# Adversarial Representation Mechanism Learning for Network Embedding

Dongxiao He, Tao Wang, Lu Zhai, Di Jin, Liang Yang, Yuxiao Huang, Zhiyong Feng*, and Philip S. Yu, *Life Fellow, IEEE*

**Abstract**—Network embedding which is to learn a low dimensional representation of nodes in a network has been used in many network analysis tasks. Some network embedding methods, including those based on Generative Adversarial Networks (GAN) (a promising deep learning model), have been proposed recently. Existing GAN-based methods typically use GAN to learn a Gaussian distribution as a prior for network embedding, which makes it difficult to distinguish the node representation from Gaussian distribution. It did not apply the adversarial learning strategy on the representation mechanism but just on representation results. Thus, it does not make full use of the essential advantage of GAN, and leads to compromised performance of the method. To address this problem, we propose a novel adversarial learning framework consisting of three players for network embedding, which applies the adversarial learning strategy on the representation mechanism, called Adversarial representation mechanism GAN (ArmGAN). Specifically, the first two players, named encoder and competitor, aim to learn two different representation mechanisms (i.e., two ways projecting data onto latent space). They compete with each other to improve their representation mechanisms. The third player is the discriminator, which discriminate the representation mechanism of the encoder from that of the competitor. In addition, we design a perturbation strategy to produce fake networks from the original network, and feed the fake networks to the competitor to obtain a "fake" representation mechanism. We evaluated ArmGAN on a variety of tasks including node clustering, node classification, link prediction and visualization. Moreover, we compared ArmGAN with 10 state-of-the-art methods (including DGI, which is well-known for its high accuracy) on 7 real-world networks. The experimental results show the significant superiority of ArmGAN over the existing methods.

**Index Terms**—Network embedding, Generative adversarial network, Graph neural network, Social network analysis.

## 1 Introduction

Networks provide a ubiquitous way to organize data, where edges represent complex relationships and nodes encode rich information in the data. An effective method for analyzing networks is network representation learning (a.k.a., node embedding), which aims to learn the low-dimensional latent representation of nodes in the network [1], [2], [3], [4]. As the learned representations encode the topology and node content information, the representations can be used for network analysis tasks such as link prediction, node classification, network visualization, user recommendation and community detection [5], [6], [7], [8], [9]. Network representation learning algorithms can be divided into two categories, semi-supervised algorithms and unsupervised algorithms. Methods in the first category, such as MMDW [10], GCN [11], and SSNE [12], introduce a small amount of prior information. Methods in the second category do not use any label information, which includes DGI [13], DeepWalk [1], node2vec [14], MNMF [15], LINE [2], GraRep [3], TADW [16], SNE [17], TriDNR [18] and AANE [19]. In this paper, we focus on unsupervised representation learning, since it is the more general and popular category.

Generative Adversarial Networks (GANs) [20] have become a powerful deep generative model. GAN is inspired by the two-player game in game theory. The two players in GAN are a generator $G$ (generating data that resemble real data) and a discriminator $D$ (distinguishing real data from generated data). In other words, the generator's goal is to "fool" the discriminator by generating data that are as similar to the real data as possible. The discriminator's goal is to "debunk" the generator by discriminating between real data and generated data.

While GANs were originally proposed to generate images, recently they have been extended to network embedding. For example, ARVGA [21] leverages adversarial learning to regularize the embedding results of graph autoencoder [22], i.e., forcing the embedding to match Gaussian distribution. This framework was further extended by letting autoencoder reconstruct both the topology and node attributes instead of just reconstructing network structure in ARVGA [23]. ANE [24] proposes an inductive variant of DeepWalk [1] for preserving network structure properties in latent space and leverages adversarial learning by matching latent representations to given priors (such as Uniform or Gaussian distribution). VANE [25] proposes a multi-view adversarial framework that is based on two adversarial games, where the first game enhances the comprehensiveness of the node representation by discriminating different views information and the second game ensures the robustness of node representations by fitting the distribution of node representations to a given noise distribution. GANE


