IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 30, 2020, accepted December 20, 2020, date of publication December 24, 2020,
date of current version January 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3047116

Graph Embedding Framework Based on Adversarial and Random Walk Regularization

WEI DOU, WEIYU ZHANG, ZIQIANG WENG, AND ZHONGXIU XIA
School of Computer Science and Technology, Qilu University of Technology (Shangdong 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 Graph embedding aims to represent node structural as well as attribute information into a low-dimensional vector space so that some downstream application tasks such as node classification, link prediction, community detection, and recommendation can be easily performed by using simple machine learning algorithms. The graph convolutional network is a neural network framework for machine learning on graphs. Because of its powerful ability to model graph data, it is currently the best choice for graph embedding. However, most existing graph convolutional network-based embedding algorithms not only ignore the data distribution of the latent codes but also lose the high-order proximity between nodes in a graph, leading to inferior embedding. To mitigate this problem, we investigate how to enforce latent codes to match a prior distribution, and we introduce random walk to preserve high-order proximity in a graph. In this paper, we propose a novel graph embedding framework, Adversarial and Random Walk Regularized Graph Embedding (ARWR-GE), which jointly preserves structural and attribute information. ARWR-GE adopts an adversarial training scheme to enforce the latent codes to match a prior distribution, and by employing the skip-gram model, nodes in a random walk sequence are closer in the latent space. We evaluate our proposed framework by using three real-world datasets on link prediction, graph clustering, and visualization tasks. The results demonstrate that our framework achieves better performance than state-of-the-art graph embedding algorithms.

INDEX TERMS Graph embedding, graph convolutional network, random walk, adversarial scheme.

I. INTRODUCTION

Graphs are universal data structures that can represent complex relational data that ubiquitously exist in the real world, including social networks [1], [2], paper citation networks [3], [4], protein-protein interaction networks [5], [6], etc. Mining valuable information from graph data is an important research topic of data mining. However, the high computational complexity, low parallelizability, and inapplicability of machine learning methods to graph data have made traditional graph analysis algorithms including shortest path, centrality measurement, etc., profoundly challenging. To resolve the above problems, graph embedding aims to learn a project from graph data in the original topological space to low-dimensional vector space while encoding structural and semantic information. The vector representation obtained could effectively support extensive graph analysis tasks including node classification [7], [8], node clustering [9], [10], link prediction [11], [12], graph classification [13], etc. Because of the universality of the embedding vectors, graph embedding technology can be applied to many fields and tasks such as social networks and recommender systems [14] by using the off-the-shelf machine learning method.

Graph embedding algorithms have become one of the indispensable and hot topics in the domain of data mining and machine learning in recent years. These algorithms are mainly divided into three categories: matrix factorization-based methods, random walk-based methods, and deep learning-based methods. The motivation of matrix

The associate editor coordinating the review of this manuscript and approving it for publication was Jing Bi.

factorization-based methods is a graph that can usually be expressed in some form of relation matrix, such as an adjacency matrix and Laplacian matrix. Therefore, this approach is an intuitive way to obtain node embeddings by matrix factorization to achieve the effect of dimensionality reduction. The commonly used factorization methods include singular value decomposition (SVD) and eigenvalue decomposition. For example, GraRep [15] presented a node embedding method by decomposing the global structural information matrix, which integrated various k-step relational information. TADW [16] introduced text features of vertices into the matrix factorization framework. The disadvantage of this type of method is the high computational complexity of matrix factorization, which makes it difficult to expand the application on large-scale graph data.

Capturing the graph structure is a primary concern while generating node embeddings. The random walk has been widely used in graph data analysis to capture the structure information, the basic idea of which is that the embedding of nodes in a random walk path should be similar. Unsupervised representation learning methods have been widely studied and applied in the field of natural language processing (NLP). Deepwalk [17] experimentally verified that the nodes in the random walk sequence follow the power-law as well as the words in the document. Therefore, the walking path obtained when performing random walks can be taken as the sentence in the corpus, and the vertices in the network can be regarded as vocabulary in the text corpus. Then, combined with mature NLP technology Word2Vec [18], nodes can be mapped to a low-dimensional vector space. Some subsequent algorithms are improvements and extensions to DeepWalk, including Node2Vec [19], DDRW [20], FeatWalk [21], etc.

In recent years, deep learning has developed rapidly in various fields to learn multiple levels of feature representations from complex and non-linear data, and graph data are no exception [22]. GCN [23] simplified the definition of frequency-domain convolution and performed the convolution operation in the space domain, representing a semisupervised framework for node classification. However, the labels of graph data in the real world are extremely sparse, which greatly limits the performance of supervised or semi-supervised graph embedding algorithms. Therefore, various unsupervised graph embedding algorithms have been proposed one after another. SDNE [24] is an autoencoder-based model and simultaneously optimizes the first-order and second-order proximity to characterize the local and global graph structure. GAE [25] extends the graph convolutional neural network to an unsupervised framework.

With its simple encoder-decoder structure and efficient encode capability, GAE has found utility in many fields. However, GAE has two flaws that cause it to have inferior embeddings. First, the objective of GAE is to reconstruct the adjacency matrix of the original graph, and thus, its reconstruction loss ignores the distribution of the latent representation. Second, it cannot capture high-order proximity because too many convolutional layers of the encoder will lead to over-smoothing. In this paper, we propose Adversarial and Random Walk Regularized Graph Embedding (ARWR-GE) to make up for the above deficiencies and obtain a more robust embedding. We employ an adversarial training scheme to enforce the latent representation to match a prior distribution. The high-order proximity of the graph is a structural characteristic that cannot be ignored. We perform random walks on the graph and combine the skip-gram model to maximize the likelihood of nodes sequences. Therefore, ARWR-GE can not only enforce the latent codes to match a prior distribution but also preserve high-order proximity, which makes the embedding more robust and restores the structural characteristics of the graph. In this paper, the proposed framework regularizes the graph embedding by adversarial training scheme and random walk. Our main contributions are summarized as follows.

- We introduce an adversarial training scheme to discriminate whether a sample is generated from the encoder-generated embedding or a prior distribution.
- We employ random walk and skip-gram models to bring nodes that are directly or indirectly connected in the original graph closer in latent vector space.
- We conduct extensive experiments on three datasets through three tasks: link prediction, node classification, and visualization. The experimental results demonstrate the effectiveness of the proposed model.

