Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

GAN-based Deep Neural Networks for Graph Representation Learning

Ming Zhao[1] | Yinglong Zhang*[2]

[1]Fujian Key Laboratory of Granular Computing and Application, Minnan Normal University, Zhangzhou, Fujian, China
[2]School of Physics and Information Engineering, Minnan Normal University, Zhangzhou, Fujian, China

Correspondence
*Yinglong Zhang, School of Physics and Information Engineering, Minnan Normal University, Zhangzhou, Fujian, 363000, China. Email: zhang_yinglong@126.com

Present Address
Fujian Key Laboratory of Granular Computing and Application, Minnan Normal University, Zhangzhou, Fujian, 363000, China

Abstract

Graph representation learning has attracted increasing attention in a variety of applications that involve learning on non-Euclidean data. Recently, generative adversarial networks(GAN) have been increasingly applied to the field of graph representation learning, and large progress has been made. However, most GAN-based graph representation learning methods use adversarial learning strategies directly on the update of the vector representation instead of the embedding mechanism, which does not make full use of the essential advantages of GAN. The essential advantage of GAN is the final embedding mechanism rather than the embedding representation itself. To address this problem, we propose to use adversarial idea on the reconstruction mechanism of deep autoencoders. Specifically, the generator and the discriminator are the two basic components of the GAN structure. We use the deep autoencoder as the discriminator, which can capture the highly non-linear structure of the graph. In addition, the generator another generative model is introduced into the adversarial learning system as a competitor. A series of empirical results proved the effectiveness of the new approach.

KEYWORDS:
Graph representations learning; Generative adversarial network; Autoencoder; Graph structure

1 | INTRODUCTION

The graph is a common representation of information management in many real-world problems, such as social networks, word coexistence networks, and communication networks. Graph representation learning is beneficial to many real-world applications, for example, recommendation system[1], text embedding[2], social network analysis[3], link prediction[4], node classification[5], visualization[6] and knowledge graph representation[7], etc. The essential graph representation learning is to learn the low-dimensional representation of the vertices, which encodes the relationship and structure information between the vertices in the graph[8,2,9].

In terms of technology used, the existing methods fall into two major categories. One is the model based on matrix factorization. Some early methods, such as Local linear representation[10] used the linear combination of adjacent vertices as the vertex representation to preserve the local structure, converts the problem into the calculation of feature vectors. Besides, Laplace eigenmap[11] also preserved the local features of the data, turning the optimization problem into a generalized eigenvalue decomposition problem. The directed graph embedding[12] further extended the Laplace method. Different from the previous methods, [13] introduced modularity into the loss function. Furthermore, DeepWalk[8] obtained the truncated vertex sequence through random walk, and further expands the local structure. Afterward, TADW[14] proved that the algorithm process of DeepWalk is essentially a matrix factorization process.

Another graph representation learning method is based on a depth model. SDNE[15] is one of the earliest deep learning algorithms for graph representation learning. SDNE adopts a semi-supervised learning method, which effectively maintains the first-order and second-order approximation. After that,[16] focuses on maintaining global structure, which approximates the embedding of target vertex by aggregating neighborhood embedding, and learns graph embedding recursively. On this basis,[17] introduces nonlinear tuple similarity. There is another method that focuses on graph attributes[18]. Combining nonparametric probabilistic modeling with deep neural networks, DepthLGP[19] proposes a deeply transformed high-order laplacian gaussian process approach.

In recent years, GAN based models have been introduced into graph representation learning. For example, GraphGAN[20] is a shallow model that captures the network structure by fitting the connection distribution. ANE[21] proposed a model to capture the network structure properties, and designed minimax optimization problems to enhance robustness. And A-RNE[22] focused on sampling high-quality negative vertices to achieve a better similarity ranking among vertices pairs. CANE[23] designed a novel adversarial learning framework to capture the network communities.

However, most methods focus on preserving diverse network structures and properties, while ignoring the fact that networks are usually noisy and incomplete. The existing methods usually directly update the representation of the node to match an arbitrary distribution, but this strategy does not consider the noise data of the graph itself. We believe that applying the adversarial learning strategy to the embedding mechanism that maps vertices to the latent space is a better way, rather than just requiring representation distribution to follow priors.

In this paper, we propose a GAN-based deep neural network (DnnGAN) for graph representation learning. We adopt a structural deep autoencoder as a discriminator, which tries to restructure the neighbor relationship of the vertices and accurately predicts whether the input vertex pair is positive according to the middle layer representation.

The generator is devoted to generating deceptive vertex pairs(the vertex pairs that are close to each other in the representation space)to deceive the discriminator. The training process can be formulated as a two-player game, the generator and the discriminator maximize and minimize the model objective function, respectively.

The key contributions of the proposed method are summarized as follows:

1) We apply the adversarial learning strategy to the embedding mechanism of the deep autoencoder to tackle the issues caused by connection noises and network incompleteness.
2) We formulate an adversarial learning structure that includes structured deep autoencoders, which generate a more robust embedding representation by jointly considering both locality preserving and global reconstruction constraints.
3) We conduct extensive experiments on graph reconstruction, link prediction, and node classification tasks to evaluate the effectiveness of the proposed method.

The rest of the manuscript is organized as follows: The next section describes the related work. Section 3 covers the proposed method, and the experimental evaluation is discussed in Section 4. A conclusion and future work are provided in Section 5.

2 | RELATED WORK

2.1 | Graph Embedding Methods

Most of the existing works are aimed at exploring the structural features of graphs and preserving them in low-dimensional embedding representation[10,8,24].

On the one hand, a considerable number of methods based on matrix factorization have been proposed[10,11,14,9,25]. Among them, Locally linear embedding[10] designed a high-dimensional space nearest neighbor linear reconstruction to capture the local neighborhood structures, and Laplace eigenmap preserved local distances based on edge weights. TADW[14] proved that DeepWalk[8] is essentially a matrix decomposition process, and DeepWalk incorporates the text features via matrix factorization. Further, GraRep[9] derived a transition matrix to preserve high-order proximity, and then HOPE[25] preserved asymmetric transitivity in approximating the high-order proximity.

On the other hand, neural network-based methods have gained a lot of attention. GCN[26] as an efficient variant of convolutional neural networks used convolution operator on the graph and iteratively aggregates the embedding of neighbors for vertices, and centrality information of vertices is utilized in[27] to learn vertice importance of link formation. In addition, the vertex popularity is incorporated as a structural feature in RaRE[24]. Similar to GCN, GraphSAGE[28] used a sample and aggregate method to learn vertex features and neighbor relationships. There are also some graph representation Learning learning methods based on hyper-graphs, such as HGNN[29] applied the spectral convolution to hyper-graphs, and DHNE[30] adopted the deep autoencoder to capture the structural information of the hyper-edges.

