Adversarial Attention-Based Variational Graph Autoencoder

ZIQIANG WENG, WEIYU ZHANG, AND WEI DOU

School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China
Corresponding author: Weiyu Zhang (zwy@qlu.edu.cn)

This work was supported in part by the National Key Research and Development Project under Grant 2018YFC704, in part by the National Natural Science Foundation of China under Grant 61502259, and in part by the Natural Science Foundation of Shandong Province under Grant ZR2017MF056.

ABSTRACT Autoencoders have been successfully used for graph embedding, and many variants have been proven to effectively express graph data and conduct graph analysis in low-dimensional space. However, previous methods ignore the structure and properties of the reconstructed graph, or they do not consider the potential data distribution in the graph, which typically leads to unsatisfactory graph embedding performance. In this paper, we propose the adversarial attention variational graph autoencoder (AA VGA), which is a novel framework that incorporates attention networks into the encoder part and uses an adversarial mechanism in embedded training. The encoder involves node neighbors in the representation of nodes by stacking attention layers, which can further improve the graph embedding performance of the encoder. At the same time, due to the adversarial mechanism, the distribution of the potential features that are generated by the encoder are closer to the actual distribution of the original graph data; thus, the decoder generates a graph that is closer to the original graph. Experimental results prove that AA VGA performs competitively with state-of-the-art popular graph encoders on three citation datasets.

INDEX TERMS Attention layers, adversarial mechanism, variational graph autoencoder.

I. INTRODUCTION

Compared with Euclidean spatial data, such as image, text, and voice data, non-Euclidean data, such as graph data, have been difficult to process. Therefore, graph embedding algorithms have become a research hotspot. Graph research focuses on node classification [1], link prediction [2], graph classification [3], and graph generation [4], and three types of graph embedding algorithms that are based on factorization, random walks and graph neural networks are established.

Graph embedding algorithms that are based on factorization, for example, graph factorization [5], aim at identifying a balance point between the adjacency matrix and the regularization term such that the generated vector retains most of the information of the adjacency matrix. LINE [6] utilizes this strategy and attempts to embed the adjacency matrix, and the second-order similarity of the nodes is maintained in the vector. HOPE [7] introduced a higher-order similarity matrix and retained the high-order similarity via the generalized singular value decomposition.

A representative algorithm that is based on the random walk is DeepWalk [8]. This method randomly walks upstream and downstream of nodes to generate equal-length sequences as the adjacency features of nodes are utilized for skip-gram model training. Node2Vec [2] applies a weight to the random walk in the direction of DFS and BFS based on the former so that the generated sequence more accurately reflects the structural information of the node.

Recently, neural networks have been used in reliable intelligent path following control [9] and control system stability [10]. The graph embedding algorithm has gradually transitioned to the era of neural networks. Kipf et al. [1] simplified the definition of frequency domain convolution and proposed GCN for conducting the convolution operation in the space domain, which significantly increased the computational efficiency of graph convolution models. Since then, many variants of GCN have been proposed [11]–[15]. GraphSAGE [16] does not restrict the sampling information to the topology of the nodes; instead, it uses the inherent characteristics of the nodes. It also abandons the diffusion mechanism, which involves a massive number of parameters, to realize distributed training of large-scale graph data and inductive learning. Graph attention (GAT) networks [17] conduct aggregation operations on neighboring nodes using an attention mechanism to adaptively allocate neighbor weights.
These methods are supervised graph embedding methods. In recent years, the application of graph data has become increasingly widespread. The structure of the graph is more complicated. In practical application scenarios, many data labels often have high acquisition thresholds. Therefore, it is of substantial value to investigate methods for efficient unsupervised graph representation learning on graph data. The graph autoencoder, which is based on the reconstruction loss, is a typical unsupervised learning approach. GAE and VGAE [18] use an encoder to obtain latent vectors, and a decoder uses latent variables to reconstruct the graph structure. Due to the high-dimensional and complex distributional characteristics of the graph data, the distribution of the feature vector that is encoded by the encoder deviates from the actual distribution. To solve the problem regarding the distribution of the encoded data, DVNE [19] directly embeds the nodes according to a Gaussian distribution and uses the Wasserstein distance as the similarity measure between the distributions, thereby effectively modelling the uncertainty of the nodes in the network. ARGA and ARVGA [20] further introduced the adversarial mechanism [21], which forced the encoder to generate potential vectors that were closer to the data in terms of distribution via adversarial training. CAAE [22] employs Bayesian personalized ranking (BPR) as a discriminative model, which improves the discriminative performance of the discriminator. Although these autoencoders have yielded reasonable results, they do not consider the difference in node importance. Since the importance may differ substantially among neighbor nodes, to learn a robust and stable node embedding, the weights of neighbor nodes should be changed during the training process.

