is the clustered NMI values, the horizontal axis is the different k

values, and the four colors are the four data sets. At $k = 2/3$, most of the histograms are the highest, and only the light green and dark blue results do not differ much at different k values, proving that the low-pass filter designed in this paper is an improvement over the traditional convolution operation.

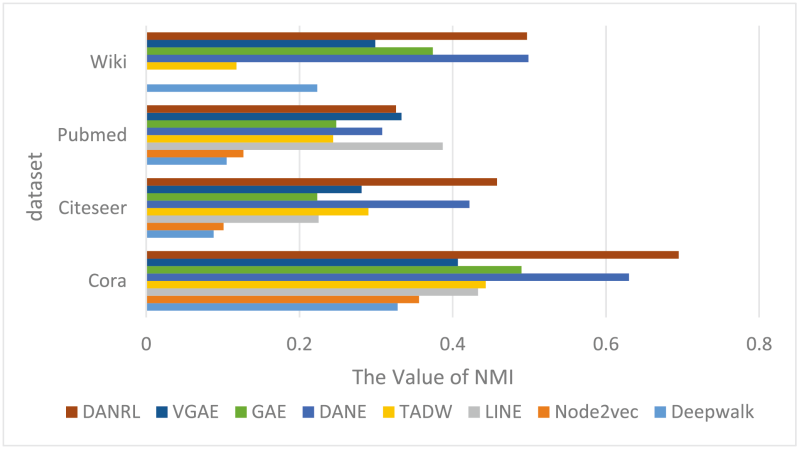

Figure 5. Node clustering comparison algorithm's NMI values.

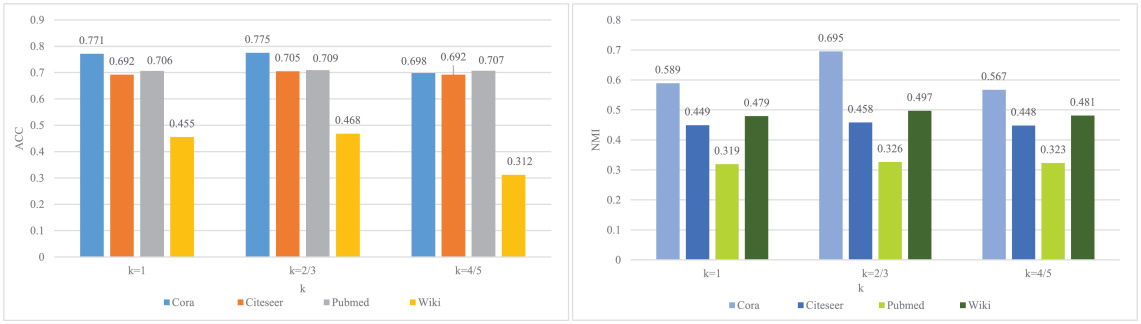

Figure 6. Experimental results for different k values.

Table 5 shows the average running time of the DANRL algorithm compared with other algorithms, where the bolded values are the results of the algorithm DANRL proposed in this paper. Deepwalk is chosen for the traditional representation learning method, and three methods are chosen for combining attribute information: TADW, GAE, and VGAE. The average running time of the DANRL algorithm for one epoch on the Cora dataset is 0.4602 s, compared to the running time of the TADW algorithm of 0.8546 s, the running time of the GAE algorithm of 0.5554 s, and the VGAE algorithm's running time of 0.5063 s, which is much shorter. For the relatively large dataset Pubmed, the running time of the DANRL algorithm is 17.4906 s, which is shorter than the running time of all four of the other methods. While the Wiki dataset is small and has a lot of attribute information, Deepwalk does not consider the node attributes, so the running time is the shortest at 1.4997 s. In summary, the results show that the efficiency of the methods in this paper is better.

Table 5. Comparison of average running time between DANRL and other algorithms.