The remainder of the paper is organized as follows. Section II discusses the related works. We detail our ARWR-GE framework completely in Section IV. In Section V, we describe the experimental setup and analyze the experimental results. Finally, we conclude our work in Section VI.

II. RELATED WORKS

Graph embedding represents the original graph information in low-dimensional vectors, which can be facilitated in graph analysis tasks such as link prediction, community discovery, and node clustering. Matrix factorization, random walk, and deep neural networks are three common methods in graph embedding.

Matrix factorization-based embedding methods factorize the adjacency matrix or Laplacian matrix of a graph. LLE [26], LE [27], and other spectral methods aim to preserve the local geometry structure of the data and represent them with a lower dimension space. These approaches are parts of dimensionality reduction techniques and can be regarded as the pioneers of graph embedding. HOPE [28] proposed a high-order proximity embedding that decomposed the high-order proximity matrix rather than the adjacency matrix using a generalized SVD. GraRep [15] first constructed K state transition probability matrices $A^k(k = 1, 2, \ldots, K)$ and then performed SVD decomposition of these K matrices to obtain the low-dimensional representation of step K. Finally, the K representations were spliced together to form the final representation in order to ensure that the network representation can depict the global information. M-NMF [29] is based on

the framework of non-negative matrix factorization (NMF) and incorporates community structure in the learning of graph embeddings.

DeepWalk [17] first applied random walk to graph embedding, opening a new era of graph embedding. Inspired by the language model Word2Vec [18], DeepWalk adopts the random walk strategy to generate several paths for each node in a graph and then sends them to the skip-gram [30] model to learn graph embedding. Node2Vec [19] changed the random walk strategy of DeepWalk, introduced biased random walk into it, and explored the homophily and structural equivalence of graph by combining depth-first search and breadth-first search. Struc2Vec [31] constructed a multilayer graph, where each layer denotes a hierarchy in measuring the structural similarity, and then applied random walk followed by skip-gram learning on the multilayer graph to generate the embedding of each node. DDWR [20] combined DeepWalk and SVM to learn the node embeddings that well captured the topological structure while also being discriminative for the node classification task.

Some graph embedding methods based on depth models have been proposed [32], [33]. DNGR [34] used the PMI matrix (pointwise mutual information matrix) to express the probability of co-occurrence between nodes while improving the previous method of utilizing SVD to decompose the PMI matrix (using the stacked denoising autoencoder for feature extraction). ANE [35] mainly consisted of two components, i.e., a structure-preserving component and an adversarial learning component. The autoencoder component is used to capture the global proximity, and the adversarial component preserves the local proximity [36].

However, the abovementioned graph embedding algorithms do not consider the graph content and cannot be directly applied to attributed graph embedding. The attributed graph embedding method aims at combining the graph structure and node attribute information to obtain a node low-dimensional vector representation that fully characterizes the graph. TADW [16] proved that DeepWalk is equivalent to matrix factorization. On this basis, TADW decomposes the adjacency matrix and constrains it with a text representation matrix to obtain node representations that simultaneously depict structural information and attribute information. TriDNR [37] coupled DeepWalk and Doc2Vec [38] model, taking into account information regarding structure, attributes, and labels. FeatWalk [21] and Gat2Vec [39] have similar ideas: they both design a random walk with compatible attributes and structures and then employ a shallow neural network to learn the node embeddings. DANE [40] uses two autoencoders to encode the structure and attribute information separately, constrains their consistency and complementary, and then stitches two vectors as the final representation. MVC-DNE [41] also uses two deep autoencoders to learn the graph embedding of the structure and attributes through the idea of multiview learning. ANRL [42] employs a neighbor enhancement decoder model that takes node features as input while reconstructing the target neighbors.

The graph convolutional neural network is a framework that applies convolutional ideas to attribute graph embedding and shows powerful performance in processing graph data. GCN is the first-order approximation of spectral convolution, which makes the graph convolution neural network widely used, and some variants of GCN have also been proposed. GAE is a typical approach that applies GCN to unsupervised graph embedding algorithms; it uses the GCN layer as the encoder, and the decoder reconstructs the adjacency matrix of the graph. ARGA [43], an extension of GAE, was proposed as an adversarial training scheme to regularize the latent codes. RWR-GAE [45] adds random walk on the basis of the graph autoencoder, which improves the embedding effect of GAE. AAVGA [46] replaced the graph convolutional neural network in ARGA with a graph attention network, paying more attention to the importance of node neighbors.

III. METHODOLOGY

In this section, we present the details of the proposed framework ARWR-GE. First, we formally introduce the notation and problem definition in our work, and then, we detail each component of the framework. The main structure of the frame is shown in Fig. 1.

TABLE 1. Notations.

Notations	Descriptions
G	the given graph
$\mathcal{V} = \{v_1, v_2, ..., v_n\}$	set of vertices in the given graph
$\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$	set of edges in the given graph
$n = \lvert\mathcal{V}\rvert$	number of nodes
$\lvert\mathcal{E}\rvert$	number of edges
$\mathbf{A} \in \mathbb{R}^{n \times n}$	the adjacency matrix of given graph
$\mathbf{X} \in \mathbb{R}^{n \times m}$	the node attribute matrix of given graph
d	embedding dimension
$\mathbf{Z} \in \mathbb{R}^{n \times d}$	the embedding matrix
c	context window size
r	walks per vertex
l	walk length

A. NOTATIONS AND PROBLEM DEFINITION

The summary of notations is listed in Table 1. Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be a graph, where $\mathcal{V}$ denotes the set of n nodes and $\mathcal{E}$ represents the set of edges. $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix describing the edges and $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the attribute matrix representing m attributes potentially associated with the n nodes. Let $\mathbf{A}_i$ be the i^{th} row vector of $\mathbf{A}$ and let $\mathbf{A}_{ij} = 1$ if there is an edge between nodes i and j; otherwise, $\mathbf{A}_{ij} = 0$. $x_i \in \mathbf{X}$ indicates the content features associated with each node v_i.

Problem Definition: Given a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, we aim to map the node $v_i \in \mathcal{V}$ to low-dimensional vectors $z_i \in \mathbb{R}^d$ with the formal format $f : (\mathbf{A}, \mathbf{X}) \mapsto \mathbf{Z}$. We want $\mathbf{Z}$ to capture the topological structure as well as the attribute proximity.