In this study, we focus on the differential representation of neighbor nodes and propose a novel adversarial attention-based variational graph autoencoder (AA VGA). The objectives are to differentiate structural information and to jointly apply adversarial regularization mechanisms to increase the accuracy of graph embedding results. Our encoder utilizes neighbors that differ in terms of weight in the feature representation of the nodes by stacking graph attention layers to generate potential feature vectors, and we add multiple sets of independent attention mechanisms so that the encoder can allocate attention to multiple related features between the central node and neighbor nodes. The adversarial mechanism supervises the encoder and forces its generated vectors to obey the prior distribution of the graph data. Our key contributions are summarized as follows:

- For unsupervised weight learning, we propose an attention-based graph variational autoencoder that infers the feature weights of nodes. The introduction of attention improves the graph embedding learning performance of the graph variational autoencoder, minimizes the reconstruction errors of the graph structure, and captures the highly non-linear network structure.
- To regularize the distribution of the encoded data, we introduce an adversarial component into the attention-based variational graph autoencoder. This component can determine whether the input is from the low-dimensional representation of the graph network or the true distribution of the samples. The discriminator encourages the encoder to generate low-dimensional variables with distributions that are closer to the true distribution of the data and to learn an effective representation of the graph.
- We developed a variation of the graph autoencoder, namely, the adversarial attention-based variational graph autoencoder (AAVGA), for learning a graph representation. We use AAVGA for unsupervised graph link prediction and graph clustering visualization. Experimental results prove that our method performs competitively with other methods.

II. RELATED WORK
A. GRAPH REPRESENTATION LEARNING
The graph representation learning methods can be divided into three categories: factorization-based methods, random-walk-based methods, and deep-learning-based (GNN-based) methods. The factorization-based methods decompose the adjacency matrix, Laplacian matrix, and node similarity matrix to convert the node information into a low-dimensional vector while preserving the structural similarity. These methods, which include GF [5], GraRep [23], and HOPE [7], rely on the decomposition calculation of the correlation matrix; hence, they have high time and space complexities. In recent years, inspired by the word vector method, various methods treat sequences that are generated by random walks in the graph as sentences and nodes as words, such as DeepWalk [8] and Node2Vec [2], which realize large-scale graph embedded learning. However, these methods have shortcomings such as insufficient utilization of graph structure information and fusion graph attribute information. Graph signal processing (GSP) [24] migrates the basic concepts of signal processing, such as Fourier transform and filtering, to graphs to realize the compression, transformation, and reconstruction of the graph signal structure; this laid the foundation for graph embedding learning. Inspired by the definition of graph signal convolution filtering in GSP, a set of neural network theories that are based on graph convolution operations have been developed in recent years. Bruna et al. [25] introduced convolution into graph neural networks and developed a graph convolution network model that was based on the concept of frequency domain convolution operations. Since then, researchers have continuously proposed improvements and expansions of this neural network model that are based on frequency-domain image convolution [26], [27].

With strong end-to-end learning capabilities, GNN can be combined with corresponding unsupervised loss functions to realize unsupervised graph representation learning. According to the design, the loss functions can be divided into two categories: loss functions that are based on contrast and loss functions that are based on reconstruction. Based on the contrast loss, GraphSAGE [16] uses neighbors as context.
Recently, Weihua Hu et al. used subgraphs as the context for contrast learning [28], and DGI [29] regards full graphs as the context. An autoencoder that is based on the reconstruction loss, namely, VGAE [18], uses GNN to encode and learn graph data based on VAE [30]. In ARVGA [20], an adversarial mechanism was added for training the discriminator and forcing the latent vector of the graph data that is generated by the encoder to obey the prior distribution. However, these frameworks do not consider the differences among the graph neighbor nodes during the encoding process, which may cause the original graph structure to be destroyed during the encoding process. We introduce the attention mechanism to solve this problem.

B. ATTENTION MODELS

Our method is inspired by the graph attention network (GAT) [17]. It uses an attention mechanism to conduct aggregation operations on neighboring nodes and realizes the adaptive allocation of neighbor weights. Similar to GraphSAGE [16], the graph attention model retains complete locality information; it can also conduct inductive learning. Based on this, gated attention networks (GaANs) were developed [31]. Furthermore, we combine GAT [17], VGAE [18], and GAN [21], and while ensuring the prior distribution of the data, the expressive ability of the graph encoder is improved.

