

Article

Network Embedding Algorithm Taking in Variational Graph AutoEncoder

Dongming Chen , Mingshuo Nie , Hupo Zhang, Zhen Wang and Dongqi Wang *

Software College, Northeastern University, Shenyang 110169, China; chendm@mail.neu.edu.cn (D.C.); niemingshuo@stumail.neu.edu.cn (M.N.); hupo Zhang96@163.com (H.Z.); 2071322@stu.neu.edu.cn (Z.W.)
* Correspondence: wangdq@swc.neu.edu.cn

Abstract: Complex networks with node attribute information are employed to represent complex relationships between objects. Research of attributed network embedding fuses the topology and the node attribute information of the attributed network in the common latent representation space, to encode the high-dimensional sparse network information to the low-dimensional dense vector representation, effectively improving the performance of the network analysis tasks. The current research on attributed network embedding is presently facing problems of high-dimensional sparsity of attribute eigenmatrix and underutilization of attribute information. In this paper, we propose a network embedding algorithm taking in a variational graph autoencoder (NEAT-VGA). This algorithm first pre-processes the attribute features, i.e., the attribute feature learning of the network nodes. Then, the feature learning matrix and the adjacency matrix of the network are fed into the variational graph autoencoder algorithm to obtain the Gaussian distribution of the potential vectors, which more easily generate high-quality node embedding representation vectors. Then, the embedding of the nodes obtained by sampling this Gaussian distribution is reconstructed with structural and attribute losses. The loss function is minimized by iterative training until the low-dimension vector representation, containing network structure information and attribute information of nodes, can be better obtained, and the performance of the algorithm is evaluated by link prediction experimental results.

Keywords: attributed network; network embedding; random walk; autoencoder



Citation: Chen, D.; Nie, M.; Zhang, H.; Wang, Z.; Wang, D. Network Embedding Algorithm Taking in Variational Graph AutoEncoder. *Mathematics* **2022**, *10*, 485. <https://doi.org/10.3390/math10030485>

Academic Editors: András Benczúr, Domenico Ursino and Bálint Molnár

Received: 11 January 2022

Accepted: 1 February 2022

Published: 2 February 2022

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



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Attribute networks are widely employed to model connections between entities in the real world, where the connected edges of nodes denote the relationships between objects and the attribute information of the nodes in the description about the nodes themselves. For example, the citation network [1,2] describes citations between articles by academic authors, and the node attribute information describes the abstract and subject of the article. Social networks [3,4] describe interpersonal relationships between people, the node also contains a user's area, email, and release of words, pictures, and video, etc. Protein interaction network [5]: In addition to the topological information about protein interactions, each node also features attribute information about proteins in biology. Performing data analysis and data mining on these attribute networks, using network information for prediction and decision making, and discovering useful information latent in them, have strong academic and commercial value. Common network analysis tasks include node classification [6], node clustering [7], community detection [8–10], link prediction [11,12], and anomaly detection [13]. The networks are growing in size in the big data era, some reaching hundreds of millions of nodes, with high and sparse dimensions that not only have complex structures but are also rich in information. If such large-scale networks are directly applied to data mining, not only will they consume a large amount of data storage resources, but also the time complexity of computation is high, which is not conducive to algorithm deployment and application.

Attribute network embedding [14] enables the topological structure information of an attribute network and node attribute information to be effectively fused in a common latent representation space, which preserves the information of both. The schematic diagram of attribute network embedding is shown in Figure 1. The accuracy of the downstream network analysis task is further improved by enhancing the network node vector representation by employing information about the attributes of the nodes, based on the previous consideration of only the network structure information.

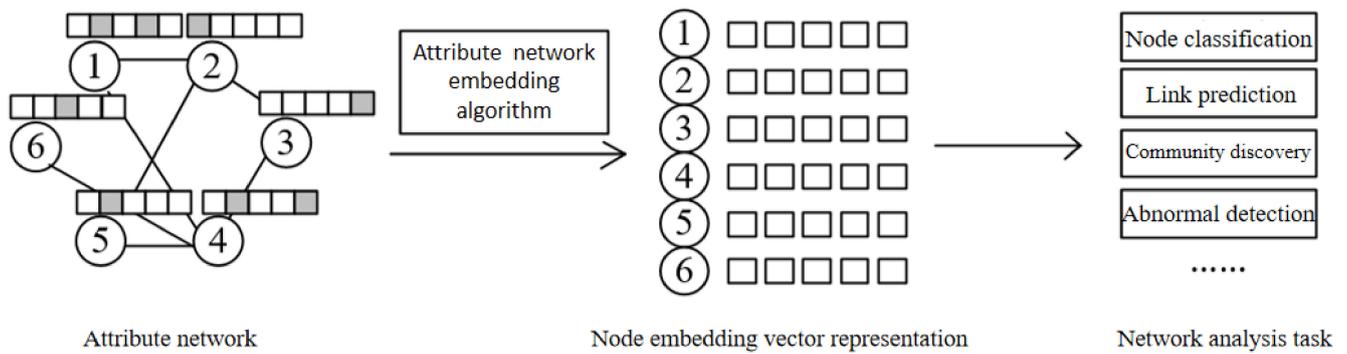


Figure 1. Diagram of attributed network embedding.