2.2 | Adversarial Representation Learning

Most recently, there is growing concerned about the application of GAN in graph representation learning. For instance, ARGA[31] used an autoencoder to reconstruct the topology to obtain a compact representation, and used adversarial training to make the representation match the prior distribution. NetRA[32] used the LSTM autoencoder with vertex sequences as input to learn vector representations that are regularized by locality preserving constraint and generative adversarial training process. ProGAN[33] attempted to capture undiscovered proximity by simulating and approximating the true proximity distribution through triples. GraphGAN[20] employed both generative and discriminative models to fit the connectivity distribution of direct neighbor vertices of target vertice. AMIL[34] used the mutual information between the autoencoder and the GAN to learn embedding representation. Most of them only use adversarial learning to update the embedding representation rather than learning the embedding mechanism, and they only apply to attribute graphs. Differently, in this paper, we leverage the mapping mechanism of the autoencoder as part of the adversarial learning to reduce the interference caused by connection noises and graph incompleteness. In addition, the locality-preserving and global reconstruction constraints are jointly considered during the training process to improve the robustness and generalization ability of the model.

3 | GAN-BASED DEEP NEURAL NETWORK

3.1 | Problem Definition

Graph representation learning attempts to map high-dimensional graph data into low-dimensional embedding space while preserving the graph structure. To be more specific, given a unweighted and undirected graph $\mathcal{G} = (V, E)(n = |V|)$, with $V = \{v_i\}_{i=1}^{n}$ as vertex set, $E = \{e_{i,j}\}_{i,j=1}^{n}$ as the edge set. A is a 0-1 matrix, which used to describe the connectivity between vertices, i.e., adjacency matrix. $A_{i,j} = 1$ means that v_i and v_j are connected, otherwise the opposite. The objective of the graph representation is to convert each of the n vertices into an embedding vector $y_i \in R^k$ (where $k \ll n$), and preserves the captured structural information.

3.2 | Framework of DnnGAN

We formulate the GAN structure nets for graph representation learning, and improve the ability to capture the global structure by adopting a deep autoencoder for the GAN discriminator. The framework is shown in Figure 1.

The generator (G) generates (or selects) the vertices most likely to be connected to v_c from all vertices V according to the current parameter ϕ, which is the negative samples in Figure 1. We denote G as $G_\phi\left(\cdot \mid v_c\right)$. Correspondingly, the positive samples are selected from the edge set E. Negative samples and positive samples together serve as the input of the discriminator. On the other hand, the discriminator(D) uses an autoencoder framework similar to a deep autoencoder[15]. It reconstructs the neighborhood structure of the vertices, so that the vertices with similar neighbor structures have more similar representations in the middle layer. Meanwhile, the D calculates the similarity score of the vertex pair according to the middle layer representation (the Y in Figure 1) as the output. We denote D as $D_\theta\left(v, v_c\right)$. The discrimination loss is calculated by comparing the output with the ground truth. We choose the cross-entropy function ($tf.nn.sigmoid\_cross\_entropy\_with\_logits$) imported from TensorFlow in Python for calculation. The model updates the parameters based on the discrimination loss. In addition, the D will additionally update the parameters based on the reconstruction loss unsupervised.

In the adversarial training process, G and D are continuously improved. The connectivity distribution of the generator is getting closer and closer to the true distribution $p_t\left(\cdot \mid v_c\right)$. The D assigns a higher score to the positive vertex pairs and reduces the score of the negative vertex pairs. Like other regular GAN models[35,36,37,38,33], our model is essentially a minimax game, and its objective function is expressed as:

$$\min_{\phi} \max_{\theta} O(G, D) = \sum_{c=1}^{n} \left(E_{v \sim p_t(\cdot|v_c)} \left[\log D_\theta\left(v, v_c\right)\right] + E_{v \sim G_\phi(\cdot|v_c)} \left[\log\left(1 - D_\theta\left(v, v_c\right)\right)\right] \right). \tag{1}$$

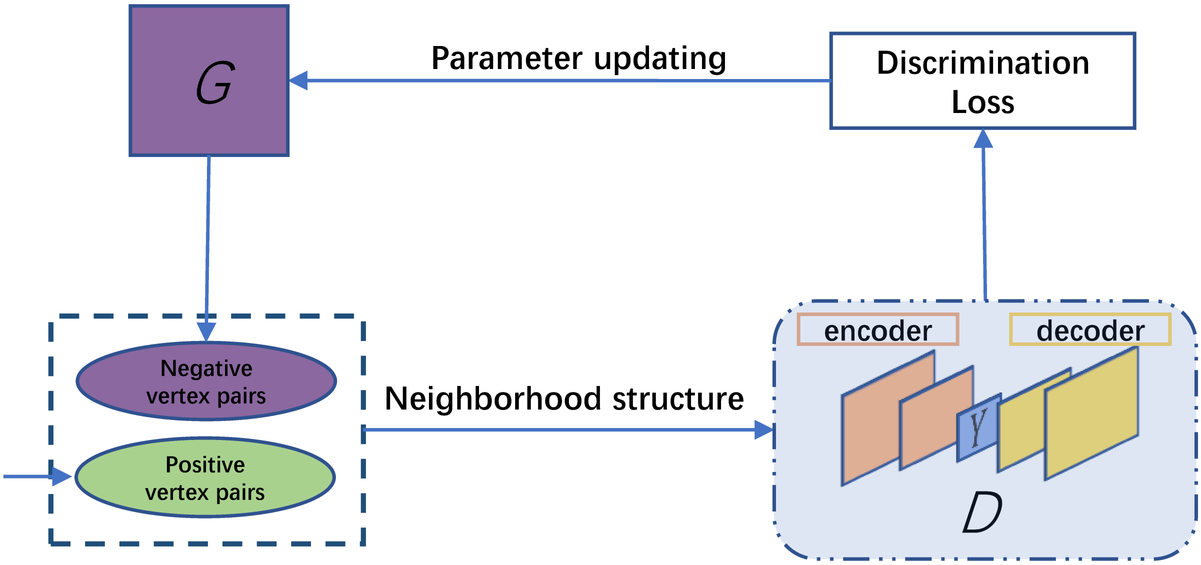

FIGURE 1 Illustration of DnnGAN framework.