C. ADVERSARIAL MODELS

The adversarial strategy in our method is derived from GAN [21]: a generator and a discriminator are constructed to optimize each other in a minimax game. GAN can enhance the generalization performance of a graph embedding model. GraphGAN [32] was the first network to utilize the adversarial strategy in graph learning. ANE [33] regards embedding vectors as the generated result and uses GAN [21] as an additional regularization term in available network embedding methods such as DeepWalk [8] by imposing the distribution of the real data as a prior distribution. ARVGA [20] used the above strategies in variational autoencoders but did not consider the difference in attention between nodes. Our method improves this.

III. AAVGA ARCHITECTURE

In this section, we will introduce our proposed AAVGA. The main structure of the framework is illustrated in Fig. 1. It is divided into three significant models: an encoder, a decoder, and a discriminator.

A. GRAPH ATTENTION AUTOENCODER

1) GRAPH ATTENTION ENCODER

VGAE [18] and ARVGA [20] use the original two-layer GCN [1] as the encoder. Our proposed AAVGA combines the strategy of GAT [17] and replaces the two-layer graph convolutional network in the general encoder with a single-layer graph attention network to generate the latent representation of the graph data. The formal definition is as follows: Let the feature vector that corresponds to node \( v_i \) in layer \( L \) be \( h_i \), \( h_i \in \mathbb{R}^{d(l)} \), where \( d(l) \) represents the feature-length of the node. After an aggregation operation with the attention mechanism as the core, the output is the new feature vector \( h'_i \) of each node, where \( h'_i \in \mathbb{R}^{d(l+1)} \) and \( d(l+1) \) represents the length of the output feature vector. We refer to this aggregation operation as a graph attention layer. Assuming that the center node is \( v_i \), we set the weight coefficients of neighbor nodes \( v_j \) to \( v_i \) as:

\[
e_{ij} = a \left( W h_i, W h_j \right)
\]

where \( W = \mathbb{R}^{d(l+1) \times d(l)} \) is the weight parameter of the node feature transformation of this layer and \( a(\cdot) \) is a function that calculates the correlation between two nodes. In principle, here, we can calculate the weight parameter from any node to node \( v_i \) in the graph; however, to simplify the calculation, we limit it to the first-order neighbors. a can use the inner product of the vectors to define a parameter-free correlation calculation \( \langle w_h, w_{h_j} \rangle \). Alternatively, it can be defined as a neural network layer with parameters. If \( a: \mathbb{R}^{d(l+1)} \times \mathbb{R}^{d(l+1)} \rightarrow \mathbb{R} \) is satisfied, a scalar value that represents the correlation between the two vectors is output. We select a single layer of the fully connected layer:

\[
e_{ij} = \text{Leaky ReLU} \left( a^T [W h_i || W h_j] \right)
\]
Its weight parameter is $a \in \mathbb{R}^{2^{d(i+1)}}$, and the activation function is LeakyReLU. To better assign weights, we must normalize the calculated correlation with all neighbors via softmax normalization:

$$
\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp (e_{ij})}{\sum_{k \in N(v_j)} \exp (e_{ik})}
$$

(3)

where $\alpha_{ij}$ is the weight coefficient. Via the processing of (3), it is ensured that the sum of the weight coefficients of all neighbors is 1. Combine (2) and (3) to obtain the complete weight coefficient calculation formula:

$$
\alpha_{ij} = \frac{\exp (\text{Leaky ReLU} (a^T [W_h || W_h]))}{\sum_{k \in N(v_j)} \exp (\text{Leaky ReLU} (a^T [W_h || W_h]))}
$$

(4)

After calculating the above weight coefficients, according to the strategy of weighted summation of the attention mechanism, the new feature vector of node $v_i$ is:

$$
h'_i = \sigma \left( \sum_{j \in N(v_i)} \alpha_{ij} W_h \right)
$$

(5)

To further improve the expression ability of the attention layer, we have also incorporated a multi-head attention mechanism into AAVGA, in which (5) is used to form $K$ groups of independent attention mechanisms, in contrast to GAT [17]. Reduced to the dimension of the output latent feature vector, we replaced the concatenation operation with the averaging operation:

$$
h'_i = \sigma \left( \frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N(v_i)} \alpha_{ijk}^{(k)} w(k) h_j \right)
$$

(6)

By summing multiple sets of independent attention mechanisms, the multi-head attention mechanism can impose the attention distribution on multiple relevant features between the central node and neighbor nodes, which enhances the encoder’s representation ability.