Graph neural networks (GNNs) [15] are also employed to perform network embedding. The graph convolutional network (GCN) [16] and the GraphSAGE model [17] employ graph convolutional networks, sampling nodes, and local neighbors for feature aggregation to learn the vector representation of nodes. The graph autoencoders (GAEs) [18] algorithm is an attribute network embedding algorithm that introduces a deep autoencoder. Based on the idea of the autoencoder, after using GCN encoding, the structural error is reconstructed in the decoder to learn the vector representation of the node. Variational graph autoencoders (VGAEs) [18] build on GAEs by using graph variational autoencoders to learn the embedding representation of attribute networks. The co-embedding model for static attributed networks (CSAN) [19] is also an attribute network embedding algorithm based on graph autoencoder, but in the attribute reconstruction part, the matrix product of the GCN encoder of the graph convolutional network and the multilayer perceptron MLP processing is employed to reconstruct the attribute errors.

The GAE algorithm is the attribute feature matrix employed as the GCN encoder input when extracting the attribute features of the network. The node attribute feature matrix is generally obtained by processing the descriptive information of the nodes using a bag-of-words model or a TF-IDF model. If the word list is too large, it tends to be characterized by high-dimensional sparsity and great algorithmic complexity. Moreover, these feature matrices are directly employed as encoder input, which does not reflect or fully employ the node attribute information. In addition, the GAE algorithm takes a single mapping approach to generate node embedding vectors in the encoder part, while the variogram autoencoder can generate the distribution of latent vectors, which is more capable of generating high-quality potential vectors after sampling, and the GAE algorithm does not consider the reconstruction loss of attribute information.

To address the above problems, this paper proposes a network embedding algorithm taking in a variational graph autoencoder (NEAT-VGA). The main idea of the NEAT-VGA algorithm is to first pre-process the attribute features, and the obtained attribute feature learning vectors are inputted into the variational graph autoencoder, together with the adjacency matrix, to complete the attribute network embedding task. Initially, the MHRWAE algorithm is employed to learn the attribute features of the nodes for the attribute network to obtain a low-dimensional vector representation of the node attributes. Then, the feature learning matrix and the adjacency matrix of the network are employed as input to the variogram autoencoder algorithm to generate a Gaussian distribution of the latent vectors, sample the Gaussian distribution to obtain the embedding vectors of

the nodes, reconstruct the structural and attribute losses, and iteratively train for attribute network embedding. The final experimental results show that the proposed algorithm in this paper has better link prediction results compared with the benchmark comparison algorithm, i.e., the proposed network representation learning algorithm in this paper has better representation performance.

2. Related Works

The DeepWalk [20] algorithm is inspired by the Word2Vec [21], which treats the sequence of nodes obtained by random walking in the network as sentences in the text and the nodes in the sequence as words in the text, to obtain the embedding representation of the nodes by maximizing the probability of predicting the node's context by employing the skip-gram model. The LINE [22] algorithm learns the embedding representation of a node to design and optimize the objective function with the definition of first-order and second-order similarities of the nodes. The first-order similarity refers to the existence of contiguous edges between nodes, while the second-order similarity refers to the common neighbor nodes between nodes. With the introduction of second-order similarity, LINE greatly alleviates the sparsity problem in the network, and the obtained node embedding representation can also maintain the local and global structural information of the network to the greatest extent. Based on DeepWalk, Node2Vec [23] introduces parameters to control the path of random walk, allowing the choice of whether to bias the random walk towards depth or breadth, and the sequence of nodes generated by the random walk, thus enabling better preservation of local and global structural information of the network.

With the development of deep learning in the fields of natural language processing and computer vision, many algorithms have also emerged that employ deep learning for network embedding, which employs neural networks to extract the structure of attribute networks and attribute information to combine to obtain the embedding representation of network nodes. The self-translation network embedding (STNE) [24] algorithm employs a sequence model to map a sequence of nodes' content information to a sequence of nodes for network embedding. The attributed social network embedding (ASNE) [25] algorithm combines the information about the attributes of nodes in social networks during the learning node embedding. The attributed network representation learning (ANRL) [26] algorithm is an algorithm that employs a neighborhood-enhanced autoencoder and an attribute-aware skip-gram model, employed to capture node attribute and network structure information for attribute network embedding algorithm, which employs the node's attribute information as input to the autoencoder to reconstruct the attribute information of neighboring nodes.

3. Methodology

The NEAT-VGA algorithm consists of the 5 following components:

- (1) **Node Attribute Feature Learning:** We employ the Metropolis–Hastings random walk (MHRW) algorithm [27] sampling, node sequence generation corpus, and Doc2Vec [28] model training to preprocess the attribute information, which is the attribute feature learning, named the MHRWAE algorithm. High-dimensional sparse attribute feature matrix learning can obtain low-dimensionality and better reflect the attribute feature learning matrix.
- (2) **Attribute Network Encoder:** The adjacency matrix and the attribute feature learning matrix are employed as input to the variational graph autoencoder, which takes GCN for encoding and maps the attribute network to a Gaussian distribution.
- (3) **Structure Reconstruction Decoder:** Sampling the Gaussian distribution to obtain a vector representation of the nodes, the structure decoder is employed to reconstruct the adjacency matrix of the network, i.e., the structural information of the network.
- (4) **Attribute Reconstruction Decoder:** Sampling the Gaussian distribution to obtain a vector representation of the nodes and using the attribute decoder to reconstruct the attribute feature learning matrix of the encoder input.

- (5) **Loss Function Definition:** We define a novel loss function, and the matrix reconstructed by the decoder is compared with the input information of the variogram autoencoder to construct the loss function, considering the effects of structural and attribute reconstruction. The final learning yields vector representations that are as good as possible in low-dimensional space for both structure and attributes.

The framework of the NEAT-VGA algorithm is shown in Figure 2.