- *D. He, T. Wang, L. Zhai, D. Jin, Z. Feng are with College of Intelligence and Computing, Tianjin University, Tianjin 300350, China. E-mail: {hedongxiao, wt2019216113, xi1895, jindi, zyfeng}@tju.edu.cn.*
- *L. Yang is with School of Artificial Intelligence, Hebei University of Technology, Tianjin 300131, China. E-mail: yangliang@vip.qq.com*
- *Y. Huang is with Data Science, George Washington University, Washington, D.C. 20052, USA. E-mail: yuxiaohuang@gwu.edu.*
- *P.S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA, and also with the Institute for Data Science, Tsinghua University, Beijing 10085, China. E-mail: psyu@cs.uic.edu.*
- **Zhiyong Feng is the corresponding author.*

[26] is capable of performing feature representation learning and link prediction simultaneously, using GANs to regularize the vertex pairs by forcing the generated vertex pairs to resemble the real data. To sum up, most existing network embedding approaches with adversarial learning can be categorized into the framework of adversarially forcing the embedding results to follow a given or latent distribution. A more comprehensive introduction of GAN-based network embedding methods are given in Related Work.

It is worth noting that these existing methods typically apply the adversarial learning strategy on the representation result, e.g. matching the distribution of representation to an arbitrary prior, such as Gaussian distribution in most cases. However, this strategy makes it difficult to distinguish the representation from Gaussian noise, since it requires the representation to obey the Gaussian distribution, which roughly equals adding a Gaussian regular term to the representation. While this is reasonable to some extent, it does not make full use of the essential advantage of adversarial learning. We believe it is better to apply the adversarial learning strategy on the representation mechanism that projects data onto latent space (to make the system robust and effective) rather than on the representation itself (which simply requires the representation distribution to follow some priors). However, it is difficult to apply the adversarial learning strategy on the representation mechanism for the GAN models, as they contain only two players (i.e., encoder/generator and discriminator) and only the encoder realizes the representation mechanism from data space to latent space.

To address this problem, we propose a novel adversarial learning framework for network embedding, called Adversarial representation mechanism GAN (ArmGAN), which applies the adversarial learning strategy on the representation mechanism. To achieve the adversarial training of the representation mechanism, the new framework contains three players. The first two players host two different representation mechanisms (i.e., two different ways to project data space onto latent space), named encoder and competitor. They compete with each other to improve their representation mechanisms under the guide of adversarial principle and the whole framework. The third player is the discriminator, which discriminates the representation mechanism of the encoder from that of the competitor. The three players adversarially learn with each other in the new system. This is fairly different from the framework of the existing GAN-based network embedding methods since it is impossible for them to achieve the adversarial training of the representation mechanism because they have only one player to host representation mechanism. Furthermore, the new framework has a new type of relationship between the three players. The goal of the encoder and that of the discriminator are consistent, as they work together to let the "real" encoding mechanism from encoder be taken as "real" one. In contrast, the goal of the competitor is to "fool" the discriminator by acting like encoder, i.e., it pretends to be the real representation mechanism to deceive the discriminator. The remaining question is how to design the competitor player. There are three conditions the competitor should meet. First, it should be a competitive representation mechanism from data to latent space. Second, it should be a "fake" representation mechanism. Third, it should be a neural network itself so as to adapt to the overall neural network framework (so that the whole model can be trained jointly using backpropagation). To meet the first and the third condition, we use another encoder (which is a neural network with different weights and different optimization objectives) as the competitor. At the same time, to satisfy the second condition, we design a procedure that produces fake networks from the original network, and feed the fake networks to the competitor to form a "fake" representation mechanism.