The G wants to generate samples that minimize the objective function, and the D tries to maximize the objective function. The competitive relationship drives them to improve their models until the G is indistinguishable from the true connectivity distribution.

3.2.1 | Discriminator

In view of it can capture the global structure by reconstructing the second-order proximity[15], we adopt the autoencoder as the discriminator. The vector $\mathbf{a}_i = [A_{i,1}, A_{i,2}, \cdots, A_{i,n}]^T$ characterizes the neighborhood structure of the vertex, and similarity between $\mathbf{a}_i$ and $\mathbf{a}_j$ determines the second-order proximity between vertices v_i and v_j. Thus D takes $\mathbf{a}_i$ as input, and the processing of data in the discriminator is shown in Figure 2. The encoder encodes $\mathbf{a}_i$ into a d-dimensional embedding representation $\mathbf{y}_i$, and the decoder maps $\mathbf{y}_i$ to the reconstruction space, which denotes as $\hat{\mathbf{y}}_i$.

One of the purposes of the discriminator is to reduce reconstruction errors. The loss function is as follows:

$$O_{unsup} = \sum_{i=1}^{n} \|\hat{\mathbf{a}}_i - \mathbf{a}_i\|_2^2 . \tag{2}$$

The corresponding algorithm is presented in Alg. 2. In real situations, graphs are usually sparse and incomplete in fact. Most of the elements in $\mathbf{a}_i$ are zero. However, the reconstruction of non-zero elements is more important, so we increase the penalty for reconstruction errors of non-zero elements as follows:

$$\begin{aligned} O_{unsup} &= \sum_{i=1}^{n} \left\| \left(\hat{\mathbf{a}}_i - \mathbf{a}_i\right) \circ \mathbf{t_i} \right\|_2^2 \\ &= \|(\hat{A} - A) \circ T\|_F^2, \end{aligned} \tag{3}$$

where $\circ$ means the Hadamard product. $\mathbf{t_i} = [T_{i,1}, T_{i,2}, \cdots, T_{i,n}]^T$. If $a_{i,j} = 0, t_{i,j} = 1,$ else $t_{i,j} = \alpha > 1$, α is the penalty coefficient for non-zero element reconstruction error.

Another purpose of the discriminator is to improve the ability to correctly distinguish between positive and negative samples. That is to maximize the objective function:

$$O_{sup} = \begin{cases} \log D_\theta\left(v, v_c\right), & if\ v \sim p_t\left(\cdot \mid v_c\right); \\ 1 - \log D_\theta\left(v, v_c\right), & if\ v \sim G_\phi\left(\cdot \mid v_c\right), \end{cases} \tag{4}$$

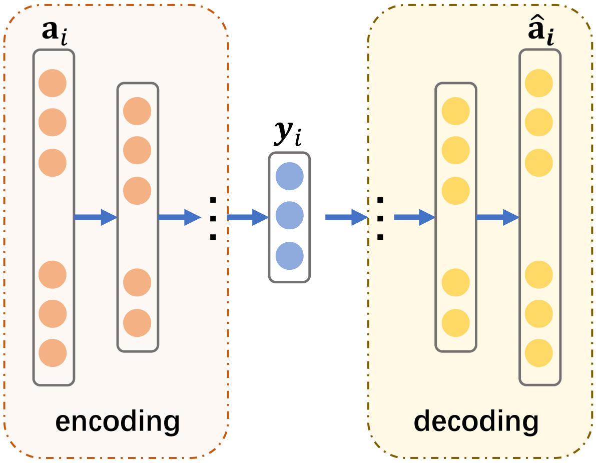

FIGURE 2 Illustration of autoencoder framework.

where D is defined as the sigmoid function of the vertex pair inner product:

$$D_\theta\left(v, v_c\right) = \sigma\left(\mathbf{y}_v^\top \mathbf{y}_{v_c}\right) = \frac{1}{1 + \exp\left(-\mathbf{y}_v^\top \mathbf{y}_{v_c}\right)}, \tag{5}$$

where $\mathbf{y}_v$ and $\mathbf{y}_{v_c}$ are the embedding representations of vertex v and vertex v_c in the middle layer of the discriminator.

Ultimately, the ultimate goal of the D is to maximize the objective function:

$$O_{dis} = O_{sup} - O_{unsup}. \tag{6}$$

The O_{sup} considers locality-preserving in the way of adversarial learning, and O_{unsup} preserves the global structure with unsupervised learning. The objective function combines them in a unified framework by applying the adversarial learning strategy to the embedding mechanism of deep structure autoencoder.

3.2.2 | Generator

As the opponent, G selects the vertexes that are most likely to be connected to V_c based on the current parameter ϕ, and tries to make the generated samples get a higher score in the discriminator, i.e., minimizing the following objective function.

$$O_{\text{gen}} = \log\left(1 - D_\theta\left(v, v_c\right)\right). \tag{7}$$

The generator updates its parameters according to the result returned by the discriminator. In other words, the generator shifts its connectivity distribution to increase the scores of its generated samples, as judged by D. Furthermore, the discriminator captures the global structure by preserving the second-order proximity, so that the locality preservation and the global structure restriction are jointly considered in the adversarial learning process.

3.3 | Discriminator Optimization

To optimize the model of D, the goal is to maximize the objective function O_{dis}, the key step is to calculate the partial derivatives:

$$\begin{aligned} \frac{\partial \mathcal{O}_{\text{dis}}}{\partial \theta_d} &= \frac{\partial \mathcal{O}_{\text{unsup}}}{\partial \theta_d} \\ \frac{\partial \mathcal{O}_{\text{dis}}}{\partial \theta_e} &= \frac{\partial \mathcal{O}_{\text{sup}}}{\partial \theta_e} + \frac{\partial \mathcal{O}_{unsup}}{\partial \theta_e}, \end{aligned} \tag{8}$$

where θ_e and θ_d are the encoder and decoder parameters of D respectively. Since the discrimination result of D is based on the encoded vector, the parameter update of O_{sup} loss only involves θ_e.

