# Distribution-induced Bidirectional Generative Adversarial Network for Graph Representation Learning

Shuai Zheng[1,2], Zhenfeng Zhu[1,2,*], Xingxing Zhang[1,2], Zhizhe Liu[1,2], Jian Cheng[3,4], Yao Zhao[1,2]
[1]Institute of Information Science, Beijing Jiaotong University, Beijing, China
[2]Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing, China
[3]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[4]University of Chinese Academy of Sciences, Beijing, China
[1,2]{zs1997,zhfzhu,zhangxing,yzhao}@bjtu.edu.cn, [3,4]jcheng@nlpr.ia.ac.cn

## Abstract

*Graph representation learning aims to encode all nodes of a graph into low-dimensional vectors that will serve as input of many compute vision tasks. However, most existing algorithms ignore the existence of inherent data distribution and even noises. This may significantly increase the phenomenon of over-fitting and deteriorate the testing accuracy. In this paper, we propose a Distribution-induced Bidirectional Generative Adversarial Network (named **D-BGAN**) for graph representation learning. Instead of the widely used normal distribution assumption, the prior distribution of latent representation in our DBGAN is estimated in a structure-aware way, which implicitly bridges the graph and feature spaces by prototype learning. Thus discriminative and robust representations are generated for all nodes. Furthermore, to improve their generalization ability while preserving representation ability, the sample-level and distribution-level consistency is well balanced via a bidirectional adversarial learning framework. An extensive group of experiments are then carefully designed and presented, demonstrating that our DBGAN obtains remarkably more favorable trade-off between representation and robustness, and meanwhile is dimension-efficient, over currently available alternatives in various tasks. The source code is released in https://github.com/SsGood/DBGAN.*

## 1. Introduction

A graph is a collection of nodes and edges that can be used to model relationships and processes between data in a variety of scenarios, such as biomedical networks, citation networks, and social networks. Therefore, graph analysis is a necessary step to explore the internal information of these networks. However, due to the complex topology and high data dimension of graph data, most of the current machine learning methods for simple sequences or grids design are not suitable for graph data analysis. As a general approach to these problems, Graph representation learning aims to represent sparse raw features of graph nodes as compact low-dimensional vectors while preserving enough information for subsequent downstream tasks, such as link prediction [6,38], clustering [26,36], and recommendation [30,34]. In recent years, a variety of graph representation learning methods have been proposed, which can be broadly summarized into two categories: proximity-based algorithms and deep learning-based algorithms.

By applying matrix factorization, proximity-based algorithms, such as GraRep [2], HOPE [27], M-NMF [39] attempt to factorize the graph adjacency matrix to obtain the node representation. While for probabilistic models, such as DeepWalk [30], line [33], and node2vec [13], they learn the node representation with local neighborhood connectivities through randomwalk and various order proximities. These methods are all focused on preserving the original neighborhood relationship in a low dimensional space. Recent studies have also shown that probabilistic models and matrix factorization-based algorithms are equivalent and can be implemented by a unified model [31].

Deep learning-based approaches are receiving increasing attention, most of which use the auto-encoder framework to capture the latent representation. SDNE [37] and DNGR [3] use deep auto-encoders to model the positive point-wise mutual information (PPMI) while preserving the structure of the graph. The GAE [17] first merges the GCN [16] as an encoder into the auto-encoder framework to seek the latent representation by reconstructing the adjacency matrix. In addition, MGAE [36], GDN [20], and GALA [29] attempt to preserve node feature in latent representation by building

*Corresponding author.

learnable decoders and encoders on a GAE basis. In fact, most of the above methods are to reconstruct either the adjacency matrix or the node feature, rather than the reconstruction on both together. However, for good low-dimensional latent representations, the topology of the graph and the node feature should be preserved at the same time.

It is worth noting that none of the above methods have explicitly exploited the latent distribution of the graphical data, and thus, the distribution consistency across domains(graph space and feature space) cannot be well preserved, which leads to poor generalization of the representation and sensitivity to noise. Due to the strong ability of the generative adversarial network(GAN) [12] for distribution fitting, some works have introduced adversarial learning into the field of graph representation learning to improve the performance of the learned latent representation. In GraphGAN [38] and ProGAN [10], the generated fake node pairs and node triplets compete with the real data to enhance the robustness of latent representation. These methods ignore the global structure and node feature, and fail to preserve the distributed consistency, resulting in the insufficiency in generalization ability. Besides, the normal distribution $\mathbb{N}(0,1)$ has been generally pre-assumed in AIDW [5] and ARGA [28] to guide the generation of latent representations. However, in most cases, it is not suitable to model the latent distribution of graph data by $\mathbb{N}(0,1)$, and an inaccurate prior distribution can cause the model to be over-smoothing or even misleading.

Motivated by the observations mentioned above, we propose a distribution-induced bidirectional GAN for unsupervised graph representation learning, named as DBGAN. To enhance the generalization ability of the representation, different from unidirectional mapping of data to representation in ARGA [28] and AIDW [5], we not only apply adversarial learning to the encoder but also construct a generator for modeling the mapping of latent representation to graph data, establishing a bidirectional mapping between the two spaces, thus, the distribution consistency and sample consistency of the node representations are preserved in the latent space. Furthermore, to preserve the structural consistency of graph data, we perform prior distribution estimation in latent space using the learned cross-domain prototypes. This will facilitate the robustness of node representations and alleviate the over-smoothing problem caused by normal distribution assumption like in ARGA [28]. We evaluate the effectiveness of latent representations learned by GBGAN on both link prediction and node clustering tasks. The contributions are highlighted in the following aspects:

- We propose a Distribution-induced Bidirectional Generative Adversarial Network (DBGAN), for graph representation learning with a dimension-efficient property. To the best of our knowledge, it is the first work to consider prior distribution estimation in adversarial learning.