	Cora	Citesser	Pubmed	Wiki
Deepwalk	0.6298	1.2638	32.7469	1.4997
TADW	0.8546	1.6376	30.5875	53.3486
GAE	0.5554	0.9978	22.9045	17.2567
VGAE	0.5063	0.9056	20.0006	16.4976
DANRL	**0.4602**	**0.8547**	**17.4906**	**18.6905**

4.4. Model Variants

To verify the effectiveness of the method proposed in this paper, the variant model was set up with the same parameter settings as shown in Table 3, and the experimental results are shown in Table 6, where the bolded values are the results of the method proposed in this paper. For the Citeseer dataset, the accuracy of the dual autoencoders is 0.705, and the accuracies of the variant models are 0.699 and 0.692, and the model in this paper is improved by nearly 1 percentage point. For the Cora and Pubmed datasets, the accuracy of only the structure encoder (i.e., M) is 0.666 and 0.506, respectively, and the results are much worse than the accuracy of the dual encoders, 0.775 and 0.709, but the accuracy of only the attribute encoder (i.e., X) is 0.753 and 0.673, and the results are better than those of the other variant model, indicating that the network structure and attribute information for different datasets have different levels of importance. Compared with the first two model variants, the accuracies are 0.775, 0.705, and 0.709 on the three data sets, which are all improved and prove the superiority of the model in this paper.

Table 6. Model variant.

Model	ACC		
	Cora	**Citeseer**	**Pubmed**
Structure-only (M)	0.666	0.699	0.506
Attribute-only (X)	0.753	0.692	0.673
Str + Attribute (M + X)	**0.775**	**0.705**	**0.709**

Figure 7 is a histogram of the clustering accuracy (ACC) of the variant model versus the model in this paper. The vertical axis is the accuracy value and the horizontal axis is the dataset. The blue color is the result of this paper's model, the light purple color is variant 1, i.e., only the structural encoder, and the gray color is variant 2, i.e., only the attribute encoder. From the bar height, we know that the results for the Citeseer dataset do not differ much, and on the Pubmed and Cora datasets, the model of this paper reflects a clear advantage. The reason is that the Citeseer dataset has a relatively small number of nodes, but a large number of attributes and edges, and the hidden geometric structure and attribute information appear for this problem.

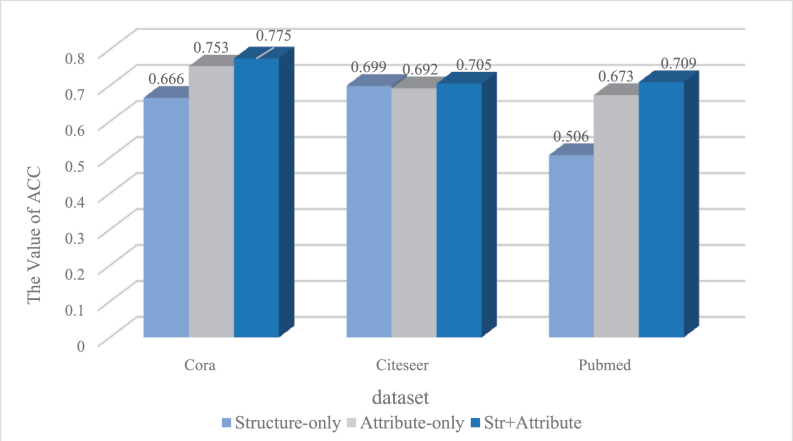

Figure 7. Model variant results.

5. Conclusions

This paper proposes a dual autoencoder learning method for attribute network representation. One channel autoencoder uses the multi-hop attention mechanism to capture the high-order neighborhood information of the node, calculate the importance weight of the node neighbors, and ensure the local and global structure of the network. The other

channel autoencoder adopts a low-pass filter to iteratively obtain the attribute information of the neighbors in the node neighborhood based on the network structure. The dual autoencoders ensure the learning of the structure and attribute respectively, and then the adaptive fusion of structure embedding and attribute embedding ensures the interactivity and mutual restriction between the two kinds of information, which makes up for the lack of interactivity of the structure and attribute information in the current network representation learning and the lack of joint learning node representation of the two kinds of information. Th experiments on real datasets demonstrated the superiority of the method in this paper for attribute network representation learning. In the future, we plan to investigate the problem of hidden geometric structures (e.g., hierarchical structures) and attribute information confrontation in attribute networks.