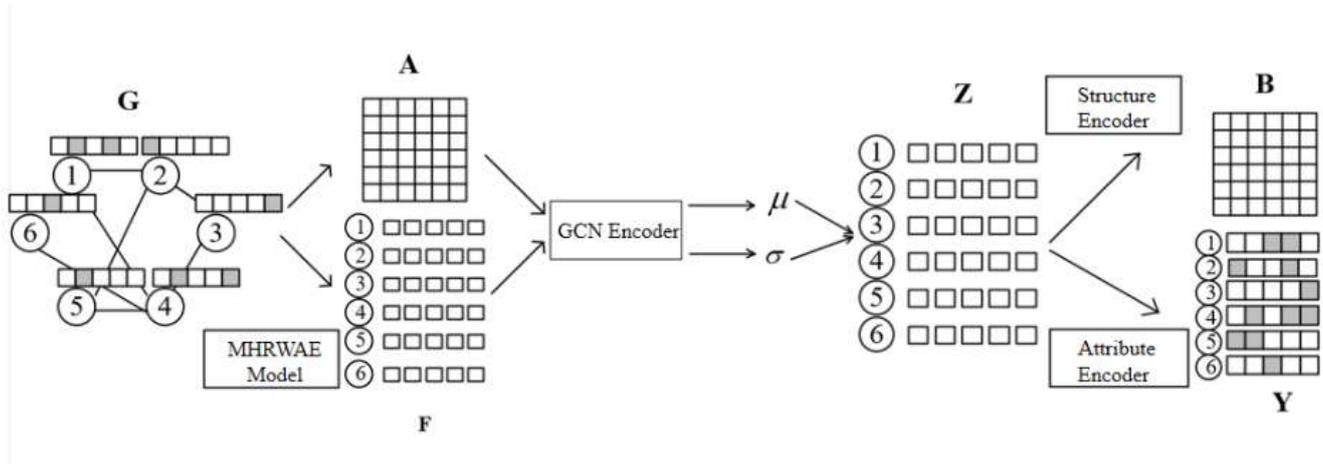


Figure 2. The framework of the NEAT-VGA algorithm.

3.1. Preliminaries

The goal of the NEAT-VGA algorithm is to learn the feature vector of each node based on the attribute network graph. Symbols and descriptions are shown in Table 1.

Table 1. Description of mathematical symbols.

Symbol	Description
G	Attribute network diagram
V, E	Node set, edge set
$ V , E $	Number of nodes, number of edges
A	The adjacency matrix of the topological structure of the attribute network graph
d	Attribute vector dimension of each node
$X \in R^{ V \times d}$	Attribute matrix
m	Attribute feature learning vector dimension
$F \in R^{ V \times m}$	Node attribute feature learning matrix
g	The number of sample node sequences
l	The length of each sample node sequence
t	Neighborhood size when node attributes are aggregated
n	The vector dimension represented by each node embedding
μ	Node embedding represents the mean of Gaussian distribution
σ	Node embedding represents the variance of Gaussian distribution
p	Training times of the node attribute feature learning part
q	The number of training times of the image autoencoder attribute network embedding
$Z \in R^{ V \times n}$	Attribute network embedding representation matrix

3.2. Aggregation-MHRWAE

The random walk-based network embedding algorithms sample node sequences with random walks, making the sampled sequences biased toward nodes of a large degree, and such algorithms do not consider the attribute information of nodes during the training process. The stitching relationship between structure vectors and attribute vectors is too complicated to fuse the structure and attributes well by these methods directly. In this paper,

we propose the MHRWAE algorithm, which is a module of the NEAT-VGA algorithm. Its main function is to preprocess the attribute features of the nodes of the attribute network.

The main idea of the MHRWAE algorithm is that the embeddings of some nodes in the network should be more similar if they have similar attributes and the distribution of attributes of neighborhood nodes similarly. The framework of this algorithm is shown in Figure 3. The MHRWAE algorithm employs MHRW to generate a fixed-length sequence of nodes and the structural information of the network is utilized to generate a corpus based on the aggregation of node neighborhood relationships and node attribute information. The corpus is then trained using skip-gram with negative sampling to fuse the network structure and attribute information of the nodes to obtain a low-dimensional representation vector for each node.

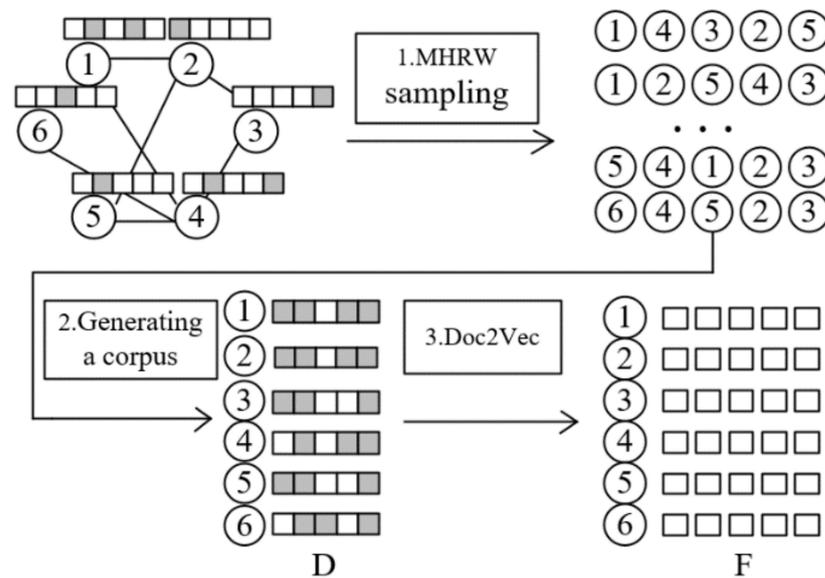


Figure 3. The framework of the MHRWAE algorithm.