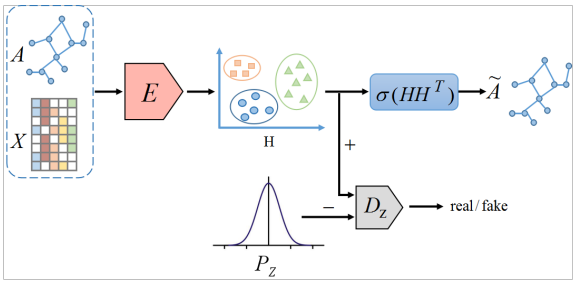



Figure 1. architecture of ARGA [28] and AIDW [5]. $A$ and $\tilde{A}$ represent the adjacency matrix and reconstructed adjacency matrix, respectively. $X$ denotes the node feature matrix. " $+$ " denotes the real samples and " $-$ " denotes the fake samples.

- To improve generalization ability while preserving representation ability, the sample-level and distribution-level consistency are well balanced via bidirectional adversarial learning.

- Unlike the widely used normal distribution assumption, we innovatively estimate structure-aware prior distribution of latent representation by bridging the graph and feature spaces with learned prototypes, thus generating robust and discriminative representations.

- Significant improvements over currently available alternatives demonstrate that our DBGAN creates a new baseline in the area of graph representation learning.

## 2. GANs for Representation Learning

GAN [12] has demonstrated its strong distribution fitting ability in various fields since it was first proposed by Goodfellow. AAE [24], BiGAN [7], and ALI [9] have already explored the application of adversarial learning in the field of image representation. And most recently, BigBiGAN [8] based on BiGAN has achieved amazing performance in image representation learning.

The success of the above works shows that the distribution fitting ability of GAN can be used not only to generate data but also to understand data. Thus, GAN has been introduced into the field of graph representation learning in various forms [5, 28, 38]. **From the perspective of sample generation**, Ding et al. [6] use the generator to generate fake samples in low-density areas between subgraphs to enable the classifier to take into account the density characteristics of the graph data. To preserve the structure information, ProGAN [10] applies the generator to generate triplets of nodes to discover the proximity in the original space and preserving it in the low dimensional space. **From the perspective of latent distribution fitting**, NetRA [43] uses adversarial learning to keep the latent representations

away from the noise representation generated by the normal distribution to improve the anti-jamming capability of the representations. As shown in Fig.1, ARGA [28] and AIDW [5] take a similar approach and introduce adversarial learning into [17] and [30] respectively, to improve the generalization ability of the representations.

Although the above methods have achieved good performance, the disadvantages of them are also obvious. [6, 10, 38] only consider the local structure information, ignores the global structure and distribution consistency, resulting in noise sensitivity, which makes the representation less robust. Besides, in [5, 28], the node feature hasn't been utilized, and the pre-assumed normal distribution won't ideally conform to the complex graph data in reality, which makes the model tend to be over-smoothing and further reduces the generalization of the learned latent representation.

## 3. Methodology

In this section, we first give the problem definition of graph representation learning, then present bidirectional adversarial learning in DBGAN, and finally introduce a prior distribution estimation method for latent representation by prototype learning.

### 3.1. Problem Definition

An undirected graph is given as $\mathcal{G} = (V, \mathcal{E})$, where $V = \{v_1, \cdots, v_n\}$ consists of a set of nodes with $|V| = n$, and $\mathcal{E}$ is a sets of edges with $e_{ij} \in \mathcal{E}$. $X = \{x_1, \cdots, x_n\} \in \mathbb{R}^{d \times n}$ denotes the node feature matrix of a graph, where $x_i$ represents the raw feature of node $v_i$. The graph structure can be represented by the adjacency matrix $A$ with $A_{ij}$ = 1 if $e_{ij} \in \mathcal{E}$, otherwise $A_{ij}$= 0. The degree matrix is represented by diagonal matrix $D$ with $D_{ii} = \sum_j A_{ij}$, and $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized adjacency matrix. In the following sections, we denote $A$ as the normalized adjacency matrix.

For a given graph $\mathcal{G}$, graph representation learning aims to map nodes $v_i \in V$ to latent representation $h_i \in H$ where $H = \{h_1, \cdots, h_n\} \in \mathbb{R}^{q \times n}$ denotes the latent representation matrix. In particular, both the structure of $A$ and the node content from $X$ are expected to be well preserved in $H$ space.

### 3.2. Overall Framework

The overall framework of DBGAN is shown in Fig.2. In the encoding phase, the encoder $E$ accepts $A$ and $X$ as inputs and outputs a latent representation matrix $H$. After that, $E$ and the data $Z$ sampled from the prior distribution $P_{z|(X,A)}$ are input into the discriminator $D_z$ for adversarial training, where $z_i \in Z$ and $h_i \in H$ are positive and negative samples, respectively. Meanwhile, the generator $G$ accepts $Z$ and $A$ as input and outputs the fake feature matrix $X'$ of the graph, after which $X'$ as negative samples and $X$ as positive samples are sent to the discriminator $D_x$ for adversarial training. In the reconstruction phase, $H$ is fed to $G$, and then outputs the rebuilt $\tilde{X}$. In addition, $H$ is reconstructed into $\tilde{A}$ through the reconstruction process $\tilde{A} = sigmoid(HH^T)$. In this work, we use GCN [16] as encoder $E$ and generator $G$, and MLP for both discriminators $D_z$ and $D_x$.