Author Contributions: Conceptualization, J.W.; methodology, J.W.; formal analysis, J.W.; writing—review and editing, J.W.; writing—original draft preparation, Z.Z.; software, Z.Z.; validation, Z.Z.; investigation, Z.Z.; resources, B.L.; data curation, M.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Hebei Natural Science Foundation (F2021205014), funded by Science and Technology Project of Hebei Education Department (ZD2022139), supported by the Natural Science Foundation of Hebei Province (F2019205303), supported by the Central Guidance on Local Science and Technology Development Fund of Hebei Province (226Z1808G), funded by The Introduction of Overseas Students in Hebei Province (C20200340), and supported by the Hebei Normal University Science and Technology Fund Project (L2019Z10).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zhou, J.Y.; Liu, L.; Wei, W.Q.; Fan, J.X. Network representation learning: From preprocessing, feature extraction to node embedding. *ACM Comput. Surv.* **2022**, *55*, 1–35. [CrossRef]
2. Amara, A.; Taieb, M.A.H.; Aouicha, M.B. Network representation learning systematic review: Ancestors and current development state. *Mach. Learn. Appl.* **2021**, *6*, 100130. [CrossRef]
3. Xu, M.J. Understanding graph embedding methods and their applications. *SIAM Rev.* **2021**, *63*, 825–853. [CrossRef]
4. Li, B.; Pi, D. Network representation learning: A systematic literature review. *Neural Comput. Appl.* **2020**, *32*, 16647–16679. [CrossRef]
5. Bai, L.; Yao, L.N.; Li, C.; Wang, X.Z.; Wang, C. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17804–17815.
6. Su, C.; Tong, J.; Zhu, Y.J.; Cui, P.; Wang, F. Network embedding in biomedical data science. *Briefings Bioinform.* **2020**, *21*, 182–197. [CrossRef]
7. Chen, L.; Xie, T.; Li, J.T.; Zheng, Z.B. Graph Enhanced Neural Interaction Model for recommendation. *Knowl. Based Syst.* **2022**, *246*, 108616. [CrossRef]
8. Wang, J.H.; Li, H.K.; Liang, L.N.; Zhou, Y. Community discovery algorithm of complex network attention model. *Int. J. Mach. Learn. Cybern.* **2022**, *13*, 1619–1631 . [CrossRef]
9. Wang, J.H.; Yang, J.T.; He, Y.C. Research on semi-supervised community discovery algorithm based on new annealing. *J. Eng.* **2020**, *12*, 1149–1154. [CrossRef]
10. Wang, J.H.; Yang, J.T.; Shi, S.L. Semi-Supervised Community Discovery Algorithm Based on Node Similarity. In Proceedings of the 2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering, Dalian, China, 14–16 November 2019. [CrossRef]
11. Chai, B.F.; Wang, J.H.; Yu, J. A parameter selection method of the deterministic anti-annealing algorithm for network exploring. *Neurocomputing* **2017**, *226*, 192–199. [CrossRef]
12. Sun, H.L.; He, F.; Huang, J.B.; Sun, Y.Z.; Li, Y.; Wang,C.Y.; He, L.; Sun, Z.; Jia, X. Network embedding for community detection in attributed networks. *ACM Trans. Knowl. Discov. Data* **2020**, *14*, 1–25. [CrossRef]
13. Xu, W.H.; Y, K.H.; Li, W.T.; Ding, W.P. An emerging fuzzy feature selection method using composite entropy-based uncertainty measure and data distribution. *IEEE Trans. Emerg. Top. Comput. Intel.* **2022** , 1–13. [CrossRef]