B. GRAPH AUTOENCODER

1) GRAPH CONVOLUTIONAL NETWORKS

GCN is the first-order local approximation of the convolution of the spectral graph. Its input is the adjacency matrix and attribute matrix of the graph, which can convert the structural information and attribute information into a low-dimensional vector representation. Given a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{Z})$, the graph convolutional network is a spectral convolution function $f(\mathbf{Z}^l, \mathbf{A}|\mathbf{W}^l)$:

$$\mathbf{Z}^{l+1} = f(\mathbf{Z}^l, \mathbf{A}|\mathbf{W}^l) \tag{1}$$

where $\mathbf{Z}^l$ is the output of the l^{th} layer after convolution and the input of the $l+1^{th}$ layer convolution, $\mathbf{Z}^0 = \mathbf{X}$, and $\mathbf{W}^l$ is the parameter associated with the l^{th} layer. In GCN, the graph convolutional network can be expressed with the function $f(\mathbf{Z}^l, \mathbf{A}|\mathbf{W}^l)$ as follows:

$$f(\mathbf{Z}^l, \mathbf{A}|\mathbf{W}^l) = \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^l\mathbf{W}^l) \tag{2}$$

Here, $\widetilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$ is the diagonal node degree matrix of $\widetilde{\mathbf{A}}$ and σ is the activation function, $\mathbf{W}^l$ is the parameter matrix to be learned in the l^{th} layer neural network.

2) GRAPH AUTOENCODERS

Similar to ordinary autoencoders, graph autoencoders are composed of an encoder and a decoder. However, the graph autoencoder employs GCN layers as an encoder to obtain the latent representation of nodes and the inner product as the decoder. Specifically, we use a two-layer GCN as the encoder:

$$\mathbf{Z}^1 = f_{ReLU}(\mathbf{X}, \mathbf{A}|\mathbf{W}^0) \tag{3}$$

$$\mathbf{Z}^2 = f_{linear}(\mathbf{Z}^1, \mathbf{A}|\mathbf{W}^1) \tag{4}$$

The first layer uses $\mathbf{X}$ and $\mathbf{A}$ as the input of the graph autoencoder, outputs the feature matrix $\mathbf{Z}^1$ of the first layer, and then uses $\mathbf{Z}^1$ as the input of the second layer GCN to generate the embedding matrix $\mathbf{Z}^2$ of nodes. *ReLU* and *linear* are the types of neural network activation functions of the first and second layers, respectively, and $\mathbf{W}^0$ and $\mathbf{W}^1$ are the parameter matrix of the first and second layers, respectively. The embedding generated by the encoder $\mathcal{G}(\mathbf{Z}, \mathbf{A}) = q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$ is used to reconstruct the original graph:

$$\hat{\mathbf{A}} = sigmoid(\mathbf{Z}\mathbf{Z}^T) \tag{5}$$

A high-quality $\mathbf{Z}$ should make the reconstructed adjacency matrix as similar to the original adjacency matrix as possible because the adjacency matrix determines the structure of the graph. We minimize the reconstruction error by:

$$\mathcal{L}_g = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] \tag{6}$$

If a graph autoencoder performs well, then the latent representation that it generates contains the attribute information of the node, and the adjacency matrix reconstructed by the latent representation is similar to the original adjacency matrix. The graph autoencoder is different from the ordinary automatic encoder. It uses GCN [23] as the encoder and the inner product of the embedding as the decoder to reconstruct the original graph. The ordinary autoencoder can stack hidden layers to make the model obtain stronger representation ability, but this method is not advisable in a graph autoencoder. Since too many GCN layers will cause the model to exhibit over-smoothing, which will have a negative impact on the final result, only two layers of GCN are used in GAE [25] to obtain the best performance of the model.

C. VARIATIONAL GRAPH AUTOENCODER

The variational graph autoencoder is the variant of the graph autoencoder, which also uses GCN to generate the node representation. The difference is that the variational graph autoencoder first determines a (multi-dimensional) Gaussian distribution through GCN and then samples $\mathbf{Z}$ from this distribution:

$$q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{n} q(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}) \tag{7}$$

$$q(\mathbf{z_i} \mid \mathbf{X}, \mathbf{A}) = \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \mathrm{diag}\left(\boldsymbol{\sigma}^2\right)\right) \tag{8}$$

where $\boldsymbol{\mu} = \mathbf{Z}^2$ is a matrix of mean vectors $\mathbf{z}_i$, and $\boldsymbol{\sigma} = f_{linear}(\mathbf{Z}^1, \mathbf{A}|\mathbf{W}^{1\prime})$ is the covariance matrix. $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ share the parameter of the first layer convolution $\mathbf{W}^0$. The decoder reconstructs the adjacency matrix with $\mathbf{z}$:

$$\hat{\mathbf{A}}_{ij} = sigmoid\left(\mathbf{z}_i^T\mathbf{z}_j\right) \tag{9}$$

The variational graph autoencoder still aims to make the reconstructed graph as similar as possible to the original graph. In addition, it intends for the distribution calculated by GCN to be as similar as possible to the standard Gaussian. Therefore, the loss function is composed of cross entropy and KL divergence. To train the variational graph autoencoder, we optimize the variational lower bound:

$$\mathcal{L}_{vg} = \mathbb{E}_{q(\mathbf{Z}|(\mathbf{X},\mathbf{A}))}[\log p(\mathbf{A} \mid \mathbf{Z})] - \mathbf{KL}[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})\|p(\mathbf{Z})] \tag{10}$$

Among them, $q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})$ is the distribution calculated by GCN, and $p(\mathbf{Z})$ is the standard Gaussian.