### 3.3. Bidirectional Adversarial Learning

Different from adversarial learning as in AIDW [5] and ARGA [28], we propose a bidirectional adversarial learning algorithm that establishes a mutual mapping between graph data and latent representation. It is capable of balancing the consistency between distribution-level and sample-level, thus leading to a significant improvement of generalization ability in latent representation space.

The bidirectional adversarial learning is mainly implemented in two streams. One is composed of $E$ and $D_z$ to model the mapping from graph data to representation, and the other is composed of $G$ and $D_x$ for the reverse mapping. Completely different from the bidirectional adversarial learning in [7], our DBGAN makes full use of the prior distribution in latent space, which acts as not only the target of output for the encoder $E$ but also the source of input for the generator $G$. The superiority of our bidirectional adversarial learning method can be claimed in three aspects: **(i)** bidirectional mapping is more beneficial to exploiting the inherent graph structure than unidirectional mapping. It facilitates the trade-off of distribution-level consistency and sample-level consistency, resulting in more generalized representations; **(ii)** the application of adversarial learning in our DBGAN can address the over-fitting problem well, which to some extent improves the robustness of representation; **(iii)** if the capacity is allowed to be sufficient for encoder and decoder, the auto-encoder may degrade into a copying task instead of extracting more useful information about the data distribution [11]. However, the capacity has no effect on DBGAN, since $G$ and $E$ will not be optimized synchronously with the same batch of data, thus enforcing the reconstruction constraints on latent representation.

**Adversarial loss.** Adversarial loss is used to minimize the distance between two distributions. Here we use the Wasserstein distance in [1] to measure the difference between the graph data distribution $\mathbb{P}_r(x)$ and prior distribution of latent representation $\mathbb{P}_z(x)$, and it can be defined as

$$W\left[\mathbb{P}_z, \mathbb{P}_r\right] = \max_{f, \|f\|_L \leq 1} \mathbb{E}_{z \sim \mathbb{P}_z}\left[f\left(z\right)\right] - \mathbb{E}_{x \sim \mathbb{P}_r}\left[f\left(E(x)\right)\right] \tag{1}$$

where $f$ denotes the discriminant function, and $\|f\|_L \leq 1$ represents a condition that the discriminant function needs to satisfy the Lipschitz constraint with Lipschitz constant 1. Here the gradient penalty term proposed in [14] is used

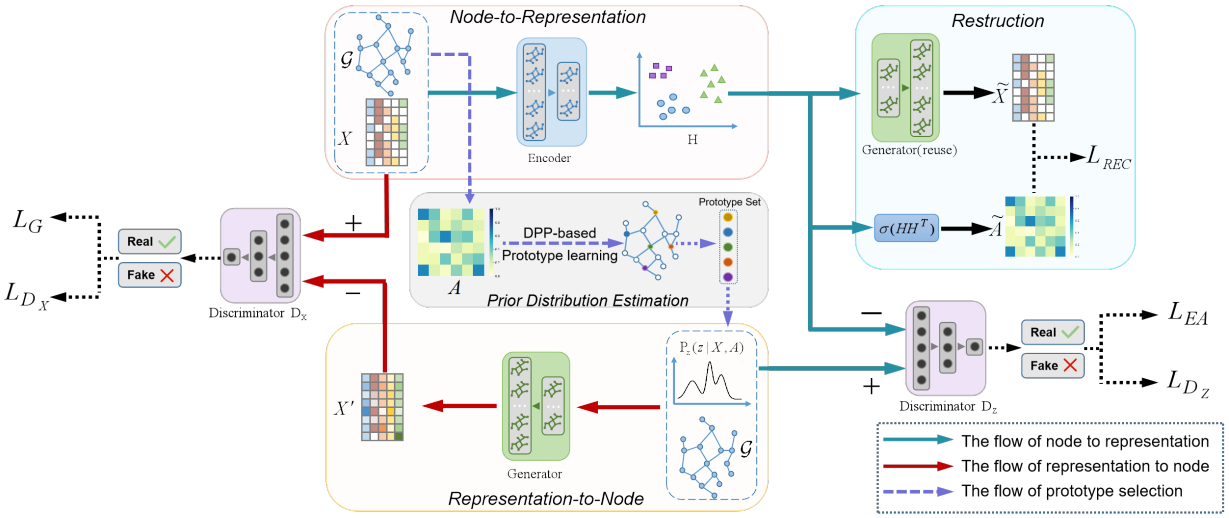



Figure 2. Architecture overview of our DBGAN. $A$ and $\tilde{A}$ represent the adjacency and reconstructed adjacency matrix, respectively. $X$, $X'$, and $\tilde{X}$ denote the node raw feature, the generated feature, and the reconstructed feature, respectively. $L_{REC}$ denotes reconstruction loss, $L_G, L_{EA}, L_{D_X}$, and $L_{D_Z}$ denote the adversarial loss for $G, E, D_X$, and $D_Z$, respectively. And $\mathbb{P}_z(z|X,A)$ denotes the estimated prior distribution, " $+$ " and " $-$ " represent the real and fake samples, respectively.