14. Yuan, K.H.; Xu, W.H.; Li, W.T.; Ding, W.P. An incremental learning mechanism for object classificationbased on progressive fuzzy three-way concept. *Inf. Sci.* **2022**, *584*, 127–147. [CrossRef]
15. Pan, G.S.; Yao, Y.; Tong, H.H.; Xu, F.; Lu, J. Unsupervised Attributed Network Embedding via Cross Fusion. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Virtual, 8–12 March 2021; pp. 797–805. [CrossRef]
16. Bandyopadhyay, S.; Biswas, A.; Kara, H.; Murty, M.N. A multilayered informative random walk for attributed social network embedding. *Eur. Conf. Artif. Intell.* **2020**, *325*, 1738–1745. [CrossRef]
17. Tong, N.; Tang, Y.; Chen, B.; Xiong, L.R. Representation learning using attention network and cnn for heterogeneous networks. *Expert Syst. Appl.* **2021**, *185*, 115628. [CrossRef]
18. Chen, X.W.; Xu, W.H. Double-quantitative multigranulation rough fuzzy set based on logical operations in multi-source decision systems. *Int. J. Mach. Learn. Cybern.* **2022**, *13*, 1021–1048. [CrossRef]
19. Xu, W.H.; Guo,Y.T. Generalized multigranulation double-quantitative decision-theoretic rough set. *Knowl. Based Syst.* **2016**, *105*, 190–205. [CrossRef]
20. Xu, W.H.; Li, W.T. Granular computing approach to two-way learning based on formal concept analysis in fuzzy datasets. *IEEE Trans. Cybern.* **2016**, *46*, 366–379. [CrossRef]
21. Liao, L.Z.; He, X.N.; Zhang, H.W.; Chua, T. Attributed social network embedding. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 2257–2270. [CrossRef]
22. Chen, J.; Zhong, M.; Li, J.X.; Wang, D.H.; Qian, T.Y; Tu, H. Effective Deep Attributed Network Representation Learning with Topology Adapted Smoothing. *IEEE Trans. Cybern.* **2021**, *52*, 5935–5946. [CrossRef]
23. Wang, X.; Zhu, M.Q.; Bo, D.Y.; Cui, P.; Shi, C.; Pei, J. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks.In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 6–10 July 2020; pp. 1243–1253. [CrossRef]
24. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 23–27 August 2014; pp. 701–710. [CrossRef]
25. Grover, A.; Leskovec, J. Node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864. [CrossRef]
26. Zhang, D.K.; Yin, J.; Zhu, X.Q.; Zhang, C.Q. Network representation learning: A survey. *IEEE Trans. Big Data* **2018**, *6*, 3–28. [CrossRef]
27. Tang, J.; Qu, M.; Wang, M.Z.; Zhang, M.; Yan, J.; Mei, Q.Z. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1067–1077. [CrossRef]
28. Wang, D.; Cui, P.; Zhu, W. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234. [CrossRef]
29. Li, Z.; Wang, X.; Li, J.X; Zhang, Q.P. Deep attributed network representation learning of complex coupling and interaction. *Knowl. Based Syst.* **2021**, *212*, 106618. [CrossRef]
30. Cui, P.; Wang, X.; Pei, J.; Zhu, W.W. A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 833–852. [CrossRef]
31. Zhang, Z.; Yang, H.X.; Bu, J.J.; Zhou, S.; Yu, P.G.; Zhang, J.W.; Ester, M.; Wang, C. ANRL: Attributed Network Representation Learning via Deep Neural Networks. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; Volume 18, pp. 3155–3161. [CrossRef]
32. Zhao, Z.Y.; Zhou, H.; Li, C.; Tang, J.; Zeng, Q.T. DeepEmLAN: Deep embedding learning for attributed networks. *Inf. Sci.* **2021**, *543*, 382–397. [CrossRef]
33. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.
34. Hamilton, W.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1025–1035.
35. Huang, X.; Song, Q.Q.; Li, Y.N.; Hu, X. Graph recurrent networks with attributed random walks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 732–740. [CrossRef]
36. Park, C.; Kim, D.; Han, J.W.; Yu, H. Unsupervised attributed multiplex network embedding. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 5371–5378. [CrossRef]
37. Wu, Z.H.; Pan, S.R.; Chen, F.W.; Long, G.D.; Zhang, C.Q.; Yu, P.S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
38. Wang, Y.; Sun, Y.B.; Liu, Z.W.; Sarma, S.E.; Bronstein, M.M.; Solomom, J.M. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.* **2019**, *38*, 1–12. [CrossRef]
39. Pan, Y.; Hu, G.; Qiu, J.; Zhang, Y.; Wang, S.; Shao, D.; Pan, Z. FLGAI: A unified network embedding framework integrating multi-scale network structures and node attribute information. *Appl. Intell.* **2020**, *50*, 3976–3989. [CrossRef]
40. Zhang, X.Y.; Li, J.R.; Mi, J.S. Dynamic updating approximations approach to multi-granulation interval-valued hesitant fuzzy information systems with time-evolving attributes. *Knowl. Based Syst.* **2022**, *238*, 107809. [CrossRef]
41. Xu, W.H.; Yuan, K.H.; Li, W.T. Dynamic updating approximations of local generalized multigranulation neighborhood rough set. *Appl. Intell.* **2022**, *52*, 9148–9173. [CrossRef]

42. Xu, W.H.; Yu, J.H. A novel approach to information fusion in multi-source datasets: A granular computing viewpoint. *Inf. Sci.* **2017**, *378*, 410–423. [CrossRef]
43. Wang, J.; Zhang, D.; Liang, L. A Classification Model with Cognitive Reasoning Ability. *Symmetry* **2022**, *14*, 1034. [CrossRef]
44. Wang, Y.S.; Yao, H.X.; Zhao, S.C. Auto-encoder based dimensionality reduction. *Neurocomputing* **2016**, *184*, 232–242. [CrossRef]