D. ADVERSARIAL TRAINING SCHEME

We introduce the adversarial training mechanism to force the encoder to generate embedding to match a prior distribution. We train a discriminator based on a multi-layer perceptron, which is trained to distinguish whether the embedding is sampled from the prior distribution (positive simple) or generated by the encoder (negative simple). The loss function of discriminator $\mathcal{D}(\cdot)$ is defined as follows:

$$\mathcal{L}_d = -\mathbb{E}_{\mathbf{z}\sim p_z}\log\mathcal{D}(\mathbf{Z}) - \mathbb{E}_{\mathbf{X}}\log(1-\mathcal{D}(\mathcal{G}(\mathbf{X}, \mathbf{A}))) \tag{11}$$

where p_z is the prior distribution, and we choose the Gaussian distribution in this paper. It is worth noting that the encoder $\mathcal{G}(\cdot)$ is shared by the autoencoder and adversarial model, and

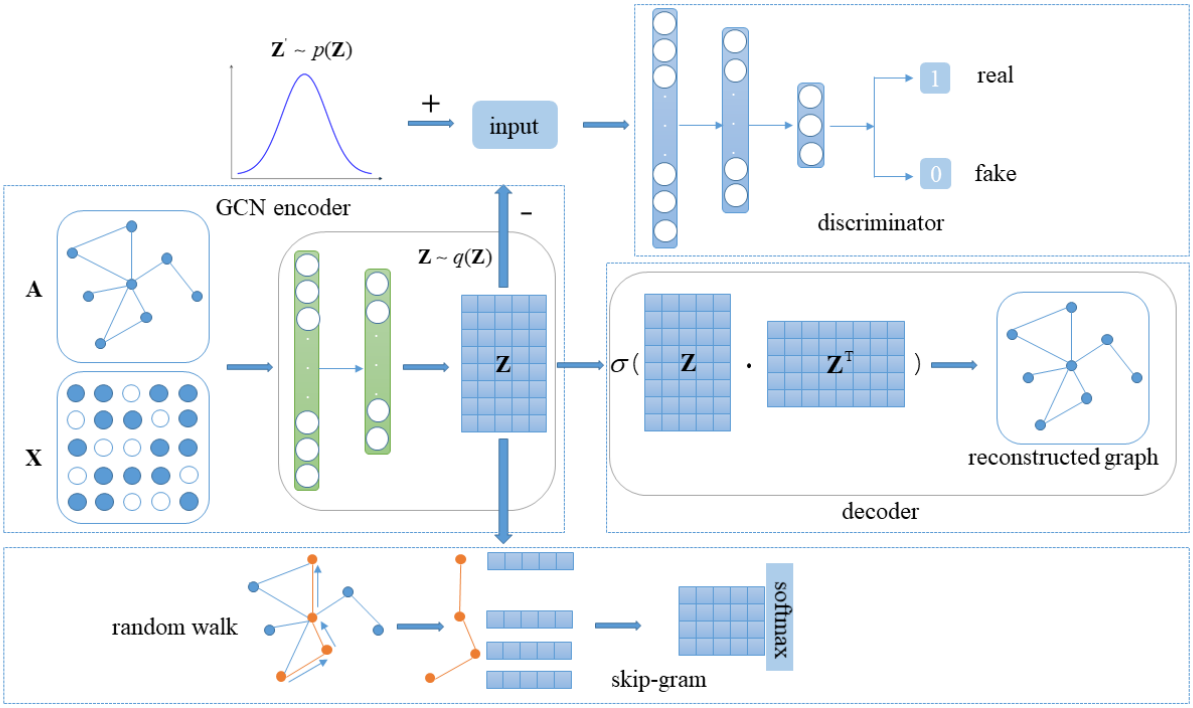

FIGURE 1. The architecture of the proposed ARWR-GE model. The middle part of the figure is a graph autoencoder, which inputs are the adjacency matrix A and attribute matrix X of the graph, and the embedding Z is generated by the graph convolution encoder to complete the reconstruction of A. The top half of the framework corresponds to an adversarial network, which is trained to distinguish whether the data come from the embedding generated by the encoder or the prior distribution. The data sampled from the prior distribution are regarded as positive samples and represented by a plus sign; the data sampled from the latent representation output by the encoder are regarded as negative samples and represented by a minus sign. The bottom half shows the random walk and skip-gram model, which function to make the embeddings in the same random walk path closer to each other in latent space.

jointly optimizing $\mathcal{D}(\cdot)$ and $\mathcal{G}(\cdot)$ can make the potential code match the prior distribution:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{\mathbf{z}\sim p_z}[\log \mathcal{D}(\mathbf{Z})] + \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{X}, \mathbf{A})))] \quad (12)$$

E. RANDOM WALK + SKIP-GRAM

The adversarial mechanism only constrains the distribution of the latent representation. Although the adjacency matrix of the graph is reconstructed at the decoding end of the graph autoencoder, the two-layer GCN is insufficient to capture the structural information of the graph. Inspired by DeepWalk [17] and Node2Vec [19], which assume that nodes with similar context should be similar in latent semantic space, we combine random walk and the skip-gram model to mine high-level structural information of the graph. Based on this approach, we propose to incorporate node attribute information together with a high-level structure. Specifically, the objective function minimizes the following log probability of the skip-gram model by giving current node v_i with its embedding $\mathbf{z}_i$:

$$\mathcal{L}_{sg} = -\sum_{i=1}^{|\mathcal{V}|} \sum_{v_j \in C_i} \log \Pr(v_j|\mathbf{Z}_i) \quad (13)$$

where $C_i = \{v_{i-w}, \ldots, v_{i+w}\}$ is the node context in random walk generated by center node v_i, and w is the window size. The conditional probability of $\Pr(v_j|\mathbf{Z}_i)$ is the likelihood of the center node context v_j by giving its embedding $\mathbf{z}_i$. Therefore, the probability is formulated as:

$$\Pr(v_j|\mathbf{z}_i) = \frac{\exp(\mathbf{v}_j^T \cdot \mathbf{z}_i)}{\sum_{s=1}^{n} \exp(\mathbf{v}_s^T \cdot \mathbf{z}_i)} \quad (14)$$