to implement the Lipschitz constraint and the discriminant function is learned by the discriminator $D_z$. Hence, Eq.(1) can be taken as the objective of $D_z$, while the objective of $E$ is the opposite. According to Eq.(1), we can define the adversarial losses of $D_z$ and $E$ as follows

$$\begin{aligned}\mathcal{L}_{D_z}(z,x) = &-\mathbb{E}_{z\sim\mathbb{P}_z}[D_z(z)] + \mathbb{E}_{x\sim\mathbb{P}_r}[D_z(E(x))] \\ &+\lambda\mathbb{E}_{\hat{z}\sim\mathbb{P}_{\hat{z}}}[\|\bigtriangledown_{\hat{z}} D_z(\hat{z}) - 1\|] \quad (2)\\ \mathcal{L}_{EA}(z,x) = &\mathbb{E}_{z\sim\mathbb{P}_z}[D_z(z)] - \mathbb{E}_{x\sim\mathbb{P}_r}[D_z(E(x))] \quad (3)\end{aligned}$$

where $\hat{z}$ denotes random interpolation of $E(x)$ and $z$ sampled from $\mathbb{P}_z$. When $E$ is updated, $D_z$ will not change. Thus, $\mathbb{E}_{z\sim\mathbb{P}_z}[D_z(z)]$ in Eq.(3) will not provide gradients for $E$, and then Eq.(3) can be simplified as

$$\mathcal{L}_{EA}(x) = -\mathbb{E}_{x\sim\mathbb{P}_r}[D_z(E(x))] \quad (4)$$

Likewise, by switching the roles of $\mathbb{P}_z$ and $\mathbb{P}_x$, we can get the adversarial losses of $G$ and $D_x$ as follows

$$\begin{aligned}\mathcal{L}_{D_x}(x,z) = &-\mathbb{E}_{x\sim\mathbb{P}_r}[D_x(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[D_x(G(z))] \\ &+\lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[\|\bigtriangledown_{\hat{x}} D_x(\hat{x}) - 1\|] \quad (5)\\ \mathcal{L}_G(z) = &-\mathbb{E}_{z\sim\mathbb{P}_z}[D_x(G(z))] \quad (6)\end{aligned}$$

where $\hat{x}$ denotes random interpolation of $G(z)$ and $x$ sampled from $\mathbb{P}_x$.

**Reconstruction loss.** In addition to the adversarial loss that guarantees the distribution-level consistency between the graph space and raw feature space, the reconstruction loss $\mathcal{L}_{REC}(x)$ is also enforced for sample-level consistency. This is essential to further improve the representation ability in latent representation space, by both node feature reconstruction and adjacency matrix reconstruction.

We follow the settings in [17] to get the reconstructed adjacency matrix $\tilde{A}$ from the latent representation, and here $\tilde{A}$ should be similar to real adjacency matrix $A$. Besides, by the mapping established by $G$ of the latent representations to the graph data, we can get the reconstructed feature matrix $X' = G(E(X))$. The reconstruction loss can be defined as follows

$$\mathcal{L}_{REC}(x) = \mathbb{E}_{x\sim\mathbb{P}_r}[d(X,X')] + \mathbb{E}_{x\sim\mathbb{P}_r}[d(A,\tilde{A})] \quad (7)$$

where $X' = G(E(X))$, $A' = sigmoid(E(X)\cdot E(X)^T)$, and $d(x,y) = x\log y + (1-x)\log(1-y)$. Therefore, the overall loss of the encoder $E$ can be written as

$$\mathcal{L}_E(x) = \mathcal{L}_{EA}(x) + \alpha\mathcal{L}_{REC}(x) \quad (8)$$

It is worth noting that the effectiveness of our DBGAN can be claimed by Theorem 1.

**Theorem 1.** *Assuming $W[\mathbb{P}_z, \mathbb{P}_r]$ and $W[\mathbb{P}_r, \mathbb{P}_z]$ converge, i.e., $H = E(X) \sim \mathbb{P}_z$, and $G(Z) \sim \mathbb{P}_r$, it can be inferred that $X' = G(E(X)) \sim \mathbb{P}_r$. Thus, $X'$ and $X$ will obey an identical distribution $\mathbb{P}_r$, i.e., $X' \sim \mathbb{P}_r$ and $X \sim \mathbb{P}_r$. Then, $X \approx G(E(X))$ can be obtained as the reconstruction error converges.*

### 3.4. Prior distribution estimation for latent representation

For the methods based on prior distribution assumptions [5, 28], the prior distribution $\mathbb{P}_z$ is critical to their performances. For example, for graph data with multiple categories, it is not reasonable to use normal distribution $\mathbb{N}(0,1)$

as $\mathbb{P}_z$ to represent the graph. Besides, by bidirectional adversarial learning, an appropriate $\mathbb{P}_z$ can improve the robustness and discriminability of the representation. Since we have no more priors except for the given $A$ and $X$, an intuitive approach is to estimate $\mathbb{P}_z(z|X)$ that approximates to $\mathbb{P}_z(z)$ by a non-parametric estimation method such as Kernel Density Estimation (KDE). In addition, we use PCA to reduce the dimension of $X$ to get $X_p = \{x_i\}_{i=1,\cdots,n}$, and then we can get $\mathbb{P}_z(z|X)$ as follows

$$\mathbb{P}_z(z|X) = \frac{1}{n}\sum_{i=1}^{n} K_b(z - x_i) = \frac{1}{nb}\sum_{i=1}^{n} K(\frac{z - x_i}{b}) \quad (9)$$

where $K(\cdot)$ is a kernel function, $b$ denotes the bandwidth, and $K_b(\cdot)$ is the scaled kernel function.