It can be seen from Eq. 3 that O_{unsup} has only one variable $\hat{a}$, so we first get the partial derivatives:

$$\begin{aligned}\frac{\partial \mathcal{O}_{\text{unsup}}}{\partial \theta_d} &= \frac{\partial \mathcal{O}_{\text{unsup}}}{\partial \hat{A}} \cdot \frac{\partial \hat{A}}{\partial \theta_d} \\ \frac{\partial \mathcal{O}_{\text{unsup}}}{\partial \hat{A}} &= 2\left(\hat{A} - A\right) \circ T,\end{aligned} \tag{9}$$

Based on back-propagation, we can iteratively calculate $\partial \hat{A} / \partial \theta_d$. In the same way, we can calculate $\partial \mathcal{O}_{\text{unsup}} / \partial \theta_e$.

The next step is to calculate $\partial \mathcal{O}_{\text{sup}} / \partial \theta_e$. $\mathcal{O}_{\text{sup}}$ is essentially the cross-entropy between the discrimination result and ground truth. The calculation process is as follows:

$$\frac{\partial \mathcal{O}_{\text{sup}}}{\partial \theta_e} = \frac{\partial \mathcal{O}_{\text{sup}}}{\partial Y} \cdot \frac{\partial Y}{\partial \theta_e}, \tag{10}$$

Through the cross entropy function ($tf.nn.sigmoid\_cross\_entropy\_with\_logits$) update the middle layer vector representation of the discriminator Y. After that, updating the encoder parameters θ_e in the process of backward propagation.

3.4 | Generator Optimization

As a competitor, the generator hopes to minimize the probability that the discriminator will correctly assign the negative label to the generated sample. The optimization of the generator is essentially adjusting the parameters ϕ to approximate the true connectivity distribution. The key is to calculate the gradient of O_{gen} with respect to ϕ:

$$\begin{aligned}&\nabla_\phi O_{gen} \\ &= \nabla_\phi \sum_{c=1}^{n} E_{v \sim G_\phi(\cdot|v_c)} \left[\log\left(1 - D_\theta\left(v, v_c\right)\right)\right] \\ &= \sum_{c=1}^{n} \sum_{i=1}^{m} \nabla_\phi G_\phi\left(v_i \mid v_c\right) \log\left(1 - D_\theta\left(v, v_c\right)\right) \\ &= \sum_{c=1}^{n} \sum_{i=1}^{m} G_\phi\left(v_i \mid v_c\right) \nabla_\phi \log G_\phi\left(v_i \mid v_c\right) \log\left(1 - D_\theta\left(v, v_c\right)\right) \\ &= \sum_{c=1}^{n} E_{v \sim G_\phi(\cdot|v_c)} \left[\nabla_\phi \log G_\phi\left(v \mid v_c\right) \log\left(1 - D_\theta\left(v, v_c\right)\right)\right],\end{aligned} \tag{11}$$

where m is the number of neighbor nodes of v_c generated by the generator.

In other words, $\nabla_\phi O_{gen}$ is the expected summation of $\nabla_\phi \log G_\phi\left(v \mid v_c\right)$ weighted by $\log\left(1 - D_\theta\left(v, v_c\right)\right)$. We apply gradient descent to ϕ , so the distance between two vertices in a vertex pair with a higher probability of negative samples is increased.

Algorithm 1 DnnGAN framework.

```
Input: $d$: dimension of embedding; $s$: size of generating samples; $p$: size of discriminating samples;
Output:generator $G_\phi\left(v \mid v_c\right)$; discriminator $D_\theta\left(v, v_c\right)$;
Initialize $G_\phi\left(v \mid v_c\right)$ and $D_\theta\left(v, v_c\right)$.
while not converge do
    for G-steps do
        $G_\phi\left(v \mid v_c\right)$ generates $s$ vertices for each $v_c$;
        Update $\phi$ according to Eq. (7) and (11);
    end for
    for D-steps do
        Sample $p$ positive vertices from ground truth and $p$ negative vertices from $G_\phi\left(v \mid v_c\right)$ for each $v_c$;
        Update $\theta_D$ according to Eq. (6), (9) and (10);
    end for
end while
return $G_\phi\left(v \mid v_c\right)$ and $D_\theta\left(v, v_c\right)$;
```

Algorithm 2 Discriminator's autoencoder framework.

```
Input:the graph $\mathcal{G} = (V, E)$ with adjacency matrix $A$;
Output:graph representation $Y$ and updated parameters: $\theta$;
Initialized parameters $\theta$;
while true do
    Based on $\theta$, obtain the reconstruction $\hat{A}$ and the middle layer representation $Y$ of the input vertices.
    Based on Eq. (3), (8) and (9), update the discriminator parameters $\theta$;
end while
```

TABLE 1 Statistics of datasets.

Dataset	#nodes	#edges	#label
Wiki	2363	5278	17
Cora	2708	5278	7
Citeseer	3264	4551	6
Pubmed	19717	44338	3

3.5 | Complexity Analysis

As shown in Alg. 2, the training complexity of the discriminator to reduce the reconstruction loss is $O(ncdI)$, where n is the number of vertices, c is the average degree of the graph, d is the dimensionality of the graph embedding, and I is the number of iterative updates. While c can usually be regarded as a constant in the real world. c, d, and I are independent with n. Therefore, the training complexity of the discriminator has a linear relationship with the number of vertices in the graph.

In the adversarial learning process, we use graph softmax[20] as our sampling strategy, and the sampling complexity is $\log n$. So the complexity of line 4 and line 5 in Alg. 1 are both $O(sn \cdot \log n \cdot d)$, the complexity of line 8 is $O(tn \cdot \log n \cdot d)$, and the complexity of line 9 is $O(ncdI) + O(tnd)$. Usually, we treat s, t, and d as constants, so the complexity of each iteration of the model is $O(n \log n)$.

4 | EXPERIMENTS

4.1 | Experiments Setup

We use the following four datasets for experiments, the detailed statistics of the datasets are shown in Table 1, where $|V|$, $|E|$, $|C|$ are the number of vertices, edges, and vertex categories, respectively.

1) **Wiki**[39] is a Wikipedia hyperlink graph, where the vertices are web pages and edges are hyperlinks.
2) **Citeseer, Cora**[40] are paper citation graphs. Citeseer consists of 3264 papers with 6 categories, and Cora consists of 2708 papers with 7 categories. The labels represent the research topics of the papers.
3) **Pubmed**[41] is a relatively large-scale citation graph, consists of 19717 papers with 3 categories.

We use the following five models as baselines to compare their performance in various tasks:

1) DeepWalk[8] transforms the graph into a series of vertices through a truncated random walk strategy, and learns the embedding of vertices in combination with Skip-gram[42].
2) Node2vec[43] is a variant of DeepWalk, which designs a biased random walk process to balance local attributes and global attributes.
3) Grarep[9] uses SVD to train the model and constructs different k-step probability conversion matrices to preserve high-order vertex proximity.