## 2 Related Work

Besides GAN-based network embedding methods discussed in Introduction, here we introduce other major approaches along this line. GraphGAN [27] unifies generative and discriminative thinking to generate the most likely neighbor node representation for a given node and tries to make the generator fit the underlying true connectivity distribution. DGGAN [28] extends GraphGAN to directed graph, so as to preserve the directionality of edges. ProGAN [29] proposes a novel proximity generative adversarial network for network embedding, which can generate proximities through adversarial learning. The generated proximities can help to discover the complicated underlying proximity to improve network embedding. JANE [30] proposes a joint adversarial network embedding framework which jointly distinguishes the real and fake combinations of the embeddings, topology information and node features, so as to learn the latent semantic space and capture semantic variations. Khajehnejad. et al [31] proposes an adversarial graph embedding method for fair influence maximization over social networks, which consists of an auto-encoder for graph embedding and a discriminator for discerning sensitive attributes, so as to guarantee fair influence maximization. AdONE [32] proposes an autoencoder framework to learn node embeddings for networks with outliers. It leverages adversarial learning to align the embeddings corresponding to the link structure and node attributes so that they can complement each other and further weights the objective function with outlier scores to minimize the effect of outliers. TriATNE [33] designs a tripartite adversarial learning model based on sales skills in the market, which includes a producer, a seller and a customer to preserve high order graph structure and learn more stable and robust representation. NINE [34] gives a network embedding method which aims to preserve node pair information between connected and disconnected node pairs by designing two discriminators which discriminate connected node pair and disconnected node pair respectively. It is worth noting that these existing GAN-based network embedding approaches only apply adversarial learning on representation results, which does not make full use of the essential advantage of GAN (that is to adversarially learn the representation mechanism rather than the representation itself).

## 3 Problem Definition

Consider an undirected, unweighted and attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ with $n$ nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, a

set of edges $\mathcal{E} = \{e_{ij}\} \subseteq \mathcal{V} \times \mathcal{V}$, and a set of node attribute $X \in \mathbb{R}^{n \times m}$, where $m$ represents the number of attributes of each node. The topological structure of $\mathcal{G}$ is represented by an adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ where $a_{ij} = 1$ if nodes $v_i$ and $v_j$ are connected, or $a_{ij} = 0$ otherwise. Attribute $x_i \in \mathbb{R}^m$ specifies the features or properties of node $v_i$. The objective of network embedding is to learn a low-dimensional embedding matrix $Z \in \mathbb{R}^{n \times d}$ from the topological structure $A$ and attributes $X$, whose formal format is $\mathrm{f} : (A, X) \rightarrowtail Z$, where $d$ is the dimension of embeddings. The learned embedding $Z$ should well preserve the topological structure $A$ as well as attribute information $X$.

## 4 The Approach

We first give a brief overview of the proposed method, and then introduce three elements of the model in detail. Last, we formly propose Adversarial Representation Mechanism Learning model (ArmGAN).

### 4.1 Overview

To make full use of the advantage of adversarial learning to get an effective embedding mechanism, we propose a novel adversarial learning framework with three players (the encoder, the competitor and the discriminator) and adversarially train representation mechanisms from data space to latent space. For the encoder that represents positive representation mechanism, we adopt a two-layer graph convolutional network (GCN) [11] which integrates network attributes and topology information to generate a low-dimensional embedding $Z$. Furthermore, we use a decoder and a mutual information regularity to constrain the encoder, which can be called as autoencoder with mutual information regularity (shown in the top part of Fig. 1). The decoder measures the loss of reconstructing the real network topology by using the low-dimensional embedding $Z$. The mutual information regularity lets embedding $Z$ represent the attribute information $X$ to the greatest extent. Minimizing the reconstruction loss of network topology and maximizing the mutual information between the node attributes and the embedding together can guide the encoder to generate an effective embedding $Z$ that contains both topology information and attribute information naturally. From another perspective, encoder with two-layer GCN is limited by only gathering the information of two-order neighbor nodes while the mutual information regularity can capture the non-linear statistical dependence between the real node attributes $X$ and embedding $Z$. Thus, the mutual information regularity can effectively compensate the drawback of the classic autoencoder. The above two points show that the autoencoder with mutual information regularity can play the role of positive representation mechanism well. For the competitor which should provide a "fake" (negative) but competitive representation mechanism, we design a strategy to generate "fake" network data. Meanwhile, we also use the framework of the autoencoder with mutual information regularity, but use a different objective function. Then we feed the "fake" network data to the designed competitor to form a negative representation mechanism. As the competitor produces negative representation mechanism, we also call the competitor the negative sample generator as shown in the bottom part of Fig. 1. It is worth noting that the neural networks of the negative sample generator have different weights from the positive sample generator, as the objective is different. Since the negative sample generator uses the same framework as the autoencoder with mutual information regularity (that generates the positive representation mechanism) and it is trained under a different optimization objective, the negative sample generator can generate a competitive representation mechanism. Then, the last player, i.e., the representation mechanism discriminator is used to distinguish the representation mechanism of the autoencoder with mutual information regularity from that of the negative sample generator.