We use an encoder that is based on the attention mechanism to fit $\mu$ and $\sigma$.

$$
\mu = \text{GNN}_\mu (X, A)
$$

(7)

$$
\log \sigma = \text{GNN}_\sigma (X, A)
$$

(8)

Here, $\mu$ is the matrix of mean vectors $\mu_i$, and $\log \sigma$ shares the weights $w$ with $\mu$ in the attention layer.

A variational graph encoder is defined by an inference model:

$$
P(Z|X, A) = \prod_{i=1}^{N} q(z_i|X, A)
$$

(9)

$$
q(z_i|X, A) = \mathcal{N}(z_i, \mu_i, \text{diag}(\sigma_i^2))
$$

(10)

2) DECODER

For the decoder, we follow the strategies of VGAE [18] and ARVGA [20] with the objective of reconstructing graph $A$ via the encoder:

$$
\hat{A} = \sigma (ZZ^T)
$$

(12)

$\hat{A}$ uses two nodes to represent the inner product of the vectors to satisfy the adjacency relationship, thereby predicting whether any two points in the graph are linked:

$$
P(A|Z) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij}|z_i, z_j)
$$

(13)

$$
P(A_{ij} = 1|z_i, z_j) = \sigma \left( z_i^T z_j \right)
$$

(14)

3) LOSS FUNCTION

According to experience, the standard normal distribution is selected as the prior distribution of the hidden variable $z$:

$$
P(Z) = \prod_{i} p(z_i) = \prod_{i} \mathcal{N}(z_i, 0, I)
$$

(15)

The loss of the entire encoder is defined as follows:

$$
\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{k1} = -\mathbb{E}_{q(Z|X, A)} \left[ \log p(A|Z) \right] + \text{KL} \left[ q(Z|X, A) \| p(Z) \right]
$$

(16)

If only $\mathcal{L}_{\text{recon}}$ is used as the loss function, to optimize the performance of the generator, the model will learn that the value of the variance as zero because this is a fixed value that is sampled from the normal distribution, which is helpful for reducing the number of generated samples and the real differences between the samples. However, our objective is to optimize the variational autoencoder. To realize this objective, we add the KL divergence of each independent normal distribution and the standard normal distribution to the loss function and force each normal distribution to approach the standard normal distribution.

B. ADVERSARIAL MODEL

Our adversarial model is composed of two main parts: The encoder in the graph attention autoencoder acts as the generator of the adversarial network. The generator attempts to deceive the discriminator by generating fake data, where fake data refers to the latent variables that are generated from the image data by the encoder. The loss of the generator is as formulated in (16). The discriminator attempts to distinguish whether samples are from real data or generators. The discriminator judges the data from the prior distribution $p_z$ output as positive and the data from the latent variable $z$ output as negative, and its loss is as follows:

$$
- \frac{1}{2} \mathbb{E}_{z \sim p_z} \log D(Z) - \frac{1}{2} \mathbb{E}_{x} \log \left(1 - D(G(X, A))\right)
$$

(17)

We use the Gaussian distribution as the prior distribution of the graph data. We posit that the potential vector $z$ that is generated by the encoder does not satisfy the prior distribution of the data in Euclidean space; hence, we use the standard multilayer perceptron as the discriminator. In the process of embedding and learning, the adversarial regularization constraint is imposed to reduce the deviation of the data distribution in the training process. The main objective of the adversarial model is to jointly train the encoder and the
Algorithm 1 Adversarial Attention-Based Graph Embedding

Input: The node feature matrix \( X \), and the adjacency matrix \( A \).

Output: \( Z \in \mathbb{R}^{n \times d} \)

1: for iterator = 1, 2, 3, ..., T do
2: Generate attention weight \( a^{(k)}_{ij} \) through Eq.(4);
3: Generate latent variables matrix \( Z \) through Eq.(6):
   \[ h_j = \sigma \left( \sum_{k=1}^{K} \sum_{i \in N(u_j)} a^{(k)}_{ij} \nu^{(k)}_j h_j \right) \]
4: for \( n = 1, 2, ..., N \) do
5: Sample m entities \( \{ z^{(1)} \}, \ldots, \{ z^{(m)} \} \) from latent matrix \( Z \);
6: Sample m entities \( \{ a^{(1)} \}, \ldots, \{ a^{(m)} \} \) from the prior distribution \( P_a \);
7: Update the adversarial model through Eq.(18):
\[ \min_G \max_D \mathbb{E}_{z \sim p(z)} \left[ \log D(Z) \right] + \mathbb{E}_{x \sim p(x)} \left[ \log \left( 1 - D(G(X, A)) \right) \right] \]  
8: end for
9: Update the graph autoencoder with its stochastic gradient by Eq.(16) for AAVGA.
10: end for

Train D to maximize the probability of correctly distinguishing the samples from the training data and G. Simultaneously, train G to minimize \( \log(1 - D(G(Z))) \).  