Here, $\mathbf{Z}_i$ is the embedding vector generated by the graph encoder. It can be seen that the skip-gram model and graph autoencoder share the same encoder. Therefore, the skip-gram model can optimize the parameters of the encoder so that the embedding generated by the encoder can capture the high-order proximity of the graph. $\mathbf{v}_j$ is the corresponding embedding when node v_j is the context for the node v_i. Eq.(14) expresses the probability obtained by the softmax function, and directly calculating Eq.(14) is computationally expensive because we require iterate through all the nodes in a graph when computing the conditional probability of $\Pr(v_j|\mathbf{z}_i)$. To speed up the training process and guarantee the validity of the calculation results, we adopt the negative sampling strategy proposed in [30] that samples multiple negative samples according to some noisy distributions. The negative samples here refer to nodes that are not related to the central node. Therefore, for central node $v_i \in \mathcal{V}$ to generate

its contexts $v_j \in \mathcal{C}_i$, we simplify Eq.(14) to the following objective:

$$\log \sigma(\mathbf{v}_j^T \cdot \mathbf{z}_i) + \sum_{n=1}^{|\text{neg}|} \mathbb{E}_{v_n \sim P_n(v)}[\log \sigma(-\mathbf{v}_n^T \cdot \mathbf{z}_i)] \quad (15)$$

where $\sigma(\cdot)$ is the sigmoid function, and |neg| is the number of negative samples. We randomly sample the negative contexts v_n according to the probability $P_n(v) \propto d_v^{3/4}$ as suggested in [30], where d_v is the degree of node v. For all nodes $\mathcal{V}$, we have the following loss function:

$$\mathcal{L}_{sg} = -\sum_{i=1}^{|\mathcal{V}|} \sum_{v_j \in C_i} [\log \sigma\left(\mathbf{v}_j^T \cdot \mathbf{z}_i\right) + \sum_{n=1}^{|\text{neg}|} \mathbb{E}_{v_n \sim P_n(v)} \log \sigma\left(-\mathbf{v}_n^T \cdot \mathbf{z}_i\right)] \quad (16)$$

After optimizing the Eq.(16), node attribute information consistent with high-order structure information can be preserved in the same representation space.

IV. OUR MODEL

Combining the advantages of the above components, we propose a graph embedding framework based on the graph autoencoder: **Adversarial and Random walk Regularized Graph Embedding**(**ARWR-GE**), which can transform high-dimensional complex graph data into a robust, low-dimensional latent representation that preserves the graph structure (local and high-order) and attribute information. It can be seen from Fig. 1 that our proposed novel framework is divided into three main components; namely, the middle components is the graph autoencoder, the upper tier is an adversarial network and the lower tier is the skip-gram module, which cooperates to generate high-quality graph embedding. The learning algorithm is summarized in Algorithm 1. Variational **ARWR-GE** is named **ARWR-VGE**.

First, we generate a fixed-length node sequence for each node in the graph to prepare for subsequent optimization of the skip-gram module. In step 3, we employ the GCN encoder to encode the local structural information and attribute information into $\mathbf{Z}$. Then, we take the same number of samples from $\mathbf{Z}$ and the real data distribution p_z in steps 5 and 6 to update the discriminator with the cross-entropy cost computed in step 7. After training the discriminator K rounds, the graph encoder will try to deceive the discriminator and update itself with the generated gradient in step 9. Step 10 calculates the gradient according to Equation (16) and updates the parameters of the skip-gram module and graph encoder. The graph encoder refers to the encoding part of the model, and the graph autoencoder includes the decoder. The graph autoencoder and skip-gram model share connections to the encoder, which captures the node attributes as well as graph structure information. Finally, the graph autoencoder generates gradient according to Equation (6) or Equation (10) and updates its own parameters. After T rounds of iterative

Algorithm 1 Adversarial and Random Walk Regularized Graph Embedding

Input: attributed graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$; graph embedding dimension d; window size c; walks per vertex r; walk length l; the number of iterations T; the number of steps for iterating discriminator K

Output: node embedding $\mathbf{Z} \in \mathbb{R}^{n \times d}$

1: Construct node context corpus C by starting r times of random walks with length l at each node
2: **for** iterator = 1,2,3, …, T **do**
3: Generate latent variables matrix $\mathbf{Z}$ through Equation (4);
4: **for** k = 1,2, …, K **do**
5: Sample m entities $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from latent matrix $\mathbf{Z}$
6: Sample m entities $\{\mathbf{a}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from the prior distribution p_z
7: Update the discriminator with its stochastic gradient by Equation (11)
8: **end for**
9: Update the graph autoencoder with its stochastic gradient by Equation (6) or (10);
10: Update skip-gram module with its stochastic gradient by Equation (16);
11: Update the graph autoencoder with its stochastic gradient by Equation (6) or (10);
12: **end for**

training, we take the output $\mathbf{Z}$ of the last layer of the graph encoder as the final graph embedding. In the 9^{th} and 11^{th} steps of **Algorithm 1**, we updated the parameters of the graph autoencoder twice because our model is completed under the common regularization of the adversarial mechanism and the skip-gram model. Each regularization needs to update the parameters of the graph autoencoder to make the final latent representation $\mathbf{Z}$ obtain double gain.

V. EXPERIMENTS

To validate the effectiveness of the proposed framework, we conduct link prediction and node clustering tasks on three public real-world datasets. The detailed descriptions of the datasets and baselines are shown in this section.

A. DATASETS

We evaluate our framework on three popular citation graphs, namely, Cora,[1] Citeseer, and Pubmed.[2] These graphs are also used in previous works [23], [43]. Specifically, nodes in the citation graph denote papers, and edges describe the citation relations between papers. The attributes of each node are the bag-of-words representations of the corresponding paper. We summarize statistics of the datasets in Table 2 with more descriptions as follows:

[1]https://snap.stanford.edu/data/.
[2]https://linqs.soe.ucsc.edu/data.

TABLE 2. Statistics of the datasets.

Datasets	# Nodes	# Edges	# Attributes	# Labels
Cora	2,708	5,429	1,433	7
Citeseer	3,312	4,732	3,703	6
Pubmed	19,717	44,338	500	3

- **Cora** consists of 2,708 papers from the machine learning area, and these papers are divided into seven categories: case-based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning, and theory. The citation graph consists of 5,429 edges that represent citation relationships. The attributes of a paper are described as a 1433-dimensional feature vector.
- **Citeseer** consists of 3,312 scientific papers from the CiteSeer web database and are categorized into six classes: artificial intelligence, agents, machine language, data base, information retrieval, and HCI. The citation graph consists of 4,732 links. Each paper in the dataset is described by a bag-of-words vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3,703 unique words.
- **Pubmed** consists of 19,717 scientific papers from the PubMed database about diabetes classified into three classes: diabetes mellitus experimental, diabetes mellitus type 1, and diabetes mellitus type 2. The citation graph consists of 44,338 links. Each publication in the dataset forms a dictionary that is made up of 500 unique words.