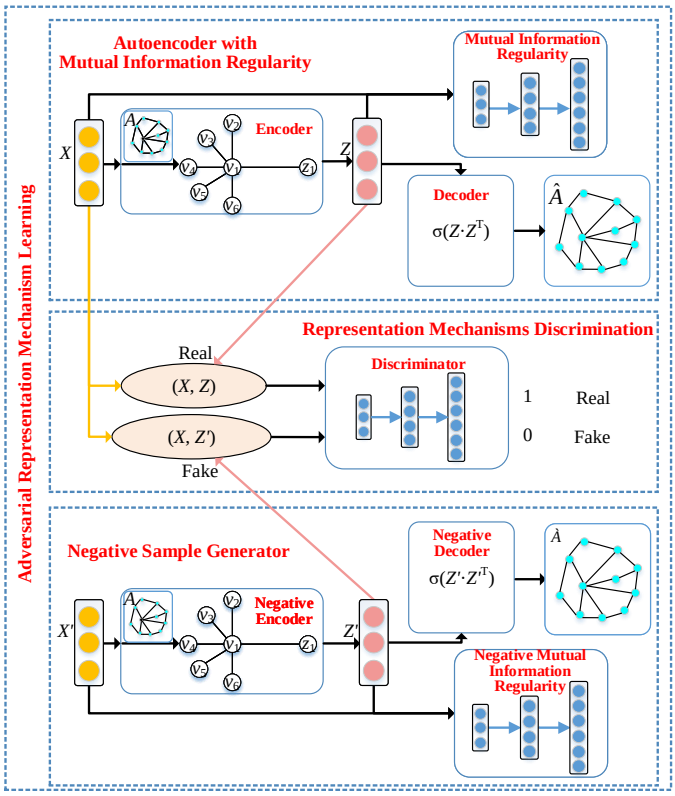



Fig. 1: The structure of ArmGAN. It consists of three parts, an autoencoder with mutual information regularity (encoder, decoder and mutual information regularity, shown on the top of the figure), the representation mechanism discrimination (discriminator, shown in the middle) and a negative sample generator (negative encoder, negative decoder and negative mutual information regularity, shown on the bottom).

Accordingly, the new model ArmGAN has a new type of relationship among three players. The representation mechanism discriminator distinguishes the positive and negative representation mechanisms. The autoencoder with mutual information regularity that generates positive representation mechanism helps the discriminator to realize its discrimination task, i.e., helps the discriminator to correctly identify the positive representation mechanism as "real". However, the purpose of the negative sample generator is the opposite, which is to deceive the discriminator by gen-