C. ALGORITHM

Algorithm 1 is our proposed framework. In step 2, the attention weight coefficient of each node is obtained, and the corresponding feature vector is obtained in step 3; the feature vectors compose the latent variable matrix \( Z \). Then, in step 5 and step 6, we obtain the same value from the generated result \( Z \) and the real data distribution \( P_z \). The samples are used to train the discriminator. In step 7, we attempt to use the encoder to fool the discriminator, which can also be understood as training the encoder with the discriminator so that the distribution of the data that are generated by the encoder is closer to the true distribution. Finally, in step 9, we use the total loss to update the parameters.

IV. EXPERIMENTS

In this section, we will evaluate AAVGA on three benchmark citation network datasets. The satisfactory performance of the framework is proved on the link prediction task in unsupervised graph learning.

A. DATASETS

We use the three most popular citation datasets (Cora, Citeseer, and Pubmed) in graph embedding learning to evaluate our model. The datasets’ structures are described in Table 1. The nodes correspond to the papers in the dataset, the features are the characteristics of each paper, and the edges represent the link relationships between papers. Consider the Cora dataset as an example: The Cora dataset is composed of machine learning papers and has been highly popular in recent years for graph deep learning. In the dataset, the thesis is classified into one of the following seven categories: case-based, genetic algorithm, neural network, probability method, reinforcement learning, rule learning, and theory. The paper selection criterion is citation by at least one other paper in the final corpus. There are 2708 papers in the corpus. There is a vocabulary that contains multiple words in the dataset; after stemming and removing the endings, only 1433 unique words remain. All words with a document frequency of less than 10 are deleted. The Cora dataset contains 1433 unique words; hence, the feature is 1433-dimensional. The absence and presence of words in the paper are represented by 0 and 1, respectively. The Citeseer and Pubmed datasets are of similar structure to the Cora dataset.

B. METRICS

We regard the graph area that is enclosed by the ROC curve and the x-axis as a comprehensive measurement index, which is denoted as AUC. Another evaluation indicator is AP, which is the graph area that is enclosed by the PR curve and the x-axis. These two indices are the primary evaluation indicators of link prediction tasks. We divide the dataset into a training set, a validation set, and a test set. There are 5% reference edges in the verification set for hyperparameter optimization and 10% reference edges in the test set, which are used to evaluate the performance. To ensure accuracy, we conducted 10 experiments on each dataset. The average and standard error of the experimental results are reported.

C. BASELINES

To prove the satisfactory performance of our proposed AAVGA framework, we compare it with six popular link prediction algorithms:

- **Spectral Clustering** [34]: This is a clustering method that is based on graph theory. The weighted undirected graph is divided into two or more optimal subgraphs such that the internals of the subgraphs are as similar as possible and the distance between the subgraphs is as large as possible to realize the common clustering objective.
- **Deep Walk** [8]: Learning a social representation through a truncated random walk can yield better results if the network has few vertices, and the method is also scalable and can adapt to changes in the network.

| TABLE 1. Citation network data for link prediction. |
|----------------------------------|-----------|-----------|
| Nodes                           | Cora      | Citeseer  | Pubmed   |
| Edges                           | 5429      | 4732      | 44338    |
| Features                        | 1433      | 3703      | 500      |
| Labels                          | 7         | 6         | 3        |
TABLE 2. Link prediction task in the citation network.