- (1) The unbiased random wandering is performed with the MHRW algorithm to generate a sequence of nodes. the MHRW algorithm samples nodes without bias towards nodes of a larger degree, and the node sequences generated by the sampling reflect the connectivity between nodes, i.e., the structural information of the network. The MHRW algorithm sampling process is where the transfer probability of the MH algorithm is employed in the Random Walk algorithm to determine the transfer probability when the current node is sampled to a neighboring node. We assume that the probability distribution given in the MH algorithm is $\pi_v = 1/|V|$, and the election probability, $Q_{i,j}$, is assumed to be $Q_{i,j} = 1/d_i$, then the transition probability of sampling from node i to its neighbor node j , as shown in Equation (1), as follows:

$$p_{i,j} = \frac{1}{d_i} \times \min(1, \frac{d_i}{d_j}), i \neq j, \tag{1}$$

where $p_{i,i} = 1 - p_{i,j}$, d_i and d_j respectively, represent the degrees of nodes i and j . $p_{i,i}$ indicates the probability of node i staying at the current node.

- (2) Corpus generation: In the process of corpus generation by the MHRWAE algorithm, unlike DeepWalk, which employs multiple sets of nodes and node context nodes as a corpus, it employs multiple sets of nodes and node neighborhood node attributes as the corpus for SGNS training. The corpus is generated employing neighborhood node attribute aggregation, in which a sequence of nodes of a given length is generated, and for each node in the sequence, the attributes of the node and its neighborhood nodes are paired and added to the multiset, with each node in the sequence completing an iterative node attribute aggregation operation to form the final corpus.

- (3) The Doc2Vec model trains the corpus: Doc2Vec is a model that generates vector representations of documents, and the PV-DBOW method in the model enables SGNS. using the corpus as input, the Doc2Vec model is employed to train the corpus and generate a vector representation of each document, i.e., a vector representation of each node is obtained.

3.3. NEAT-VGA

The training process of the NEAT-VGA algorithm is shown in Algorithm 1.

Algorithm 1. NEAT-VGA

Input: Attribute network $G = (V, E, X)$, parameter $\alpha, m, g, l, b, t, n, p, q, a_1, a_2$.

Output: Attribute network node embedding vector Z

```

1:  for  $n = 1: p$  do
2:      Pick  $v_i$ .
3:      Generate a vertex sequence  $(v_i, v_i, \dots, v_l)$  of length  $l$  by MHRW on network  $G$ .
4:      for  $j$  in  $1: l-t$  do
5:          for  $r$  in  $1: t$  do
6:              Add vertex–context–feature pair  $(v_j, f_{j+r})$  to multiset  $D$ .
7:              Add vertex–context–feature pair  $(v_{j+r}, f_j)$  to multiset  $D$ .
8:              Update  $D$ .
9:          end for
10:     end for
11: end for
12: Doc2Vec ( $D, epochs = p, m, b, a_1$ ).
13: Create  $F \in R^{|V| \times m}$ .
14: Calculate matrix  $D \sim^{-\frac{1}{2}} A D \sim^{-\frac{1}{2}}$ .
15: Initialization parameter  $\theta = [W^{(0)}, W^{(1)}, W^{(2)}, W^{(3)}, W^{(4)}]$ .
16: for  $j = 1: q$  do
17:     Calculate  $\mu$  and  $\sigma$  according to Equations (5) and (6).
18:     Calculate  $Z$  according to Equation (7)
19:     Calculate  $\mathcal{L}_s \mathcal{L}_a \mathcal{L}_{KL}$  according to Equations (9)–(12).
20:     Calculate  $\mathcal{L}$  according to Equation (13)
21:     Update  $\theta$  by Adam algorithm and learning rate =  $a_2$ .
22: end for
23: return  $Z \in R^{|V| \times n}$ .

```

3.3.1. Node Attribute Feature Learning

Pre-processing of node attribute information uses the MHRWAE algorithm, which consists of the MHRW algorithm sampling, node sequence generation corpus, and Doc2Vec model training. The nodes include the attributes of their neighboring nodes in their attributes; as a result, each node’s attributes are a sentence of attributes, and the words of the sentence are the node’s attributes and the neighboring node’s attributes.

For each node, there is a corresponding sentence describing the attributes of the node, and the addition of the neighborhood attribute makes the node with the more similar structure, the sentence with the more similar node attribute. Finally, the node numbers and attribute sentences are trained on the Doc2Vec model to obtain the attribute feature learning vector for each node.

For the attribute network $G = (V, E, X)$, according to the MHRWAE algorithm, given the vector dimension, m , of the attribute embedding representation of each node, the number of sampled node sequences, g , the length of each sampled node sequence, l , the neighborhood size, t , when node attributes are aggregated, and negative sampling parameters, such as the number, b , the learning rate, a_1 , and the number of training, p , can be obtained: $F \in R^{|V| \times m}$.

3.3.2. Attribute Network Encoder

This algorithm is implemented by introducing a variational self-encoder into the network structure using a GCN, which is used to capture information about the underlying attribute network. The GCN encoder considers the proximity of high-order nodes, solves the problem of network sparsity, and captures the non-linear information of the data through multi-layer, non-linear conversion. The graph convolutional network of each layer can employ Equation (2) to express the specific convolution process, as follows:

$$H^{(l+1)} = f(H^{(l)}, A|W^{(l)}) = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}), \tag{2}$$

where $H^{(l)}$ is the input of the l layer convolutional layer, $H^{(l+1)}$ is the output of the l layer convolutional layer $\tilde{A} = A + I$, which means that the adjacency matrix of the network and the identity matrix are added together. \tilde{D} is a diagonal matrix, and the value of \tilde{A} the diagonal is related. The specific calculation equation is $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$. $\sigma(\cdot)$ is a non-linear activation function, for example, $Relu(x) = \max(0, x)$.