B. BASELINES

We employ the following graph embedding models as baselines.

- **DeepWalk** [17] is an influential graph embedding algorithm that uses truncated random walks to generate node sequences and then employs the skip-gram model to obtain the latent embedding.
- **GAE** [25] converts the topological information and attribute information of the graph into a low-dimensional embedding through the graph convolutional neural network, and then, the decoder reconstructs the adjacency moment through the inner product.
- **VGAE** [25] employs a graph convolutional neural network to let the model learn some distributions, then samples latent representations (or embedding) from these distributions, and finally reconstructs the original graph using the obtained latent representations.
- **ARGA** [43] based on the graph autoencoder introduces an adversarial training mechanism to force the embedding to match the prior distribution.
- **ARVGA** [43] is the variant of ARGA and employs the idea of variation in the data encoding stage.
- **RWR-GAE** [45] proposes a technique that uses random walk to standardize the node representation learned by the graph autoencoder.
- **AAVGA** [46] employs a graph attention encoder involving node neighbors in the representation of nodes by stacking attention layers.

We compare ARWR-GE and ARWR-VGE to these state-of-the-art graph embedding algorithms: the first is structure-based methods, and the remaining methods use both structure and attribute information. For all baselines, we used the code released by the original authors and set the default parameters according to their report.

C. PARAMETER SETTINGS

With reference to ARGA [43], ANRL [42] and GAE [25], based on experimental verification, the parameter settings of our model are shown in Table 3. For the random walk, we set the window size w as 10, walk length l as 80, walks per node r as 10. As the Pubmed dataset is relatively large, we set the window size w as 70 and walks per node r as 30. For the encoder, we stacked two GCN layers, each with 32 and 16 neurons. The discriminator consists of two hidden layers with 16 and 64 neurons. For Cora and Citeseer, we set the number of iteration training of the model to 200, set the learning rate of the graph autoencoder and other components to 0.001 and use the Adam algorithm for optimization. Due to the relatively large scale of the Pubmed data, we iterate over 2000 training models, the learning rate of graph autoencoder and skip-gram is set to 0.001, and the discriminator is set to 0.008.

TABLE 3. Parameter settings of each dataset.

Parameters	Cora	Citeseer	Pubmed
window size w	10	10	70
walk length l	80	80	80
walks per node r	10	10	30
number of neurons in the encoder	32-16	32-16	32-16
learning rate of the graph autoencoder	0.001	0.001	0.001
number of neurons in the discriminator	16-64	16-64	16-64
learning rate of the discriminator	0.001	0.001	0.008
number of iterations	200	200	2000

D. LINK PREDICTION

Link prediction is an important graph analysis task and aims to predict or infer either missing interactions or links that may appear in the future, e.g., friend recommendation in social networks.

1) METRICS

We randomly hide 10% of existing links as positive instances and generate an equal number of nonexisting links as negative instances to test the performance, with 5% edges for hyperparameter tuning, and the rest are used for training. Subsequently, we obtain the node embedding, which will be applied to predict positive links. We rank both positive and

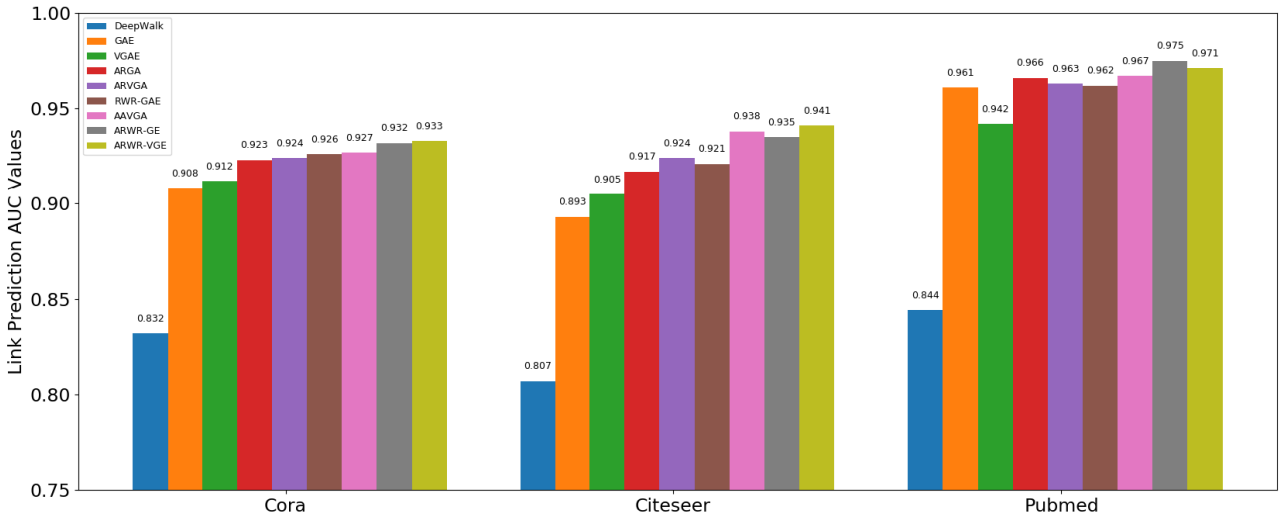

FIGURE 2. Link prediction results on the three real-world datasets. The y-axis describes the AUC value of each method, while the x-axis shows the name of the datasets.

negative instance optimization based on cosine similarity. Finally, we employ the area under the ROC curve (AUC) values to evaluate the ranking list; the higher the value is, the better the model performance.