However, there are some problems with this intuitive approach. First, the explicit structural information embedded in $A$ is completely ignored; second, the learned model is susceptible to the noisy $X$, thus reducing the robustness of representation. Therefore, we can approximate $\mathbb{P}_z(z)$ using $\mathbb{P}_z(z|X, A)$ instead of $\mathbb{P}_z(z|X)$.

**DPP-based prototype learning.** For heterogeneous $A$ and $X$, it is not trivial to obtain $\mathbb{P}_z(z|X, A)$ directly. Considering that $A$ and $X$ are structurally consistent though they are in different domains, we can utilize the cross-domain prototypes to bridge the raw feature domain and the graph domain. Thus $\mathbb{P}_z(z|X, A)$ can be replaced with $\mathbb{P}_z(z|X_{S_p}, A_{S_p})$, where $S_p$ denotes the index set for prototypes.

For prototype learning, the Determinant Point Process (DPP) [18] is adopted to select a diversified prototype subset. Specifically, the adjacency matrix $A$ is considered as the measure matrix. Given a subset $V_S \subseteq V$, whose items are indexed by $S \subseteq \mathcal{N} = \{1, \cdots, n\}$, then the sampling probability of $S$ based on the measure matrix $A$ can be defined as follows

$$P_A(S) = \frac{\det(A_S)}{\det(A + I)} \quad (10)$$

where $I$ denotes the identity matrix, $A_S \equiv [A_{ij}]_{i,j\in S}$, and $\det(\cdot)$ denotes the determinant of a matrix. Obviously, sampling probability defined here is normalized because of

$$\sum_{S\subseteq N} \det(A_S) = \det(A + I) \quad (11)$$

According to Eq.(10), a probability will be assigned to any subset of $\mathcal{N}$, which will result in a large search range for the prototype index subset. Hence, we have limited the subset size to $|S| = m$. When the size of subset $S$ is fixed to $m$, we can define the sampling probability as follows

$$P_A^k(S) = \frac{\det(A_S)}{\sum_{|S'|=k} \det(A_{S'})} \quad (12)$$

Table 1. Statistics of the used datasets.

| Dataset | #Nodes | #Edges | #Classes | #Features |
|---|---|---|---|---|
| Cora | 2708 | 5429 | 7 | 1433 |
| Citeseer | 3327 | 4732 | 6 | 3703 |
| Pubmed | 19717 | 44338 | 3 | 500 |

Similarly, according to Eq.(11), $P_A^k(S)$ is also normalized.

We explain the definition of importance probability from the geometric explanation of the matrix determinant. Considering $A_{ij}$ is computed from $\varphi(v_i)$ and $\varphi(v_j)$, where $\varphi(\cdot)$ is a nonlinear mapping function, then $\det(A)$ can be interpreted as the volume of the geometry spanned by the nodes $v_i \in V$ [18]. Therefore, the prototypes $S_p$ measured by $P_A^k(S_p)$ can better sketch the consistent distribution of $A$ and $X$.

**Structure-aware prior distribution estimation.** According to the prototype index set $S_p$ with $|S_p| = m$, a node feature matrix $X_p$ can be sampled from $X$. Then, we use PCA to reduce the dimension of $X_p$ to get $H_p$. Assuming $h_i \in H_p$ is i.i.d., $\mathbb{P}_z(z|X_{S_p}, A_{S_p})$ can be defined by

$$\mathbb{P}_z(z|X_{S_p}, A_{S_p}) = \frac{1}{m}\sum_{i=1}^{m} K_b(z - h_i) = \frac{1}{mb}\sum_{i=1}^{m} K(\frac{z - h_i}{b}) \quad (13)$$

In summary, with the flow in Eq.14, we obtain the approximation of $P_z$, i.e., $\mathbb{P}_z(z|X_{S_p}, A_{S_p})$.

$$\mathbb{P}_z(z) \to \mathbb{P}_z(z|X) \to \mathbb{P}_z(z|X, A) \to \mathbb{P}_z(z|X_{S_p}, A_{S_p}) \quad (14)$$

## 4. Experimental Results and Analysis

We first detail our experimental protocol, and then present comparison results of DBGAN with the state of the art for graph representation learning.

### 4.1. Evaluation Setup and Metrics

**Datasets.** We select three widely used graph datasets, Cora [22], Citeseer [32], and Pubmed [25], to verify the performance of DBGAN in unsupervised representation learning. Each dataset contains a complete node feature matrix $X$ and an adjacency matrix $A$. Details of three dataset statistics are in Table 1.

**Protocols and evaluation metrics.** The tasks of link prediction and node clustering are employed to evaluate the discrimination and generalization of learned node representation. In particular, for link prediction, we divided each dataset into a training set, a test set, and a validation set, with a ratio of 85:5:10. To avoid the influence of randomness, we average the results over 20 times of execution with different training set selections as in [17]. Then the mean scores and standard errors of Area Under Curve (AUC) and

Table 2. Experimental results of link prediction.