In the encoding stage, the NEAT-VGA algorithm obtains the Gaussian distribution of each node and samples the Gaussian distribution to obtain the attribute network embedding representation matrix, $Z \in R^{|V| \times n}$. The Gaussian distribution can be uniquely determined by the mean, μ , and variance, δ , and the encoder part is the process of obtaining μ and δ through the GCN encoder.

The NEAT-VGA algorithm first employs two layers of GCN to share weights $W^{(0)}$. $W^{(1)}$ after extracting features, one layer of GCN is added to obtain μ , $\log \sigma$, and weight $W^{(2)}$, $W^{(3)}$. Therefore, the generation of each μ and σ is obtained by three-layered GCN encoding, but in the first two, the network encoder that shares parameters when layering is employed as an attribute network encoder.

The activation function of each layer of graph convolutional network is $Relu(x) = \max(0, x)$, the specific attribute network encoder process is shown in Equations (3)–(6), as follows:

$$H^{(1)} = f_{relu}(F, A|W^{(0)}), \tag{3}$$

$$H^{(2)} = f_{relu}(H^{(1)}, A|W^{(1)}), \tag{4}$$

$$\mu = H^{(3)} = f_{relu}(H^{(2)}, A|W^{(2)}), \tag{5}$$

$$\log \sigma = H^{(4)} = f_{relu}(H^{(2)}, A|W^{(3)}), \tag{6}$$

After μ and σ are obtained, which is after the mean vector covariance matrix of the Gaussian distribution of each node, further sampling can be performed to calculate the attribute network embedding representation matrix, $Z \in R^{|V| \times n}$. The specific equation for obtaining Z through the mean and variance is shown in Equation (7), as follows:

$$Z = \mu + \varepsilon\sigma, \tag{7}$$

where ε obeying the standard $N(0, 1)$, Z is obeying $N(\mu, \sigma^2)$.

Therefore, first, randomly sample one ε from the standard distribution, and then calculate Z through the equation; then, you can further update the gradient through backpropagation, and iteratively train to obtain new μ and σ .

3.3.3. Structure Reconstruction Decoder

In the attribute network encoding part, the attribute network embedding representation matrix is obtained, $Z \in R^{|V| \times n}$. The reconstructed adjacency matrix, B , can be obtained,

Z , by predicting whether there is an edge between two nodes through the matrix, and the calculation is shown in Equation (8), as follows:

$$B = \text{sigmoid}(Z \times Z^T), \quad (8)$$

The function can map the input variables to between (0, 1). It is an activation function often employed in neural networks, and Z^T is the transpose matrix of matrix Z .

For structural errors, \mathcal{L}_s , calculating the error between the reconstructed adjacency matrix and the original adjacency matrix, which is Z , predicts whether there is a connection between the pair of nodes in the original network, which can be regarded as a two-classification problem in machine learning, using two-class prediction. As the loss function, the calculation Equation (9) is as follows:

$$\mathcal{L}_s = d_s(A, B) = -(\sum y \log \hat{y} + (1 - y) \log(1 - \hat{y}))/N, \quad (9)$$

where y represents the value of the element in row i and column j in A , and its value is 0 or 1, which means whether there is an edge between node i and node j in the original network. \hat{y} represents the value of the element in B . After $\text{sigmoid}(\cdot)$ function, its value is between 0 and 1. N represents the number of predicted connecting edges. The purpose of constructing the loss function in this way is to make the adjacency matrix reconstructed by the structural decoder more similar to the adjacency matrix of the original network.

3.3.4. Attribute Reconstruction Decoder

Using GCN as the attribute decoder, after the encoder obtains Z , connect a layer of GCN, output the reconstructed attribute matrix Y , and predict the attribute information of each node in the original network. The specific calculation is shown in Equation (10).

$$Y = f_{\text{relu}}(Z, A | W^{(4)}), \quad (10)$$

where, to calculate the node attribute reconstruction error, the output dimension of this layer of GCN is m .

For the attribute error \mathcal{L}_a , calculating the error between the reconstructed attribute matrix and the original attribute matrix can be defined by calculating the distance between F and the reconstructed matrix Y in the vector space \mathcal{L}_a , as shown in Equation (11).

$$\mathcal{L}_a = d_a(F, Y) = \|F - Y\|_F, \quad (11)$$

where represents $\|F - Y\|_F$, the Frobenius norm of the difference between the matrices.

3.3.5. Loss Function Definition

The loss function is jointly determined by the structure loss function, \mathcal{L}_s , the attribute loss function, \mathcal{L}_a , and the relative entropy, \mathcal{L}_{KL} .

We use KL divergence to calculate the relative entropy, and its calculation equation is shown in Equation (12), as follows:

$$\mathcal{L}_{KL} = \text{KL}[q(Z|X, A) || p(Z)], \quad (12)$$

KL divergence is employed to describe the measurement of the difference between two probability distributions. $q(Z|X, A)$ is the probability distribution calculated by the previous GCN encoder, and $p(Z)$ is a priori distribution and standard Gaussian distribution.

To control the contribution of \mathcal{L}_s and \mathcal{L}_a to the total loss function, L , during the training process, $\alpha \in [0, 1]$ parameters are introduced to the weights of \mathcal{L}_s and \mathcal{L}_a . The calculation equation L is shown in Equation (13), as follows:

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_s + \alpha\mathcal{L}_a + \mathcal{L}_{KL}, \quad (13)$$

The larger the value of α , the greater the contribution of attribute reconstruction loss to the total loss function. The minimization process is to make the reconstructed adjacency matrix and attribute matrix closer to the input of the GCN encoder and the probability distribution obtained by the GCN closer to the standard Gaussian distribution.