TABLE 2 Discriminator Structures.

Dataset	#nodes in each layer
Wiki	2363-500-128-500-2363
Cora	2708-500-128-500-2708
Citeseer	3264-500-128-500-3264
Pubmed	19717-5000-1000-128-1000-5000-19717

TABLE 3 AUC and F-Score on all datasets in Link Prediction.

Model	Wiki		Cora		Citeseer		Pubmed	
	AUC	F-Score	AUC	F-Score	AUC	F-Score	AUC	F-Score
DeepWalk	0.89	0.81	0.72	0.59	0.71	0.60	0.79	0.37
Node2vec	0.93	0.86	0.57	0.45	0.59	0.42	0.86	0.36
Grarep	0.97	0.91	0.87	0.69	0.74	0.63	0.87	0.76
SDNE	0.77	0.68	0.73	0.63	0.66	0.59	0.81	0.72
GraphGAN	0.95	0.89	0.93	0.87	0.94	0.87	0.86	0.78
DnnGAN	**0.99**	**0.96**	**0.96**	**0.91**	**0.95**	**0.89**	**0.95**	**0.89**

4) SDNE[15] applies unsupervised deep network structure to network embedding, trying to preserve first-order and second-order proximity.

5) GraphGAN[20] is a unified framework that designs models via adversarial training in a minimax game. The graph softmax solves the limitations of the traditional softmax function.

Usually the embedding dimension is set to a power of 2. To ensure fairness, we set it to 128 uniformly. The parameter settings of the baselines follow the suggestions of their authors. For more objective experimental results, we introduce the EvalNE[44] toolbox in the experiment. It comprehensively and systematically evaluates all algorithms in a unified framework.

In our approach DnnGAN, the hyper-parameters of α, s and p are tuned by using grid search on the validation set. G and D are initialized weights from the uniform distribution in $[-0.1, 0.1]$. The structure of D varies with the size of the dataset, and the size of each layer is shown in Table 2. If the model becomes more complex, the performance is almost unchanged. We use the tensorflow deep learning tools to learn the model with a starting learning rate of 0.001.

4.2 | Experiment Results

4.2.1 | Link Prediction

In link prediction, the goal is to predict whether there is an edge between two vertices. So this task can evaluate the edge predictability of different methods. Specifically, we randomly select 60% of the edges of the original graph to train all methods. The test set contains the remaining 40%of the edges as positive samples and the same number of randomly selected unconnected vertex pairs as negative samples. After training, we can get the low-dimensional representation for each vertex, and then use logistic regression to make predictions on the edges of the test set. Considering the incompleteness of the graph, we restrict the shortest path length of negative samples to be greater than 2. We use 10-fold cross-validation to evaluate the performance of all methods, and use AUC (area under the ROC curve), F-score, and Precision@K as evaluation metrics. We report the performances of AUC and F-score on all datasets in Table 3, the learning curves of G and D in Figure 3, and the change of F-score on Cora by changing the embeddings dimension d in Figure 4. Moreover, the value of Precision on top-K link prediction result is an important indicator for evaluating model stability. We use Wiki as the dataset to report the result in Figure 5.

We have the following observations: **(i)** As shown in Table 4, DnnGAN outperforms all baselines in link prediction on four datasets. It improves AUC by 9% to 95% and F-score by 11% to 89% on Pubmed. This proves that the application of the adversarial learning strategy to the embedding mechanism is an improvement. **(ii)** The training curve in Figure 3 shows that the

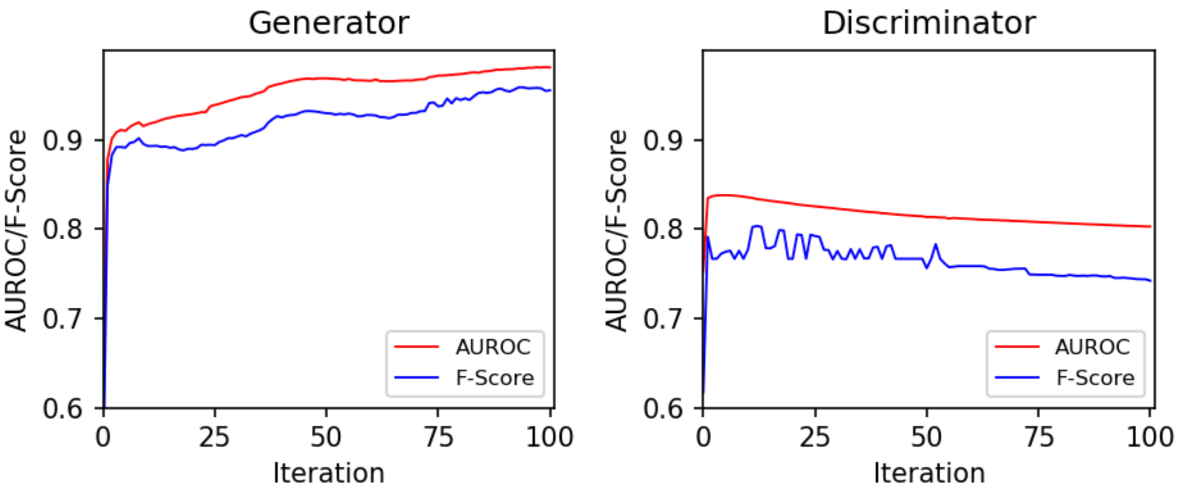

FIGURE 3 Learning curves of the generator and the discriminator of DnnGAN on Wiki in Link Prediction.

performance of the *G* increases with the number of iterations, and the performance of the *D* decreases slightly after it stabilizes. It means that the *D* can continuously provide effective supervision information for the *G*. **(iii)** As shown in Figure 4, most models achieve their best performance when the embedding dimension is about 128, and DnnGAN keeps the lead. This proves the effectiveness of DnnGAN. **(iv)**The results in Figure 5 show that as the value of k increases, the performance of our method is always better than other methods. It shows that the representation learned by our method has better predictive power on the formation of new links.