| Methods | Cora | | Citeseer | | Pubmed | |
|---|---|---|---|---|---|---|
| | AUC | AP | AUC | AP | AUC | AP |
| Spectral [26] | 84.6±0.01 | 88.5±0.00 | 80.5±0.01 | 85.0±0.01 | 84.2±0.02 | 87.8±0.01 |
| DeepWalk [30] | 83.1±0.01 | 85.0±0.00 | 80.5±0.01 | 83.6±0.01 | 84.4±0.00 | 84.1±0.00 |
| GAE [17] | 91.0±0.02 | 92.0±0.03 | 89.5±0.04 | 89.9±0.05 | 96.4±0.00 | 96.5±0.00 |
| VGAE [17] | 91.4±0.01 | 92.6±0.01 | 90.8±0.02 | 92.0±0.02 | 94.4±0.02 | 94.7±0.02 |
| ARGA [28] | 92.4±0.003 | 93.2±0.003 | 91.9±0.003 | 93.0±0.003 | 96.8±0.001 | 97.1±0.01 |
| ARVGA [28] | 92.4±0.004 | 92.6±0.004 | 92.4±0.003 | 93.0±0.03 | 96.5±0.001 | 96.8±0.01 |
| DGI [35] | 92.6±0.02 | 93.1±0.01 | 93.3±0.04 | 94.1±0.03 | 95.9±0.002 | 96.3±0.01 |
| GALA [29] | - | - | 94.4±0.009 | 94.8±0.01 | - | - |
| DBGAN | 94.5±0.01 | 95.1±0.05 | 94.5±0.04 | 95.8±0.01 | 96.8±0.01 | 97.3±0.02 |

Average Precision (AP) are reported. While for node clustering, we adopt Kmeans [21] to classify the learned representations into several clusters. As in [29], accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI) are used to measure the performance of clustering. Likewise, we still report the averaged results over 20 times of execution.

**Implementation details.** For the flow from latent representation to node as in Fig. 2, we follow the training strategy in WGAN-GP [14], where a complete iterative process is to train $G$ once after training $D_x$ 5 times. In addition, the discriminator and encoder in our DBGAN are trained synchronously, since encoder $E$ is optimized for both reconstruction loss and adversarial loss. The model uses Adam [15] as the optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and is implemented on the Tensorflow platform.

**Comparison methods.** We choose to compare with a total of fifteen unsupervised graph representation algorithms, especially those that have achieved the state-of-the-art results recently. In particular, such compared algorithms can be divided into three groups.

I. *Using node feature or graph structure only.* In general, Kmeans [21] is considered as a baseline for node clustering. Due to merely usage of topological structure of the graph, Spectral Clustering [26] usually serves as a typical social network representation learning algorithm. Big-Clam [42] is a large-scale community detection algorithm based on non-negative matrix factorization. Additionally, as one of the most representative graph representation learning algorithms, we compare with DeepWalk [30] which encodes graph nodes into latent representations by random walks. A recent algorithm DNGR [3] using auto-encoder to preserve graph structure is also employed.

II. *Using both node feature and graph structure.* Circles [19] is a node clustering algorithm that treats each node as ego and builds an ego graph that preserves the original connection relationship. RTM [4] aims to learn topic distributions of each document from text. RMSC [40] is a multi-view clustering algorithm that can effectively remove noise. TADW [41] integrates node content into Deepwalk, and explains Deepwalk by matrix factorization.

III. *Using node feature and graph structure both with GCN.* GAE [17] is the first GCN-based auto-encoder algorithm for unsupervised graph representation learning. VGAE [17] is a variational version of GAE. ARGA [28] is another variant of GAE that introduces adversarial learning into GAE. Similarly, VARGA [28] is a variational version of ARGA. DGI [35] is a GCN-based method which generates node representations by maximizing local mutual information in the patch representation of the graph. GALA [29] is the latest GCN-based unsupervised framework for graph data, which designs a decoder with Laplacian sharpening as an improvement of GAE.

### 4.2. Evaluation on Link Prediction

For link prediction task, the hyperparameters $\alpha$ and $\lambda$ are set to 1 on all three datasets, the two hidden layers of $G$ are set to 256-neuron and 512-neuron respectively, and the two hidden layers of $D_x$ are set to 512-neuron and 256-neuron respectively. In particular, on Cora dataset, we set the hidden and output layers of $E$ to 32-neuron, and the two hidden layers of $D_z$ are set to 64-neuron and 32-neuron respectively. While on Citeseer and Pubmed datasets, we set the hidden layer and output layer of $E$ to 64-neuron, and the two hidden layers of $D_z$ also to 64-neuron.

The comparative results on link prediction task are shown in Table 2. It can be concluded that, (i) compared to Spectral Clustering and DeepWalk, the spectral convolution-based auto-encoder framework can effectively improve the performance of graph representation in connection prediction tasks. In particular, our DBGAN has achieved the best performance on all three datasets, with

Table 3. Experimental results of node clustering.