| Method   | Cora     | Citeseer | Pubmed |
|----------|----------|----------|--------|
| AUC      | AP       | AUC      | AP     | AUC     | AP     |
| SC       | 84.6 ± 0.01 | 88.5 ± 0.00 | 80.5 ± 0.01 | 85.0 ± 0.01 | 84.2 ± 0.02 | 87.8 ± 0.01 |
| DW       | 83.1 ± 0.01 | 85.0 ± 0.00 | 80.5 ± 0.02 | 83.6 ± 0.01 | 84.4 ± 0.00 | 84.1 ± 0.00 |
| GAE*     | 84.3 ± 0.02 | 88.1 ± 0.01 | 78.7 ± 0.02 | 84.1 ± 0.02 | 82.2 ± 0.01 | 87.4 ± 0.00 |
| VGE*     | 84.0 ± 0.02 | 87.7 ± 0.01 | 78.9 ± 0.03 | 84.1 ± 0.02 | 82.7 ± 0.01 | 87.5 ± 0.01 |
| GAE      | 91.0 ± 0.02 | 92.0 ± 0.03 | 89.5 ± 0.04 | 89.9 ± 0.05 | 96.4 ± 0.00 | 96.5 ± 0.00 |
| VGAE     | 91.4 ± 0.01 | 92.6 ± 0.01 | 90.8 ± 0.02 | 92.0 ± 0.02 | 94.4 ± 0.02 | 94.7 ± 0.02 |
| ARGE     | 92.4 ± 0.003 | 93.2 ± 0.003 | 91.9 ± 0.003 | 93.0 ± 0.003 | 96.8 ± 0.001 | 97.1 ± 0.001 |
| ARVGE    | 92.4 ± 0.004 | 92.6 ± 0.004 | 92.4 ± 0.003 | 93.0 ± 0.003 | 96.5 ± 0.001 | 96.8 ± 0.001 |
| AAVGA    | 93.3 ± 0.005 | 94.2 ± 0.005 | 94.0 ± 0.004 | 94.6 ± 0.004 | 97.2 ± 0.002 | 97.4 ± 0.002 |

- **GAE** [18]: This is an artificial neural network that can learn an efficient representation of input graph data through encoding and decoding via unsupervised learning.
- **VGE** [18]: The encoder does not learn a low-dimensional vector representation of the samples; it learns the distribution of the low-dimensional vector representation. Suppose this representation follows a normal distribution. Then, the distribution that is represented by the low-dimensional vector is sampled to obtain the low-dimensional vector representation.
- **ARGA** [20]: An adversarial mechanism is added based on the graph autoencoder to ensure the consistency of the data distribution during the training process.
- **ARVGE** [20]: Adversarial regularization is incorporated based on a variational graph autoencoder.

For the above benchmark methods, we follow the parameter settings in the corresponding papers.

**D. EXPERIMENTAL SETUP**

In contrast to the autoencoder in the benchmark that uses double-layer GCN to fit the data, for the Cora and Citeseer datasets, we use a graph attention network layer with 64 neurons and multiple attention layers ($k = 3$) when training our framework, namely, AAVGA. We use the Adam algorithm to optimize the encoder and the discriminator, and the learning rates of the encoder and discriminator are both set to 0.001. Training is conducted for 300 iterations. Since the data scale of the Pubmed dataset is much larger than those of the previous two datasets, to increase the expressive power of the encoder, for the Pubmed dataset, we increased the number of neurons to 128 and set the number of multi-head attention layers to 6 and the number of epochs to 1000. The remaining parameters remain unchanged.

**E. EXPERIMENTAL RESULTS**

The results of the link prediction experiment are presented in Table 2. Our method, namely, AAVGA, has yielded excellent results on the three datasets. Except Cora’s AUC, all AUC and AP indicators exceeded 94%. Compared with the other benchmarks, its performance on the Citeseer dataset is the strongest. Compared with VGE, AUC and AP have improved by 3.2% and 2.6%, respectively, whereas ARVGE improved by approximately 1.6%. In comparison with non-encoder graph embedding methods, the performance improvement of our method is pronounced. On the Citeseer dataset, the AUC of AAVGA is 13.5% higher than those of spectral clustering and DeepWalk; AP increased by 9.6% and 11%, respectively. The experimental results demonstrate that competitive performance can be realized by combining the attention mechanism and the adversarial mechanism in the graph encoder.

**F. IN-DEPTH ANALYSIS**

In this section, we will investigate the roles of the attention and adversarial components in the AAVGA framework. In our experiments, we use the following variants of our architecture:

- **AAVG**: The full version of our proposed autoencoder, which includes all components.
- **AVG**: A variant of our architecture that includes all components except the adversarial components.
- **ARVGA**: A variant in which adversarial regularization that is based on a variational graph autoencoder has been incorporated.

We also compare AVG with the full versions of AAVGA and ARVGA on unsupervised link prediction. The relevant experimental settings are as described in Section 4.4. Fig. 2 presents the average AUC and AP on the test node after repeating the ten-iteration training for the three frameworks. AAVGA outperforms AVG and ARVGA on all datasets. Therefore, each component contributes to the overall performance of our architecture. AVG outperforms ARVGA, which proves that the self-attention components contribute the most in our architecture, compared to the adversarial components.