2) EXPERIMENTAL RESULTS

We perform link prediction tasks on the three datasets, and the results are shown in Fig. (2). We summarize the following observations and analyses:

- A general observation that we can draw from Fig. (2) is that ARWR-GE and ARWR-VGE consistently obtain the best AUC results on the three datasets. In particular, our method achieves approximately 2% AUC improvement over the state-of-the-art baseline RWR-GAE on the Citeseer dataset. On the Pubmed dataset, our method improves by 1.3% and 0.8% compared to the state-of-the-art methods RWR-GAE and AAVGA, respectively. Although RWR-GAE also introduces random walk, we further incorporate the adversarial training scheme to regularize the latent representation to learn a robust graph embedding. We can see that AAVGA's result is also good because it introduces an attention mechanism in the coding process, which is equivalent to the expansion of neighborhood node information. However, the experimental results show that our method is better because our method can explore higher-order neighborhood information by random walk. The experimental results demonstrate that competitive performance can be realized by combining the adversarial mechanism and random walk in the graph autoencoder.
- Compared with DeepWalk, which only uses structure information, our method achieves an AUC improvement of approximately 13%. This substantial improvement effectively illustrates the necessity of considering attribute information in graph embedding. We argue that when the graph links are sparse, it is not enough to learn the structural information to mine the potential link information. The introduction of attribute information can alleviate this problem.
- Distinctly, on the three datasets, ARWR-GE and ARWR-VGE are better than the good-performing ARGA or ARVGA, proving that based on the combination of adversarial training, the combination of the random walk strategy and skip-gram model can expand structural structure information. Therefore, the latent embedding learned by the ARWR-GE and ARWR-VGE models has better reasoning and prediction performance.

E. NODE CLUSTERING

One of the most important tasks in graph mining and graph analysis is node clustering, the goal of which is to infer the clusters in graphs based on the graph embedding. Many practical applications can be transformed into node clustering problems, such as community detection in social networks [22]. After learning embedding through the graph embedding algorithm, we can use an off-the-shelf clustering algorithm such as K-means to achieve the node clustering task.

1) METRICS

We employ five quality metrics to measure the clustering result: Accuracy (ACC), Normalized Mutual Information (NMI), precision, F-score (F1), and Adjusted Rand index (ARI) [45].

2) EXPERIMENTAL RESULTS

Similarly to previous works [19], [43], we perform node clustering task on three datasets, and the results are shown in in Table 4, Table 5 and Table 6. We we make the following observations:

TABLE 4. Node clustering results on the Cora dataset.

Model	ACC	NMI	F1	Precision	ARI
DeepWalk	0.486	0.328	0.394	0.362	0.244
GAE	0.594	0.426	0.591	0.593	0.345
VGAE	0.607	0.434	0.609	0.608	0.346
ARGA	0.638	0.447	0.617	0.644	0.353
ARVGA	0.635	0.450	0.625	0.623	0.372
RWR-GAE	0.657	0.471	0.618	0.629	0.400
AAVGA	0.654	0.462	0.625	0.627	0.387
ARWR-GE	0.667	**0.482**	0.620	0.631	0.417
ARWR-VGE	**0.687**	0.457	**0.669**	**0.687**	**0.419**

TABLE 5. Node clustering results on the Citeseer dataset.

Model	ACC	NMI	F1	Precision	ARI
DeepWalk	0.335	0.086	0.269	0.246	0.091
GAE	0.406	0.174	0.372	0.416	0.123
VGAE	0.343	0.154	0.306	0.347	0.093
ARGA	0.572	0.348	0.544	0.572	0.341
ARVGA	0.542	0.260	0.527	0.547	0.243
RWR-GAE	0.609	0.349	0.577	0.595	0.335
AAVGA	0.587	0.361	0.553	0.581	**0.346**
ARWR-GE	**0.617**	**0.358**	**0.587**	**0.606**	**0.346**
ARWR-VGE	0.614	0.337	0.583	0.596	0.337

TABLE 6. Node clustering results on the Pubmed dataset.

Model	ACC	NMI	F1	Precision	ARI
GAE	0.695	0.328	0.687	0.720	0.322
VGAE	0.606	0.217	0.611	0.612	0.194
ARWR-GE	0.727	**0.356**	0.716	0.729	0.372
ARWR-VGE	**0.737**	0.337	**0.727**	**0.737**	**0.382**

- ARWR-GE and ARWR-VGE achieve the best performance among all the baseline methods. Immediately following other methods combining structure and attributes, the clustering results have also been significantly improved compared to those of the structure-based methods. This result further justifies the usefulness of attribute information for graph embedding. The effective combination of structural information and attribute information can enable the model to learn higher-quality embedding.
- It is obvious from these tables that the clustering results of ARWR-GE and ARGA are both better than those of GAE. For the Cora dataset, we find that compared to VAGE, ARWR-VGE improves the ACC by approximately 8% and the ARI by approximately 7.3%. It can be proved that the introduction of the adversarial mechanism in the training process of the graph autoencoder forces the embedding to be more smooth and robust.
- In particular, the results show that ARWR-GE and ARWR-VGE have achieved a dramatic improvement in all five metrics compared with ARGA and ARVGA. For instance, on the Citeseer dataset, our method's performance exhibits a relative increase of 4.5%, 4.3%, 3.4% and % w.r.t. ACC, F-score and Precision compared to the best baseline ARGA result, which shows our method can better capture similar nodes in the graph through the optimization of random walk and the skip-gram model.

(a) ARWR-GE (b) ARGA

FIGURE 3. Visualization of the embedding results. On the left is the visualization of ARWR-GE in 2D space, and on the right is the visualization of the ARGA embedding results.

F. VISUALIZATION

We visualized the embedding results of the Cora dataset, as shown in Figure 3. First, we obtained the low-dimensional embedding of nodes, and then, we used the t-distributed stochastic neighbor embedding (t-SNE) to visualize graph data in 2D space. A general observation that we can draw from the figure is that the boundary of different types of nodes in the two-dimensional space is visible after the dimensionality reduction of the embedding obtained by ARWR-GE, while that of ARGA is somewhat blurred.

VI. CONCLUSION

In this paper, we propose a novel graph embedding framework, Adversarial and Random Walk Regularized Graph Embedding (ARWR-GE), as well as its variational form (ARWR-VGE), which jointly preserves structural and attribute information. To overcome the defects of the existing methods that ignore the data distribution of latent embedding, we introduce an adversarial training mechanism to force the embedding to match a prior distribution. In addition, we employ a combination of random walk and skip-gram model to enable the model to capture the structural information of graphs beyond the capabilities of graph autoencoders. We evaluate ARWR-GE and ARWR-VGE on link prediction, node clustering, and graph visualization tasks. The experimental results on several real-world datasets show that our proposed ARWR-GE and ARWR-VGE outperform representative state-of-the-art embedding approaches. In the future, we plan to extend our methods to heterogeneous graphs.