| | Methods | Cora | | | Citeseer | | | Pubmed | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | NMI | ARI | ACC | NMI | ARI | ACC | NMI | ARI |
| I | Kmeans [21] | 0.492 | 0.321 | 0.229 | 0.540 | 0.305 | 0.278 | 0.595 | 0.315 | 0.281 |
| | Spectral [26] | 0.367 | 0.126 | 0.031 | 0.238 | 0.055 | 0.010 | 0.528 | 0.097 | 0.062 |
| | Big-Clam [42] | 0.271 | 0.007 | 0.001 | 0.250 | 0.037 | 0.007 | 0.394 | 0.006 | 0.003 |
| | DeepWalk [30] | 0.484 | 0.327 | 0.242 | 0.336 | 0.087 | 0.092 | 0.684 | 0.279 | 0.299 |
| | DNGR [3] | 0.419 | 0.318 | 0.142 | 0.325 | 0.180 | 0.042 | 0.458 | 0.155 | 0.054 |
| II | Circles [19] | 0.606 | 0.404 | 0.362 | 0.571 | 0.300 | 0.293 | - | - | - |
| | RTM [4] | 0.439 | 0.230 | 0.169 | 0.450 | 0.239 | 0.202 | 0.574 | 0.194 | 0.444 |
| | RMSC [40] | 0.406 | 0.255 | 0.089 | 0.295 | 0.138 | 0.048 | 0.576 | 0.255 | 0.222 |
| | TADW [41] | 0.560 | 0.441 | 0.332 | 0.454 | 0.291 | 0.228 | - | - | - |
| III | GAE [17] | 0.596 | 0.429 | 0.347 | 0.408 | 0.176 | 0.124 | 0.672 | 0.277 | 0.279 |
| | VGAE [17] | 0.502 | 0.329 | 0.254 | 0.467 | 0.260 | 0.205 | 0.630 | 0.229 | 0.213 |
| | ARGA [28] | 0.640 | 0.449 | 0.352 | 0.573 | 0.350 | 0.341 | 0.668 | 0.305 | 0.295 |
| | ARVGA [28] | 0.638 | 0.450 | 0.374 | 0.544 | 0.261 | 0.245 | 0.690 | 0.290 | 0.306 |
| | DGI [35] | 0.554 | 0.411 | 0.327 | 0.514 | 0.315 | 0.326 | 0.589 | 0.277 | 0.315 |
| | GALA [29] | 0.745 | 0.576 | 0.531 | 0.693 | 0.441 | 0.446 | 0.693 | 0.327 | 0.321 |
| | DBGAN | 0.748 | 0.560 | 0.540 | 0.670 | 0.407 | 0.414 | 0.694 | 0.324 | 0.327 |

an improvement of $0.1\% \sim 1.9\%$ w.r.t. AUC, and $0.2\% \sim 1.9\%$ w.r.t. AP, over the strongest competitor; (ii) our DBGAN outperforms ARGA that also introduces adversarial learning by about 2.0% and 2.5% on Cora and Citeseer datasets respectively. This just verifies that the effectiveness of our proposed structure-aware prior distribution estimation by DPP-based prototype learning; (iii) the approximate 1.0% improvement of our DBGAN on Citeseer dataset over GALA that is the state-of-the-art method is achieved. This means that our DBGAN has created a new baseline in the area of graph representation learning.

### 4.3. Evaluation on Node Clustering

For node clustering task, the network setup for $G$ and $D_x$ are the same as those for link prediction task. In particular, on Cora dataset, we set the hidden and output layers of $E$ to 64-neuron and 128-neuron, the two hidden layers of $D_z$ to 64-neuron, and the hyperparameters $\alpha$ and $\lambda$ to 0.01 and 1 respectively. On Citeseer and Pubmed datasets, we set the hidden layer and output layer of $E$ to 64-neuron, the two hidden layers of $D_z$ also to 64-neuron, and the hyperparameters $\alpha$ and $\lambda$ to 1e-5 and 1 respectively.

We present the comparative results on node clustering in Table 3. It can be observed that, (i) the performance of such algorithms that use both node features and graph structure can outperform significantly than those using only one of them; (ii) Kmeans [21] that uses only node features improves the overall performance of the methods only using graph structures, by an obvious margin (from 0.80% to 20.4% for ACC). This validates that introducing node features is necessary for node clustering tasks; (iii) it is worth noting that, although GALA [29] outperforms our DBGAN slightly in several cases, it is indeed at the cost of high dimension of latent representation (e.g., 400 for GALA and 128 for ours on Cora, and 500 and 64 on Citeseer).

Table 4. Effectiveness evaluation of BAL and PDE.

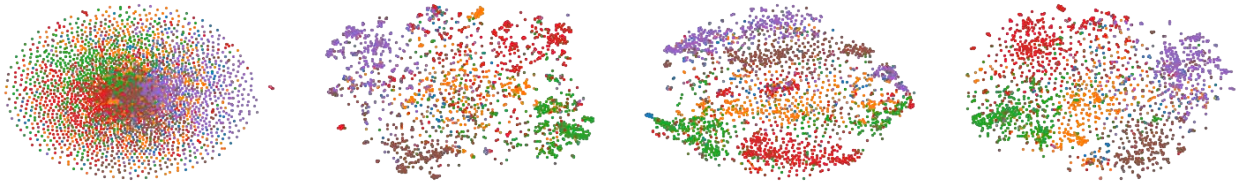
| Methods | Link Prediction | | Clustering | | |
|---|---|---|---|---|---|
| | AUC | AP | ACC | NMI | ARI |
| w/o both | 91.0 | 92.0 | 0.596 | 0.429 | 0.347 |
| w/o PDE | 93.1 | 93.9 | 0.684 | 0.472 | 0.431 |
| w/o BAL | 92.5 | 93.2 | 0.535 | 0.389 | 0.313 |
| DBGAN | 94.5 | 95.1 | 0.759 | 0.551 | 0.525 |

### 4.4. Ablation Study

On both link prediction and node clustering tasks with Cora dataset, we validate the effectiveness of bidirectional adversarial learning **(BAL)** and structure-aware prior distribution estimation **(PDE)**, respectively. For such an ablation study, the basic setup about each subnet refers to the experiments above. As shown in Table 4, both BAL and PDE are equally important for our DBGAN to learn latent node representations. Specifically, compared with the baseline method 'w/o both' without bidirectional adversarial learning and prior distribution estimation, the 'w/o PDE' and 'w/o BAL' receive obvious benefits for link prediction (e.g., 2.1% and 1.5% on AUC). Similarly, our DBGAN with both BAL and PDE achieves consistently the best performance over the three ablated methods. It can also be observed that there exists a performance decrease for 'w/o BAL' on clustering task over 'w/o both'. But the improvement of DBGAN implies that employing PDE facilitates BAL to great extent.