erating negative samples similar to the real representation mechanism, i.e., it is to obstruct the discrimination task of the discriminator. In the new system with new relationship, the three players are trained under new objective functions. The autoencoder with mutual information regularity is trained with three objectives: a traditional reconstruction error criterion from decoder using positive embedding $Z$, a mutual information criterion from mutual information regularity to represent the real node attribute information $X$ to the greatest extent, and an adversarial criterion from discriminator which is to help the discriminator to identify the positive representation mechanism as "real". The training of negative sample generator also has three objectives: the reconstruction error criterion from negative decoder using negative embedding $Z'$, a mutual information criterion from negative mutual information regularity to represent the fake node attribute information $X'$, and an adversarial criterion from discriminator which is to deceive the discriminator so as to make the discrimination process difficult. Then, the discriminator is trained by the objective that, on one hand, discriminates the representation mechanism from autoencoder with mutual information regularity as "real" as much as possible and, on the other hand, discriminates the representation mechanism from negative sample generator as "fake" as much as possible.

It is worth noting that this new adversarial learning framework ArmGAN is fairly different from existing GAN-based models for embedding which contain two players (i.e., encoder/generator and discriminator). Concretely, in the existing GAN-based models, only encoder/generator hosts a representation mechanism from data space to latent space. There is no other players that host another representation mechanism, and thus it is impossible to realize the adversarial training on representation mechanism. On the contrary, our new adversarial learning framework contains three players: two of them host the real representation mechanism and the negative representation mechanism, respectively. These two types of representation mechanisms can be adversarially trained. Moreover, the purpose of the encoder/generator in the existing GAN-based models is to deceive the discriminator by producing fake samples similar to real ones, while the purposes of the two players that host representation mechanisms in our new model ArmGAN are different. The autoencoder with mutual information regularity is to help the discriminator, while the negative sample generator is to deceive the discriminator. In other words, the goal of autoencoder with mutual information regularity and that of the discriminator are consistent. However, the goal of the negative sample generator is the opposite to that of the discriminator. In this new system, the discriminator can be thought as police, and the autoencoder with mutual information regularity can be thought as good people, while the negative sample generator is analogous to bad people. The good people aim to help the police to correctly identify them as good people. In contrast, by learning the behavior of good people and acting like good people, the bad people try to deceive the police to wrongly take them as good.

### 4.2 Autoencoder with Mutual Information Regularity

In this section, we first introduce the classical autoencoder that includes encoder and decoder, then introduce mutual information regularity, and finally give the whole autoencoder with mutual information regularity as shown in the top part of Fig. 1.

For the autoencoder, we use the graph auto-encoder proposed by [22]. In the encoder, we use the graph convolutional network (GCN), which is a flexible class of node representation mechanism that generates node representations by aggregation over local node neighborhoods, so as to extract the embedding of nodes. Here we use the classic two-layer GCN. Given the adjacency matrix $A$ and attribute matrix $X$ of a network, the model is constructed as

$$\begin{aligned} Z^{(1)} &= f_{\text{Relu}}\left(X, A|W^{(0)}\right); \\ Z^{(2)} &= f_{\text{linear}}\left(Z^{(1)}, A|W^{(1)}\right), \end{aligned} \tag{1}$$

where $Z^{(0)} = X$, and each convolutional layer is expressed by

$$f\left(Z^{(l)}, A|W^{(l)}\right) = \phi\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Z^{(l)}W^{(l)}\right). \tag{2}$$

Here $\tilde{A} = A + I$ (where $I$ is the identity matrix) and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $W^{(l)}$ denotes the weight matrix of the $l$-th layer, and $Z^{(l)}$ is the input of the $l$-th layer. $\phi$ is an activation function, and we use $\text{Relu}(t) = \max(0, t)$ in the first layer and use $\text{linear}(t) = t$ in the second layer. In the decoder, we reconstruct the network topology using the embedding derived from the encoder. With the embedding $Z(Z = Z^{(2)})$, the reconstructed graph $\hat{A}$ can be presented as

$$\hat{A} = \text{sigmoid}\left(ZZ^T\right). \tag{3}$$

We then use the cross entropy to define the reconstruction loss as

$$\mathcal{L}_{AE} = -\sum \mathbb{E}\left[a_{ij}\log\hat{A}_{ij} + (1 - a_{ij})\log\left(1 - \hat{A}_{ij}\right)\right]. \tag{4}$$