In the training process, the Adam algorithm is employed in the gradient descent method to minimize the above loss function, L . The learning rate, a_2 , is updated after iterative q times. Finally, the NEAT-VGA model obtains Z that can better reflect the structure information and attribute information of the network.

4. Experiments

For the experiments in this section, three datasets commonly employed in attribute network experiments—Cora (Avaliable at https://neusncp.com/api/view_dataset?dataset_id=173 (accessed on 1 January 2022)), CiteSeer (Avaliable at https://neusncp.com/a-pi/view_dataset?dataset_id=82 (accessed on 1 January 2022)), and PubMed (Avaliable at <https://lincs-data.soe.ucsc.edu/public/Pubmed-Diabetes> (accessed on 1 January 2022))—were chosen [29]. Some basic information about the datasets is shown in Table 2. Link prediction was employed as a downstream application to validate the performance of the model and to analyze the experimental results. An implementation of the proposed method has been made available at <https://github.com/mingshuonie/NEAT-VGA> (accessed on 1 January 2022).

Table 2. Citation network dataset.

Dataset	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
CiteSeer	3327	4732	3703	6
PubMed	19,717	44,338	500	3

4.1. Experimental Setting

The performance of the algorithm’s attribute network embedding is evaluated based on whether the edges are correctly predicted in the experiment. The specific testing procedure is that the edges in the network are randomly divided into three parts, with 85% of the edges serving as the training set and 5% and 10% of the edges removed as the validation and test sets. The network embedding algorithm was employed to generate a low-dimensional vector of network nodes, logistic regression was employed as the classifier for the binary classification problem of link prediction, the AUC (area under the curve) and AP (average precision) metrics were employed for evaluation, and the AUC and AP values of the test results were calculated. Both AUC and AP can evaluate the excellent or poor performance of the classifier. Their values range from 0 to 1, and the closer the value is to 1, the more effective the classifier is.

The experimental data sets are Cora, CiteSeer, and PubMed. AANE [30], GraphSAGE [17], ANRL-WAN [26], VGAE [18], and CSAN [19] algorithms are comparison algorithms. To ensure the fairness of the experiment, each parameter is set according to the reference. The NEAT-VGA algorithm has 200 iterations of feature learning and dimensionality of 128 dimensions for F. The number of iterations for the variational autoencoder part is 400, the learning rate is 0.05, the learning rate is 0.01, the dimensions of the encoding part are all set to 64 dimensions, and the reconstruction weight parameter, α , is 0.5.

4.2. Results and Analysis

The link prediction results obtained from the experiment are shown in Table 3. Furthermore, the histogram and the link prediction experiment results for the AUC and AP value comparison are shown in Figures 4 and 5.

Table 3. Link prediction task performances.

Method	Cora		CiteSeer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
AANE	0.767	0.720	0.785	0.765	0.783	0.754
GraphSAGE	0.795	0.763	0.802	0.791	0.840	0.829
ANRL-WAN	0.832	0.843	0.867	0.848	0.918	0.897
VGAE	0.914	0.903	0.908	0.920	0.944	0.947
CSAN	0.985	0.984	0.950	0.958	0.980	0.977
NEAT-VGA(Ours)	0.987	0.985	0.972	0.968	0.986	0.985

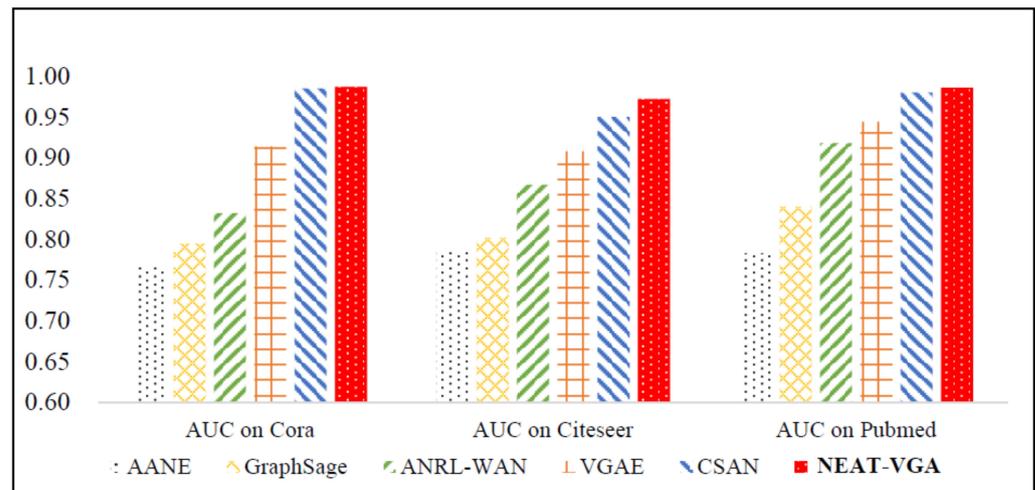


Figure 4. AUC of link prediction task.