We will discuss the effect of the multi-head attention layer in the encoder on the experimental results. In contrast to previous encoders that used double-layer GCN networks to
fit graph data, we used a single-layer graph attention network because we found that double-layer networks are outperformed by single-layer networks, which may be related to our multi-head attention mechanism. As shown in Fig. 3, we gradually increased the attention level of the bulls from 1 to 9. Among them, the link prediction performance of AA VGA gradually improved and was maximized when $k = 3$. After that, as $k$ increases, the performance fluctuates slightly, but the overall trend is decreasing. After analysis, we conclude that the optimal number of groups for multi-head attention is related to the sparseness of the adjacency matrix $A$. When the sparseness of the graph is low, oversmoothing will quickly occur.

### G. GRAPH VISUALIZATION

To present our results more intuitively and to prove the versatility of the AA VGA framework, we will visualize the graph data in this section. First, we use AA VGA to embed the three processed citation datasets (Cora, Citeseer, and Pubmed). After the training has been completed, the data are encoded by an autoencoder to obtain feature vectors. To realize the two-dimensional visualization of the graph data, we use PCA [35] to reduce the feature vector's dimension. Finally, k-means++ [36] is used to cluster the dimensionality-reduced data. The visualization results are presented in Fig. 4. According to Fig. 4, in which each color represents a category, the citation categories in each dataset are well divided. This proves the satisfactory performance of the AA VGA framework.

**FIGURE 4.** Graph data visualization. From top to bottom: Cora, Citeseer, and Pubmed.
V. CONCLUSION

In this paper, we propose a novel adversarial attention-based variational graph autoencoder (AVGVA). Most encoders use GCN to fit graph data. We believe that this is insufficient for the original graph data due to their expressive power. Therefore, we optimize the GCN in the encoder as a graph attention network while considering the differences in the links between nodes in the graph data, and a more suitable weight is assigned to each node. This improves the representation performance of the encoder. At the same time, the adversarial mechanism ensures the consistency of the distribution of the latent variables that are obtained by the coding and the prior distribution. This increases the robustness of the whole framework. The results of the link prediction experiments demonstrate that our method, namely, AVGVA, outperforms the baselines.

Future work can focus on four aspects: 1. The inner-product decoder can be replaced with a complex generative neural network. 2. The selection of a Gaussian distribution as the prior distribution of the data is not optimal. The exploration of a more reasonable prior distribution is a new direction. 3. In GAEs, many similarity measures have been adopted, and these methods differ; hence, the selection of a suitable similarity measure for a task and model system requires additional investigation. 4. We can combine GAEs with GNNs to render GAEs more powerful, which is a promising future direction.

REFERENCES

[1] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907. [Online]. Available: http://arxiv.org/abs/1609.02907

[2] A. Grover and J. Leskovec, “Node2vec: Scalable feature learning for networks,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 855–864.

[3] Z. Ying, “Hierarchical graph representation learning with differentiable pooling,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 4800–4810.

[4] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, “GraphRNN: Generating realistic graphs with deep auto-regressive models,” 2018, arXiv:1802.08773.

[5] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, “Distributed large-scale natural graph factorization,” in Proc. 22nd Int. Conf. World Wide Web, 2013, pp. 37–48.

[6] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: Large-scale information network embedding,” in Proc. 24th Int. Conf. World Wide Web, 2015, pp. 1067–1077.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 1105–1114.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2014, pp. 701–710.

[9] Y. Wang, W. Zhou, J. Luo, H. Yan, H. Pu, and Y. Peng, “Reliable intelligent path following control for a robotic airship against sensor faults,” IEEE/ASME Trans. Mechatronics, vol. 24, no. 6, pp. 2572–2582, Dec. 2019.

[10] Y. Wang, H. Shen, and D. Duan, “On stabilization of quantized sampled-data Neural-Network-Based control systems,” IEEE Trans. Cybern., vol. 47, no. 10, pp. 3124–3135, Oct. 2017.

[11] M. Schlichtkrull, “Modeling relational data with graph convolutional networks,” in Proc. Eur. Semantic Web Conf. Cham, Switzerland: Springer, 2018, pp. 593–607.

[12] J. Chen, J. Zhu, and L. Song, “Stochastic training of graph convolutional networks with variance reduction,” 2017, arXiv:1710.10568. [Online]. Available: http://arxiv.org/abs/1710.10568

[13] J. Chen, T. Ma, and C. Xiao, “FastGCN: Fast learning with graph convolutional networks via importance sampling,” 2018, arXiv:1801.10247. [Online]. Available: http://arxiv.org/abs/1801.10247

[14] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, “MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing,” 2019, arXiv:1905.00067. [Online]. Available: http://arxiv.org/abs/1905.00067

[15] N. Xu, P. Wang, L. Chen, J. Yao, and J. Zhao, “MR-GNN: Multi-resolution and dual graph neural network for predicting structured entity interactions,” 2019, arXiv:1905.09558. [Online]. Available: http://arxiv.org/abs/1905.09558

[16] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1024–1034.