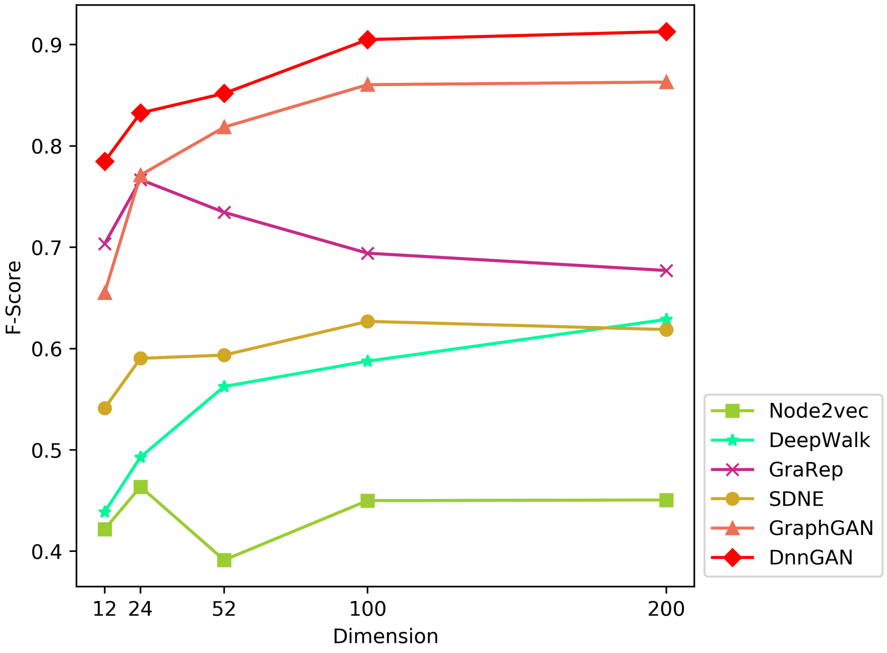

FIGURE 4 F-Score on Cora in different dimensions

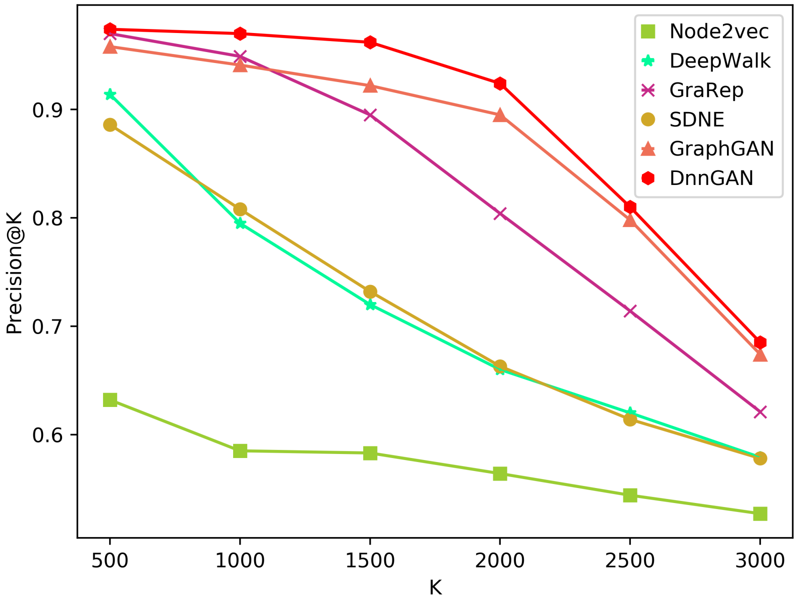

FIGURE 5 Precision@K on Wiki in Link Prediction

TABLE 4 Micro-F1 on all datasets in Node Classification.

Model	Wiki	Cora	Citeseer	Pubmed
DeepWalk	0.68	0.87	0.74	0.81
Node2vec	0.69	0.85	0.73	0.81
Grarep	0.69	0.83	0.70	0.81
SDNE	0.57	0.62	0.44	0.52
GraphGAN	0.52	0.56	0.33	0.52
DnnGAN	**0.70**	**0.89**	**0.82**	0.80

4.2.2 | Nodes Classification

In nodes classification, we aim to classify each vertex into one of multiple labels. Specifically, we use logistic regression as a classifier, take the embedding results of different models as a training set, and classify these vertices after training. The performance of vertex classification reveals whether the graph representation learning method can discover and preserve the proximity. We split the training and test sets at a ratio of 9:1, and report the Micro-F1 of all models on all data sets in Table 4. In addition, we observe how the training set reduces model performance changes. The results Micro-F1 are shown in Figure 6.

We have the following observations: **(i)** As shown in Table 4, DnnGAN performed best in three of the four data sets. In particular, compared with GraphGAN, there is an obvious improvement that Micro-F1 has increased by 0.18, 0.33, 0.49, 0.28 respectively on all data sets. This indicates that DnnGAN outperforms GraphGAN on node classification. **(ii)** In Figure 6, our method is always at the top with minimal change. In the case of a small percentage of the training set, DnnGAN can still main a decent performance, proving the strong generalization ability of DnnGAN.

4.2.3 | Visualization

Another important application of graph representation learning is visualization. We choose Pubmed as the dataset for the task of visualization. We obtain the vector representations of Pubmed under different graph representation learning models after

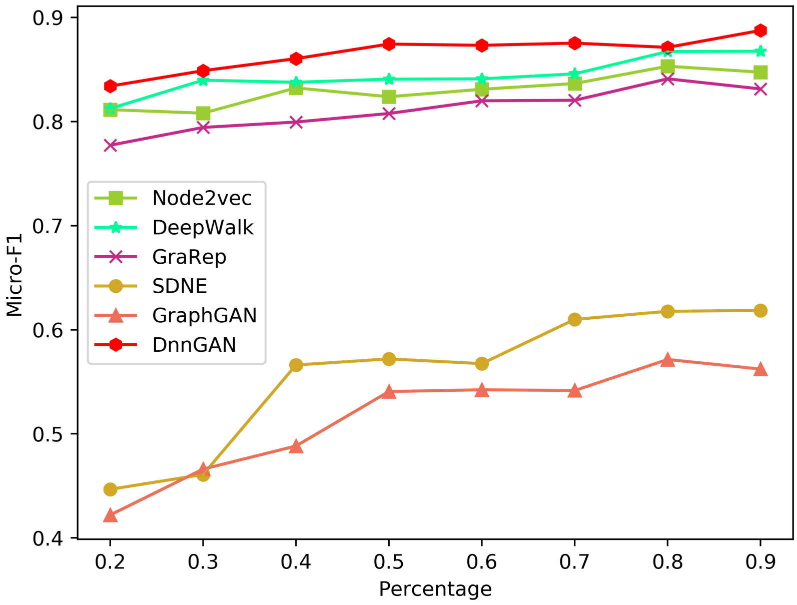

FIGURE 6 Micro-F1 on Cora

training, and use t-SNE[45] to map the representation vector to a two-dimensional space. The vertices of different categories are marked with different colors. Therefore, the ideal visualization result should make the vertices of the same category closer together.