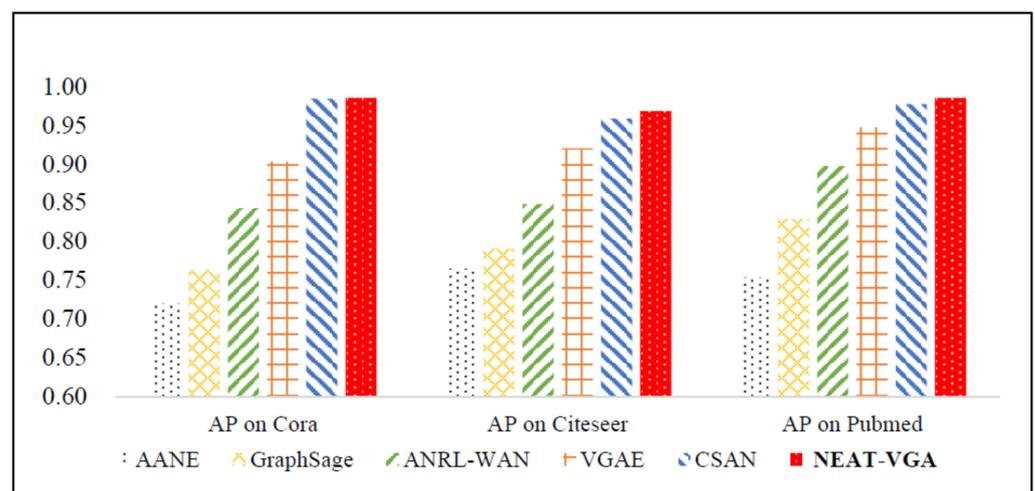


Figure 5. AP of link prediction task.

As can be seen from Table 3 and Figures 4 and 5, the network embedding is carried out on the three citation network data sets, and the link prediction experiment is carried out. The AUC and AP values are all in AANE, GraphSAGE, ANRL, VGAE, CSAN, and NEAT-VGA algorithm is gradually improved. All the compared algorithms are attributed to network embedding algorithms, which utilize the attribute information of the nodes. The NEAT-VGA algorithm proposed in this paper achieves the highest AUC and AP values for link prediction experiments by combining node attribute feature learning

and variational graph autoencoder, both of which are improved compared with the other algorithms.

The NEAT-VGA algorithm holds the best performance based on two metrics—AUC and AP. The best performance among the other compared algorithms is the CSAN algorithm, which uses multiple perceptrons for feature extraction of the attributes of the nodes and considers attribute reconstruction on the decoder. The experimental results of the NEAT-VGA algorithm and the CSAN algorithm are compared and analyzed. On the Cora data, the NEAT-VGA algorithm improved the AUC and AP metrics by 0.2% and 0.1% compared with the CSAN algorithm. On the CiteSeer data, the NEAT-VGA algorithm improved the AUC and AP metrics by 1.3% and 1% over the CSAN algorithm, which is a large improvement. On the PubMed data, the NEAT-VGA algorithm improved the AUC and AP metrics by 0.6% and 0.8% over the CSAN algorithm, which was the highest of the other algorithms.

The NEAT-VGA algorithm is an improvement on the graph autoencoder algorithm, using node attribute feature learning, which results in a significant improvement in link prediction compared with the VGAE algorithm, using a three-layered GCN variational encoder in the encoder and a loss function that reconstructs the structure and attributes jointly. On the Cora data, the AUC and AP metrics on the link prediction task for the NEAT-VGA algorithm improved by 8.0% and 9.1% compared to the VGAE algorithm. On the CiteSeer data, the NEAT-VGA algorithm improved the AUC and AP metrics on the link prediction task by 7.0% and 5.2%, compared with the VGAE algorithm. On the PubMed data, the NEAT-VGA algorithm improved the AUC and AP metrics on the link prediction task by 4.4% and 4.0%, compared with the VGAE algorithm. In summary, the improved NEAT-VGA algorithm had a very significant improvement in results.

The reasons why the NEAT-VGA algorithm can achieve the best results in all three datasets and improve the embedding performance of the attribute network are analyzed from two aspects.

First, the employ of attribute features of the nodes. Except for the NEAT-VGA algorithm and the CSAN algorithm, several other algorithms do not learn the attribute features and directly employ the node vector representations processed by the bag-of-words model and the TF-IDF model. While these vectors can retain and make employ of the node information of the attribute network, these vectors do not extract information about the attribute features of the nodes. The GCN and MLP employed by the CSAN algorithm. The NEAT-VGA algorithm, on the other hand, employs the MHRWAE algorithm to obtain an embedding representation of the node attributes that better captures the attribute information using unbiased random wandering and node neighborhood attribute aggregation.

Secondly, in the attribute network embedding part, the AANE algorithm is implemented by matrix decomposition of the attribute similarity matrix, which is less effective than other algorithms that employ graph neural networks to capture the graph structure and attribute information, such as NEAT-VGA, GraphSAGE, ANRL-WAN, VGAE, and CSAN. Compared with the other graph neural network algorithms, NEAT-VGA also employs network structures and node attribute information well when performing network embedding, employs learned node features as input, effectively extracts higher-order nonlinear information from the network, and considers the contribution of node attribute reconstruction and structure reconstruction to the total loss function.

Overall, the NEAT-VGA algorithm achieves excellent performance on all three citation networks through node-attribute feature learning, variational graph autoencoders, and loss functions that reconstruct the joint structure and attributes, enabling the model to perform well for network embedding and obtain low-dimensional vector representations that retain the structure and attribute information.

5. Conclusions

In this paper, we propose the NEAT-VGA algorithm as a solution for the problem that the attribute information matrix in the attribute network embedding is often high-

dimensional and sparse, which increases the complexity of the algorithm and does not reflect the attribute information of the nodes well; moreover, the GAE algorithm uses a single mapping method in the encoder part and does not consider the attribute reconstruction. The NEAT-VGA algorithm employs the MHRWAE algorithm as a pre-processing method for node attribute feature extraction. The experimental results of link prediction on the citation attribute network dataset verify that the proposed algorithm outperforms other comparison algorithms. However, it is mainly appropriate for the static network and lacks the mining of the dynamic change characteristics of the network, which is the future research direction.