[17] P. V. Koviä, G. Cucurull, A. Casanova, A. Romero, P. Liä, and Y. Bengio, “Graph attention networks,” 2017, arXiv:1710.10903. [Online]. Available: http://arxiv.org/abs/1710.10903

[18] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, arXiv:1611.07308. [Online]. Available: http://arxiv.org/abs/1611.07308

[19] D. Zhu, P. Cui, D. Wang, and W. Zhu, “Deep variational network embedding in wasserstein space,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 2827–2836.

[20] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized graph autoencoder for graph embedding,” 2018, arXiv:1802.04407. [Online]. Available: http://arxiv.org/abs/1802.04407

[21] J. Goodfellow, “Generative adversarial networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 3, 2014, pp. 2672–2680.

[22] D.-K. Chae, J. A. Shin, and S.-W. Kim, “Collaborative adversarial autoencoders: An effective collaborative filtering model under the GAN framework,” IEEE Access, vol. 7, pp. 37650–37663, 2019.

[23] S. Cao, W. Lu, and Q. Xu, “GraRep: Learning graph representations with global structural information,” in Proc. 24th ACM Int. Conf. Inf. Knowl. Manage., 2015, pp. 891–900.

[24] D. I. Shuman et al., “The emerging field of signal processing on graphs,” IEEE Signal Process. Mag., vol. 30, no. 3, pp. 83–98, 2013.

[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” 2013, arXiv:1312.6203. [Online]. Available: http://arxiv.org/abs/1312.6203

[26] D. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3844–3852.

[27] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, “Graph wavelet neural network,” 2019, arXiv:1904.07785. [Online]. Available: http://arxiv.org/abs/1904.07785

[28] W. Hu, B. Liu, J. Gomes, M. Zimik, P. Liang, V. Pande, and J. Leskovec, “Strategies for pre-training graph neural networks,” 2019, arXiv:1905.12265.

[29] P. V. Koviä, W. Fedus, W. L. Hamilton, P. Liä, Y. Bengio, and R. Devon Hjelm, “Deep graph infomax,” 2018, arXiv:1809.10341. [Online]. Available: http://arxiv.org/abs/1809.10341

[30] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: http://arxiv.org/abs/1312.6114

[31] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, “GaAN: Gated attention networks for learning on large and spatiotemporal graphs,” 2018, arXiv:1803.07294. [Online]. Available: http://arxiv.org/abs/1803.07294

[32] H. Wang, “Graphgan: Graph representation learning with generative adversarial nets,” in Proc. AAAI Conf. Artif. Intell., 2018, pp. 1–8.

[33] Q. Dai, “Adversarial network embedding,” in Proc. Conf. Artif. Intell., 2018, p. 4.

[34] L. Tang and H. Liu, “Leveraging social media networks for classification,” Data Mining Knowl. Discovery, vol. 23, no. 3, pp. 447–478, Nov. 2011.

[35] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” Chemometrics Intell. Lab. Syst., vol. 2, nos. 1–3, pp. 37–52, 1987.

[36] D. Arthur and S. Vassilvitskii, K-Means++: The Advantages of Careful Seeding, Stanford, CA, USA: Stanford, 2006.
ZIQIANG WENG received the B.S. degree from the Department of Information and Computing Science, Qilu University of Technology (Shandong Academy of Sciences), in 2019. He is currently pursuing the master’s degree with the School of Computer Science and Engineering, Qilu University of Technology (Shandong Academy of Sciences). His research interests include machine learning, data mining, and graph neural networks.

WEIYU ZHANG received the Ph.D. degree in computer science from the Beijing University of Posts and Telecommunications, in 2016. He is currently an Associate Professor with the Qilu University of Technology (Shandong Academy of Sciences). Until now, he has published more than ten papers in conferences and journals, such as Neurocomputing, Acta Electronica Sinica, and so on. His current research is sponsored by the Natural Science Foundation of Shandong Province, the Natural Science Foundation of China, and the National Key Research and Development Program. His research interests include machine learning, data mining, social network analysis, and graph neural networks.

WEI DOU received the B.S. degree from the Department of Information and Computing Science, Lvliang University, in 2018. He is currently pursuing the master’s degree with the School of Computer Science and Engineering, Qilu University of Technology (Shandong Academy of Sciences). His research interests include machine learning and network representation learning.