The visualized result is shown in Figure 7. We can see that: **(i)** For the baselines, effective clustering only occurs in small and scattered areas, which is still chaotic and indistinguishable when viewed as a whole; **(ii)** DnnGAN can distinguish different types of vertices more clearly. It is evidence that DnnGAN captures a more desirable global graph structure.

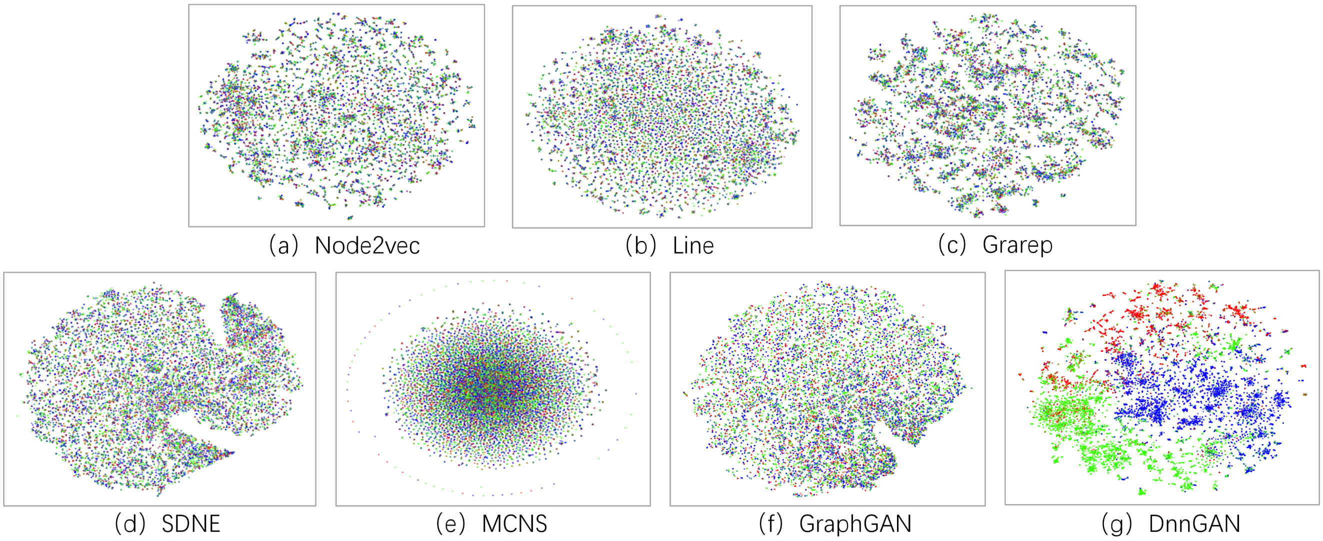
(a) Node2vec
(b) Line
(c) Grarep
(d) SDNE
(e) MCNS
(f) GraphGAN
(g) DnnGAN

FIGURE 7 Visualization of Pubmed. Each dot represents a publication. The color of the dot indicates the category of the publication.

4.2.4 | Impact of Embedding Dimension

We test the impact of dimensional changes on the link prediction performance of all methods on Cora. Figure 4 shows the results of different embedding sizes in the link prediction experiment. It is not difficult to see that a higher embedding size can improve the experimental results to a certain extent, because a higher embedding size means more vertex attribute information. But when the embedding size increases above 100, this parameter has little effect on the experimental results. Overall, choosing the correct size can make the model get better results. The ideal size range of the current model is between 100 and 150.

5 | CONCLUSIONS

To capture the highly non-linear characteristics of the graph and better utilize the essential advantage of GAN, we propose a new approach of GAN-based deep neural network for graph representation learning. Our approach adopts a deep autoencoder as discriminator, which captures the global structural information, and it formulates an adversarial learning strategy on the representation mechanism (autoencoder reconstruction) rather than on embedding representations. In addition, the negative sample generator is introduced into the adversarial learning system as a competitor. The adversarial learning system approximates the true connectivity distribution, and the local structure is preserved. The method proposed is evaluated on real data sets of different scales and fields. Experiments show that DnnGAN has superior performance in different application scenarios.

6 | ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant, No. 61762036; Headmaster Fund of Minnan Normal University under Grant, No. KJ19009; The Natural Science Foundation of Fujian Province of China, No. 2021J011007 and No. 2021J011008.

6.1 | Author contributions

Ming Zhao: Formal analysis; methodology; writing-original draft. **Yinglong Zhang**: Funding acquisition; writing-review and editing.

6.2 | Conflict of interest

The authors declare no potential conflict of interests.

References

1. Yu X, Ren X, Sun Y, et al. Personalized entity recommendation: A heterogeneous information network approach. In: ; 2014: 283–292.
2. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. Line: Large-scale information network embedding. In: ; 2015: 1067–1077.
3. Liu L, Cheung WK, Li X, Liao L. Aligning Users across Social Networks Using Network Embedding. In: ; 2016: 1774–1780.
4. Gao S, Denoyer L, Gallinari P. Temporal link prediction by integrating content and structure information. In: ; 2011: 1169–1174.
5. Tang J, Aggarwal C, Liu H. Node classification in signed social networks. In: ; 2016: 54–62.
6. Maaten v. dL, Hinton G. Visualizing data using t-SNE. *Journal of machine learning research* 2008; 9: 2579–2605.
7. Lin Y, Liu Z, Sun M, Liu Y, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In: ; 2015: 2181–2187.

8. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: ; 2014: 701–710.
9. Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information. In: ; 2015: 891–900.
10. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. *science* 2000; 290: 2323–2326.
11. Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In: ; 2001: 585–591.
12. Chen M, Yang Q, Tang X, others . Directed Graph Embedding. In: ; 2007: 2707–2712.
13. Tang L, Liu H. Relational learning via latent social dimensions. In: ; 2009: 817–826.
14. Yang C, Liu Z, Zhao D, Sun M, Chang EY. Network Representation Learning with Rich Text Information. In: IJCAI'15. ; 2015: 2111–2117.
15. Wang D, Cui P, Zhu W. Structural deep network embedding. In: ; 2016: 1225–1234.
16. Tu K, Cui P, Wang X, Yu PS, Zhu W. Deep recursive network embedding with regular equivalence. In: ; 2018: 2357–2366.
17. Tu K, Cui P, Wang X, Wang F, Zhu W. Structural deep embedding for hyper-networks. In: ; 2018: 426–433.
18. Zhu D, Cui P, Wang D, Zhu W. Deep variational network embedding in wasserstein space. In: ; 2018: 2827–2836.
19. Ma J, Cui P, Zhu W. Depthlgp: learning embeddings of out-of-sample nodes in dynamic networks. In: ; 2018: 370–377.
20. Wang H, Wang J, Wang J, et al. Learning graph representation with generative adversarial nets. *IEEE Transactions on Knowledge and Data Engineering* 2021; 33: 3090-3103.
21. Dai Q, Li Q, Tang J, Wang D. Adversarial Network Embedding. In: ; 2017: 2167–2174.
22. Dai Q, Li Q, Zhang L, Wang D. Ranking Network Embedding via Adversarial Learning. In: ; 2019: 27–39.
23. He D, Zhai L, Li Z, et al. CANE: community-aware network embedding via adversarial training. *Knowledge and Information Systems* 2021; 63(2): 411–438.
24. Gu Y, Sun Y, Li Y, Yang Y. RaRE: Social Rank Regulated Large-Scale Network Embedding. In: WWW '18. ; 2018: 359–368.
25. Ou M, Cui P, Pei J, Zhang Z, Zhu W. Asymmetric Transitivity Preserving Graph Embedding. In: KDD '16. ; 2016: 1105–1114.
26. Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. In: ; 2017: 1–14.
27. Chen H, Yin H, Chen T, Nguyen QVH, Peng WC, Li X. Exploiting Centrality Information with Graph Convolutions for Network Representation Learning. In: ; 2019: 590-601.
28. Hamilton WL, Ying R, Leskovec J. Inductive Representation Learning on Large Graphs. In: NIPS'17. ; 2017: 1025–1035.
29. Feng Y, You H, Zhang Z, Ji R, Gao Y. Hypergraph neural networks. In: . 33. ; 2019: 3558–3565.
30. Tu K, Cui P, Wang X, Wang F, Zhu W. Structural Deep Embedding for Hyper-Networks. In: ; 2018: 426-433.
31. Pan S, Hu R, Long G, Jiang J, Yao L, Zhang C. Adversarially Regularized Graph Autoencoder for Graph Embedding. In: IJCAI'18. ; 2018: 2609–2615.
32. Yu W, Zheng C, Cheng W, et al. Learning Deep Network Representations with Adversarially Regularized Autoencoders. In: ; 2018: 2663–2671.
33. Gao H, Pei J, Huang H. ProGAN: Network Embedding via Proximity Generative Adversarial Network. In: ; 2019: 1308–1316.
34. He D, Zhai L, Li Z, et al. Adversarial Mutual Information Learning for Network Embedding. In: ; 2020: 3321–3327.

35. Wang J, Yu L, Zhang W, et al. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In: ; 2017: 515–524.

36. Yu L, Zhang W, Wang J, Yu Y. SeqGAN: sequence generative adversarial nets with policy gradient. In: ; 2017: 2852–2858.

37. Goodfellow I, PougetAbadie J, Mirza M, et al. Generative adversarial networks. *Communications of the ACM* 2020; 63: 139–144.

38. Berthelot D, Schumm T, Metz L. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *CoRR* 2017; abs/1703.10717: 1-10.

39. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. Collective classification in network data. *AI magazine* 2008; 29(3): 93–93.

40. McCallum AK, Nigam K, Rennie J, Seymore K. Automating the construction of internet portals with machine learning. *Information Retrieval* 2000; 3: 127–163.

41. Yang Z, Cohen WW, Salakhutdinov R. Revisiting Semi-Supervised Learning with Graph Embeddings. In: ICML'16. ; 2016: 40–48.

42. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: ; 2013: 3111–3119.

43. Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks. *ACM* 2016: 855–864.

44. Mara A, Lijffijt J, Bie TD. EvalNE: A Framework for Evaluating Network Embeddings on Link Prediction. *CoRR* 2019; abs/1901.09691: 1–5.

45. Maaten v. dL, Hinton G. Visualizing Data using t-SNE.. *Journal of Machine Learning Research* 2008; 9: 2579-2605.

46. Tang L, Liu H. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 2011; 23: 447–478.

47. Zhang Y, Barzilay R, Jaakkola T. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics* 2017; 5: 515–528.

48. Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ; 2017: 2223–2232.

49. Salakhutdinov R, Hinton G. Semantic hashing. *International Journal of Approximate Reasoning* 2009; 50: 969–978.

50. Makhzani A, Shlens J, Jaitly N, Goodfellow I, Frey B. Adversarial autoencoders. *CoRR* 2015; abs/1511.05644: 1–16.

51. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed Representations of Words and Phrases and their Compositionality. *Advances in neural information processing systems* 2013; 26: 3111–3119.

52. Chae DK, Kang JS, Kim SW, Lee JT. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. In: ; 2018: 137–146.

53. Wang Q, Yin H, Hu Z, Lian D, Wang H, Huang Z. Neural Memory Streaming Recommender Networks with Adversarial Training. In: ; 2018: 2467-2475.

54. Huang H, Yu PS, Wang C. An Introduction to Image Synthesis with Generative Adversarial Nets. *CoRR* 2018; abs/1803.04469: 1-17.

55. Glover J. Modeling documents with Generative Adversarial Networks. *CoRR* 2016; abs/1612.09122: 1-6.

56. Liben-Nowell D, Kleinberg J. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 2007; 58(7): 1019–1031.

AUTHOR BIOGRAPHY

Ming Zhao. Fujian Key Laboratory of Granular Computing and Application, Minnan Normal University, Zhangzhou, China

Ming Zhao was born in Shanxi, China, in 1995. He is a master candidate in Fujian Key Laboratory of Granular Computing, Minnan Normal University, China, from 2019 till now. His research interests include machine learning and information network analysis.

Yinglong Zhang. Scchool of Physics and Information Engineering, Minnan Normal University, Zhangzhou, China

Yinglong Zhang received the Ph.D. degree in Computer Software and Theory from RenMin University, Beijing, China, in 2014. He is currently an Associate Professor with Minnan Normal University, China. His research interests include data mining and information network analysis.

□