Author Contributions: Conceptualization, D.C., H.Z. and D.W.; formal analysis, M.N. and Z.W.; funding acquisition, D.C. and D.W.; methodology, M.N. and H.Z.; project administration, D.C., M.N. and D.W.; resources, M.N.; software, H.Z.; supervision, D.C. and D.W.; validation, D.C.; visualization, M.N. and Z.W.; writing—original draft, M.N., H.Z. and Z.W.; writing—review and editing, D.C. and D.W. All authors have read and agreed to the published version of the manuscript.

Funding: The paper does not receive any fundings.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank anonymous reviewers for their careful reading and valuable comments, which are all valuable and helpful for revising and improving our paper, as well as the important guiding significance to our research.

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

References

- Asatani, K.; Mori, J.; Ochi, M.; Sakata, I. Detecting trends in academic research from a citation network using network representation learning. *PLoS ONE* **2018**, *13*, e0197260. [[CrossRef](#)] [[PubMed](#)]
- Huang, L.; Jiang, B.; Lv, S.; Liu, Y.; Li, D. Survey on deep learning based recommender systems. *Chin. J. Comput.* **2018**, *41*, 1619–1647.
- Su, S.; Sun, L.; Zhang, Z.; Li, G.; Qu, J. MASTER: Across Multiple social networks, integrate Attribute and STructure Embedding for Reconciliation. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; pp. 3863–3869.
- Wu, X.D.; Li, Y.; Li, L. Influence analysis of online social networks. *Jisuanji Xuebao/Chin. J. Comput.* **2014**, *37*, 735–752. [[CrossRef](#)]
- Haiyan Hong, W.L. Link Prediction Algorithm in Protein-Protein Interaction Network Based on Spatial Mapping. *Comput. Sci.* **2016**, *51*, 413–417, 434.
- Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; Yang, S. Community preserving network embedding. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-2017), San Francisco, CA, USA, 4–9 February 2017; pp. 203–209.
- Rozemberczki, B.; Davies, R.; Sarkar, R.; Sutton, C. Gemsec: Graph embedding with self clustering. In Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM-2019), Vancouver, BC, Canada, 27–30 August 2019; pp. 65–72.
- Blondel, V.D.; Guillaume, J.-L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008. [[CrossRef](#)]
- Chen, D.; Nie, M.; Wang, J.; Kong, Y.; Wang, D.; Huang, X. Community Detection Based on Graph Representation Learning in Evolutionary Networks. *Appl. Sci.* **2021**, *11*, 4497. [[CrossRef](#)]
- Huang, X.; Chen, D.; Ren, T.; Wang, D. A survey of community detection methods in multilayer networks. *Data Min. Knowl. Discov.* **2021**, *35*, 1–45. [[CrossRef](#)]
- Gao, S.; Denoyer, L.; Gallinari, P. Temporal link prediction by integrating content and structure information. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM-2011), Glasgow, UK, 24–28 October 2011; pp. 1169–1174.
- Wang, D.; Nie, M.; Chen, D.; Wan, L.; Huang, X. Node Similarity Index and Community Identification in Bipartite Networks. *J. Internet Technol.* **2021**, *22*, 673–684.
- Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network anomaly detection: Methods, systems and tools. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 303–336. [[CrossRef](#)]
- Bandyopadhyay, S.; Lokesh, N.; Murty, M.N. Outlier aware network embedding for attributed networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-2019), Honolulu, HI, USA, 27 January–1 February 2019; pp. 12–19.

15. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [[CrossRef](#)] [[PubMed](#)]
16. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
17. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS-2017), Long Beach, CA, USA, 4–9 December 2017; pp. 1025–1035.
18. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.
19. Meng, Z.; Liang, S.; Zhang, X.; McCreddie, R.; Ounis, I. Jointly learning representations of nodes and attributes for attributed networks. *ACM Trans. Inf. Syst.* **2020**, *38*, 1–32. [[CrossRef](#)]
20. 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 (KDD-2014), New York, NY, USA, 24–27 August 2014; pp. 701–710.
21. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS-2013), Lake Tahoe, CA, USA, 5–10 December 2013; pp. 3111–3119.
22. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. 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.
23. 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 (KDD-2016), San Francisco, CA, USA, 24–27 August 2016; pp. 855–864.
24. Liu, J.; He, Z.; Wei, L.; Huang, Y. Content to node: Self-translation network embedding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1794–1802.
25. Liao, L.; He, X.; Zhang, H.; Chua, T.-S. Attributed social network embedding. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 2257–2270. [[CrossRef](#)]
26. Zhang, Z.; Yang, H.; Bu, J.; Zhou, S.; Yu, P.; Zhang, J.; 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; pp. 3155–3161.
27. Gjoka, M.; Kurant, M.; Butts, C.T.; Markopoulou, A. Walking in facebook: A case study of unbiased sampling of osns. In Proceedings of the 2010 Proceedings IEEE Infocom, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
28. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML-2014), Beijing, China, 21–26 June 2014; pp. 1188–1196.
29. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; Eliassi-Rad, T. Collective classification in network data. *AI Mag.* **2008**, *29*, 93. [[CrossRef](#)]
30. Huang, X.; Li, J.; Hu, X. Accelerated attributed network embedding. In Proceedings of the 2017 SIAM International Conference on Data Mining (SIAM-2017), Houston, TX, USA, 27–29 April 2017; pp. 633–641.