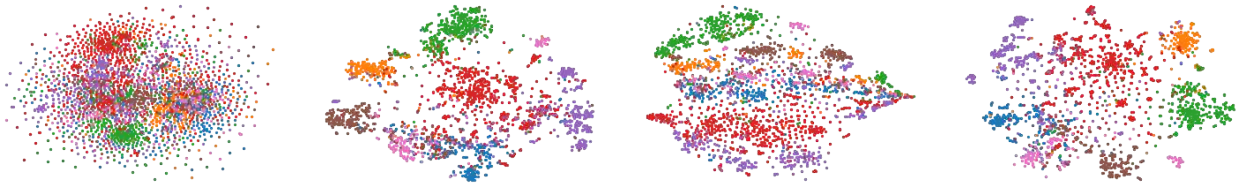
### 4.5. Efficiency Analysis

The dimension of latent representation has a great effect on graph representation learning. To verify this fact, we



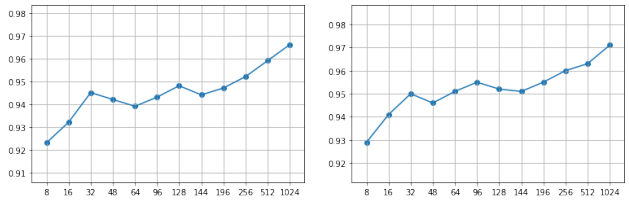
(a) Citeseer(raw) (b) Citeseer(GAE) (c) Citeseer(DGI) (d) Citeseer(DBGAN)

Figure 3. Visualization of the Citeseer dataset.



(a) Cora(raw) (b) Cora(GAE) (c) Cora(DGI) (d) Cora(DBGAN)

Figure 4. Visualization of the Cora dataset.




(a) AUC (b) AP

Figure 5. Impact of the dimension $q$ of learned latent representation on AUC and AP for link prediction task.

vary the dimension of encoder output layer from 8-neuron to 1024-neuron for Cora dataset on link prediction task. The score achieved by DBGAN is shown in Figure5. Obviously, the performance of DBGAN keeps improving with dimension increasing. For a fair comparison, a low dimension is fixed in all our experiments. In particular, all our dimension is no more than 128, while the compared methods are generally opposite. Even though in this case, we still achieve more promising results as in Tables 2 and 3. This further verifies the dimension-efficient property of DBGAN.

### 4.6. Graph Visualization

A promising unsupervised graph representation algorithm can usually preserve the original graph structure well in a low-dimensional space. To illustrate such a representation ability more intuitively, we use t-SNE [23] to visualize the learned latent representations and original node features in a two-dimensional space. Figure3 and Figure4 show the visualization results on Cora and Citeseer datasets respectively. It can be seen that although our DBGAN performs graph representation learning in an unsupervised manner, it still can generate node representations that well preserve original adjacency relationships. Meanwhile, compared with raw features and the representations learned by DGI [35], the results by our DBGAN is more discriminative, with smaller within-class scatter and larger inter-class scatter. Specially, we can find that on Cora dataset, there exists many overlaps between pink and blue dots for GAE [17], while such a phenomenon is alleviated greatly for our DBGAN.

## 5. Conclusion

In this paper, we propose a distribution-induced bidirectional adversarial learning network (named **DBGAN**) for graph representation learning. It is able to estimate the structure-aware prior distribution of latent representation via the learned prototypes, instead of the widely used Gaussian assumption, thus generating robust and discriminative representation of nodes. More importantly, the generalization ability of our DBGAN is improved greatly while preserving representation ability, by balancing multi-level consistency with a bidirectional adversarial learning framework. We have carried out extensive experiments on three tasks, and the results demonstrate the obvious superiority of our DBGAN over currently available alternatives in graph representation learning. Our ongoing research work will extend our DBGAN to graph representation learning in the semi-supervised scenario.

## Acknowledgement

This work was supported in part by Science and Technology Innovation 2030 - "New Generation Artificial Intelligence" Major Project under Grant 2018AAA0102101, in part by the National Natural Science Foundation of China under Grant 61976018 and Grant 61532005, and in part by the Fundamental Research Funds for the Central Universities under Grant 2018JBZ001 and Grant 2019YJS048.

## References


[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223, 2017.

[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900. ACM, 2015.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[4] Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009.

[5] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. Adversarial network embedding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Ming Ding, Jie Tang, and Jie Zhang. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 913–922. ACM, 2018.

[7] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[8] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544*, 2019.

[9] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[10] Hongchang Gao, Jian Pei, and Heng Huang. Progan: Network embedding via proximity generative adversarial network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1308–1316. ACM, 2019.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[14] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[18] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1193–1200, 2011.

[19] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[20] Fuzhen Li, Zhenfeng Zhu, Xingxing Zhang, Jian Cheng, and Yao Zhao. Diffusion induced graph representation learning. *Neurocomputing*, 2019.

[21] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[22] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503, 2003.

[23] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[24] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[25] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, page 8, 2012.

[26] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[27] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[28] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

[29] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6519–6528, 2019.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[31] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467. ACM, 2018.

[32] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[34] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[35] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[36] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898. ACM, 2017.

[37] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

[38] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[40] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[41] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[42] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[43] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2663–2671. ACM, 2018.