REFERENCES

[1] M. Krishnamurthy, P. Marcinek, K. M. Malik, and M. Afzal, "Representing social network patient data as evidence-based knowledge to support decision making in disease progression for comorbidities," *IEEE Access*, vol. 6, pp. 12951–12965, 2018.
[2] T. Zhou, H. Ma, M. R. Lyu and, and I. King, "Userrec: A user recommendation framework in social tagging systems," in *Proc. 24th AAAI Conf. Artif. Intell.*, Atlanta, GA, USA, 2010, pp. 1486–1491.
[3] T. M. V. Le and H. W. Lauw, "Probabilistic latent document network embedding," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 270–279.

[4] X. Wang, X. Zhang, and S. Xu, "Patent co-citation networks of fortune 500 companies," *Scientometrics*, vol. 88, no. 3, pp. 761–770, Sep. 2011.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 1024–1034.

[6] H. Chen, W. Guo, J. Shen, L. Wang, and J. Song, "Structural principles analysis of host-pathogen protein-protein interactions: A structural bioinformatics survey," *IEEE Access*, vol. 6, pp. 11760–11771, 2018.

[7] C. Li, S. Wang, L. He, P. S. Yu, Y. Liang, and Z. Li, "SSDMV: Semi-supervised deep social spammer detection by multi-view data fusion," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 247–256.

[8] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," *in: Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016, pp. 3889–3895.

[9] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3670–3676.

[10] H. Gao, J. Pei, and H. Huang, "ProGAN: Network embedding via proximity generative adversarial network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, vol. 2019, pp. 1308–1316.

[11] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. 32th AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, vol. 2018, pp. 2508–2515.

[12] Z. Li, Z. Liu, J. Huang, G. Tang, Y. Duan, Z. Zhang, and Y. Yang, "MV-GCN: Multi-view graph convolutional networks for link prediction," *IEEE Access*, vol. 7, pp. 176317–176328, 2019.

[13] E. Ranjan, S. Sanyal, and P. P. Talukdar, "ASAP: Adaptive structure aware pooling for learning hierarchical graph representations," in *Proc. 35th AAAI Conf. Artif. Intell.*, New York, NY, USA, 2020, pp. 5470–5477.

[14] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 3307–3313.

[15] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Melbourne, VIC, Australia, 2015, pp. 891–900.

[16] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, Buenos Aires, Argentina, 2015, pp. 2111–2117.

[17] 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 (KDD)*, New York, NY, USA, 2014, pp. 701–710.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2013 pp. 3111–3119.

[19] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

[20] J. Li, J. Zhu, and B. Zhang, "Discriminative deep random walk for network classification," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Berlin, Germany2016, pp. 1004–1013.

[21] X. Huang, Q. Song, F. Yang, and X. Hu, "Large-scale heterogeneous feature embedding," in *Proc. 33th AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, vol. 2019, pp. 3878–3885.

[22] B. Li and D. Pi, "Network representation learning: A systematic literature review," *Neural Comput. Appl.*, vol. 32, pp. 16647–16679, Apr. 2019.

[23] 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

[24] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1225–1234.

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

[26] S. T. Roweis, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[27] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2001, pp. 585–591.

[28] 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.

[29] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31th AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 203–209.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781

[31] L. F. Ribeiro, P. H. Saverese, D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," *Proc. 23nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Halifax, NS, Canada, 2017, pp. 385–394.

[32] L. Zhang, T. Luo, F. Zhang, and Y. Wu, "A recommendation model based on deep neural network," *IEEE Access*, vol. 6, pp. 9454–9463, 2018.

[33] T. Li, J. Zhang, P. S. Yu, Y. Zhang, and Y. Yan, "Deep dynamic network embedding for link prediction," *IEEE Access*, vol. 6, pp. 29219–29230, 2018.

[34] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 30th AAAI. Artif. Intell.*, Phoenix, AZ, USA, 2016, pp. 1145–1152.

[35] Y. Xiao, D. Xiao, B. Hu, and C. Shi, "ANE: Network embedding via adversarial autoencoders," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2018, pp. 66–73.

[36] B. Yu, Y. Li, C. Zhang, K. Pan, and Y. Xie, "Enhancing attributed network embedding via similarity measure," *IEEE Access*, vol. 7, pp. 166235–166245, 2019.

[37] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016, pp. 1895–1901.

[38] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1188–1196.

[39] N. Sheikh, Z. Kefato, and A. Montresor, "GAT2VEC: Representation learning for attributed graphs," *Computing*, vol. 101, no. 3, pp. 187–209, Mar. 2019.

[40] H. Gao and H. Huang, "Deep attributed network embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3364–3370.

[41] D. Yang, S. Wang, C. Li, X. Hang, and Z. Li, "From properties to links: Deep network embedding on incomplete graphs," in *Proc. 26th ACM Int. Conf. Inf. Knowl. Manage.*, Singapore, 2017, pp. 367–376.

[42] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang, "ANRL: Attributed network representation learning via deep neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3155–3161.

[43] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2609–2615.

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

[45] P. Y. Huang and R. Frederking, "RWR-GAE: Random walk regularization for graph auto encoders," 2019, *arXiv:1908.04003*. [Online]. Available: http://arxiv.org/abs/1908.04003

[46] Z. Weng, W. Zhang, and W. Dou, "Adversarial attention-based variational graph autoencoder," *IEEE Access*, vol. 8, pp. 152637–152645, 2020.

WEI DOU received the B.S. degree from the Department of Computer Science and Technology, Lvliang University, in 2018. He is currently pursuing the master's degree with the School of Computer Science and Technology, Qilu University of Technology (Shangdong Academy of Sciences). His research interests include machine learning, network representation learning, 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. He has vast research interests in machine learning, data mining, social network analysis, and graph neural networks. Until now, he has published more than ten papers in conferences and journals such as *Neurocomputing and Acta Electronica Sinica*. Now, his 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.

ZIQIANG WENG received the B.S. degree from the Department of Computing Science, Jining University, in 2019. He is currently pursuing the master's degree with the School of Computer Science and Engineering, Qilu University of Technology (Shangdong Academy of Sciences). His research interests include deep learning, data mining, and graph neural networks.

ZHONGXIU XIA received the B.S. degree from the Department of Computing Science, Jining University, in 2019. He is currently pursuing the master's degree with the School of Computer Science and Engineering, Qilu University of Technology (Shangdong Academy of Sciences). His research interests include deep learning and social network analysis.