In order to add the constraints of attribute information and add the non-linear statistical dependence to the encoder, we introduce mutual information regularity to the traditional autoencoder. Mutual information quantifies the dependence of two random variables $X$ and $Z$, which is equivalent to the Kullback-Leibler (KL-) divergence between the joint distribution of these two variables and the product of the marginals of these two variables [35]. KL-divergence can be expressed in two ways, including Donsker-Varadhan representation [36] and f-divergence representation [37], [38]. Belghazi et al. [39] gave a method to estimate mutual information based on neural networks by maximizing the lower-bound of Donsker-Varadhan representation or f-divergence representation. Belghazi et al. [39] also showed the expressive power of neural network insures that they can approximate the mutual information with arbitrary accuracy. To be specific, Belghazi et al. [39] trained a statistics network as a classifier to distinguish samples coming from the joint distribution or the product of marginals of two random variables. The joint samples can be sampled from the joint distribution of two variables. For the marginals of these two variables, they can be gotten by empirical samples or by shuffling the samples from the joint distribution [39]. In this paper, our mutual information regularity uses a three-layer fully connected neural network to approximate the mutual information between node attributes $X$ and

embedding $Z$. The neural network can be expressed by the function $T_\theta : X \times Z \to \mathbb{R}$ with parameter $\theta$ in some compact domain $\theta \in \Theta^k$, where $k$ is the dimension of the parameter space.

We use the combination of node attributes $X$ and the corresponding embedding $Z$ from the encoder, i.e., $(X, Z)$, as the joint distribution. To get the product of marginals, we randomly shuffle the rows of node attributes $X$ and get the corrupted node attributes $\bar{X}$, and then we input the corrupted attributes matrix $\bar{X}$ into the encoder to get the corrupted embedding $\bar{Z}$. Then we use the combination of the node attributes $X$ and the corrupted embedding $\bar{Z}$, i.e., $(X, \bar{Z})$, as the product of the marginals. In order to obtain representative embedding, we maximize the mutual information between node attributes $X$ and embedding $Z$. Then, according to [39], the mutual information $I_\Theta(X, Z)$, which is estimated by neural networks based on Donsker-Varadhan representation, is defined as follows :

$$\begin{aligned} I_\Theta(X, Z) = \sup_{\theta \in \Theta} \big( & \mathbb{E}_{(x,z) \sim p_{xz}(x,z)} T_\theta(x, z) \\ & - \log \mathbb{E}_{x \sim p_{data}(x), \bar{z} \sim p_{\bar{z}}(\bar{z})} \left( e^{T_\theta(x, \bar{z})} \right) \big) \end{aligned} \tag{5}$$

where $T_\theta(\cdot)$ is the neural network with parameter $\theta$ mentioned above and it is optimized by maximizing Eq. (5), $p_{xz}(x, z)$ is the joint distribution of $x$ and $z$, $p_{data}(x)$ is the sample distribution of $x$ and $p_{\bar{z}}(\bar{z})$ is a sample distribution of $\bar{z}$ generated by encoder using the corrupted node attributes $\bar{X}$ and real network topology.

Now we will give the autoencoder with mutual information regularity and define its objective function. Its overall objective function contains two parts: one is from decoder and the other is from mutual information regularity. The one from decoder is to reconstruct the real network topology by using the embedding $Z$ generated by the encoder (i.e., minimizing the reconstruction loss of network topology). The other is to let embedding $Z$ represent real attribute information $X$ to the greatest extent (i.e., maximizing the mutual information between the node attributes $X$ and the embedding $Z$). As we need to minimize the reconstruction loss as well as maximize the mutual information, we use the negative mutual information so that we can minimize both. Then, we get the overall objective function of the autoencoder with mutual information regularity which is defined as follow:

$$\mathcal{L}_{AMIR} = \mathcal{L}_{AE} - \beta_1 I_\Theta(X, Z). \tag{6}$$

where $\mathcal{L}_{AE}$ is the loss of the reconstructed topology, expressed as Eq. (4). $I_\Theta(X, Z)$ is the estimated mutual information, expressed as Eq. (5). $\beta_1$ is a hyperparameter, representing the proportion of the the mutual information regularity to the total objective function $\mathcal{L}_{AMIR}$. We use batch gradient descent to minimize this objective function.

### 4.3 Negative Sample Generator

In this section, we first introduce how to generate fake network data, and then introduce the negative sample generator which also uses the framework of autoencoder with mutual information regularity but with a different optimization objective. In order to distinguish the negative sample generator from the autoencoder with mutual information regularity, we called the three components of negative sample generator as the negative encoder, negative decoder and negative mutual information regularity as shown in the bottom part of Fig. 1.

We design a perturbation strategy to obtain a fake network data from the original network. In order to make the generated "fake" representation mechanism competitive, here we choose to perturb just one type of the network data (network topology or node attributes). Then the real data lets the embedding represent the real data through the corresponding constraint term (negative decoder / negative mutual information regularity). Meanwhile, the fake data lets the embedding add some noise through the other constraint term (negative mutual information regularity / negative decoder). In this paper, we choose to use the real topology and a fake attribute data, as it produces the best results in our perturbation experiments (where different strategies were used). To be specific, we preserve the original topological structure but corrupt the node attributes, $X'$, via row-wise shuffling of real node attributes $X$.

After getting the fake network, the reconstruction loss from negative decoder can be defined by using negative embedding $Z'$ which is generated by the negative encoder. It is worth noting that this reconstruction loss is different from the reconstruction loss of decoder which is defined by positive embedding $Z$ generated by the encoder. Specifically, the new reconstruction loss is defined by the following cross entropy:

$$\mathcal{L}_{AE'} = -\sum \mathbb{E} \left[ a_{ij} \log \grave{A}_{ij} + (1 - a_{ij}) \log \left( 1 - \grave{A}_{ij} \right) \right] \tag{7}$$

where $\grave{A} = \text{sigmoid}\left(Z' Z'^T\right)$ is the reconstructed matrix through the negative decoder. This is different from the reconstructed topology matrix $\hat{A} = \text{sigmoid}\left(ZZ^T\right)$ mentioned in Section 4.2, where $\hat{A}$ is generated by the decoder using positive embedding $Z$.

Then, we give the negative mutual information regularity, i.e., the other constraint term of negative encoder. This constraint term lets the negative embedding $Z'$ represent "fake" attribute information $X'$ to the greatest extent by maximizing the mutual information between the "fake" node attributes $X'$ and the negative embedding $Z'$. We use the same method as that in Section 4.2 to estimate the mutual information. Specifically, we use the combination of the "fake" node attributes $X'$ and the negative embedding $Z'$ , i.e., $(X', Z')$, as the joint distribution. To get the product of marginals, we randomly perturb the rows of the "fake" node attributes $X'$ to obtain the corrupted "fake" node attributes $\bar{X}'$, and then we input the corrupted "fake" node attributes $\bar{X}'$ and topological matrix $A$ into the negative encoder to get the corrupted "fake" embedding $\bar{Z}'$. We use the combination of the "fake" node attributes $X'$ and the corrupted "fake" embedding $\bar{Z}'$, i.e., $(X', \bar{Z}')$, as the product of the marginals. Then, the mutual information $I'_{\Theta'}(X', Z')$, which is estimated by neural networks, is defined as:

$$\begin{aligned} I'_{\Theta'}(X', Z') = \sup_{\theta' \in \Theta'} \big( & \mathbb{E}_{(x',z') \sim p_{x'z'}(x',z')} T'_{\theta'}(x', z') \\ & - \log \mathbb{E}_{x' \sim p_{x'}(x'), \bar{z}' \sim p_{\bar{z}'}(\bar{z}')} \left( e^{T'_{\theta'}(x', \bar{z}')} \right) \big) \end{aligned} \tag{8}$$

where $T'_{\theta'}(\cdot)$ is the neural networks with parameter $\theta'$ and is optimized by maximizing Eq. (8), $p_{x'z'}(x', z')$ is the joint

distribution of $x'$ and $z'$, $p_{x'}(x')$ is the sample distribution of the fake node attributes $x'$ and $p_{\bar{z}'}(\bar{z}')$ is a distribution of samples that are generated by negative encoder which uses the corrupted "fake" node attributes $\bar{X}'$ and real network topology as input.

Last, we give the negative sample generator and the overall objective function. Specifically, the negative sample generator is trained in the guidance of two objectives. One is to reconstruct the real network topology by using the negative embedding $Z'$ generated by the negative encoder. Another objective is to let negative embedding $Z'$ represent "fake" attribute information $X'$ to the greatest extent by maximizing the mutual information between the "fake" node attributes $X'$ and the negative embedding $Z'$. Similar to the discussion in Section 4.2, here we add a negative sign to the negative mutual information regularity so as to minimize both objectives as a whole. After getting a unified objective function, we train the negative sample generator by using gradient descent. The overall objective function of the negative sample generator is then defined as:

$$\mathcal{L}_{NSG} = \mathcal{L}_{AE'} - \beta_2 I^{'}_{\Theta'}(X', Z') \tag{9}$$

where $\mathcal{L}_{AE'}$ is the loss of the reconstructed topology, expressed as Eq. (7). $I^{'}_{\Theta'}(X', Z')$ is the mutual information estimated by neural networks, expressed as Eq. (8). $\beta_2$ is a hyperparameter, representing the proportion of the negative mutual information regularity to the total objective function $\mathcal{L}_{NSG}$.

### 4.4 The Representation Mechanism Discriminator

The *core* of our model is to adversarially learn the representation mechanism rather than the representation result. As discussed earlier, our new framework ArmGAN comprises two representation mechanisms, i.e., the positive representation mechanism which is hosted by autoencoder with mutual information regularity and the negative representation mechanism which is hosted by negative sample generator. The challenge is how to turn these two types of representation mechanism into recognizable inputs of the representation mechanism discriminator. In fact, the representation mechanism can be regarded as a mapping mechanism. According to [40], the mapping mechanism can be expressed approximately by the combination of the input and output of the mapping, which is much easier to track. Therefore, we use the combination of the node attributes (input of the mapping) and embedding (output of the mapping) as the recognizable input of the discriminator to represent our representation mechanism. The task of the representation mechanism discriminator (as shown in the middle part of Fig. 1) is to distinguish the representation mechanism of the autoencoder with mutual information regularity (which can be called as the positive sample generator) from that of the negative sample generator.

Specifically, for the positive representation mechanism, we use the combination of the attribute information (the input of the positive sample generator) and the node representation (the output of the positive sample generator) ,i.e., $(X, Z)$, to represent it. As for the negative representation mechanism, we adopt two ways to implement it. Each way has its own characteristics, and is suitable for solving different network analysis tasks.

*Direct mapping representation mechanism.* When using this way to implement the negative representation mechanism, we concatenate the input $X'$ of the negative sample generator (i.e., the "fake" node attributes $X'$ which are obtained by randomly shuffling the node attribute $X$) and output $Z'$ of the negative sample generator (i.e., the corresponding embedding which is generated by the negative sample generator based on the "fake" node attributions $X'$), i.e., $(X', Z')$, as the negative mapping mechanism. The task of the discriminator is to distinguish the positive representation mechanism (i.e., the mapping mechanisms sampled from the autoencoder with mutual information regularity) from the negative representation mechanism (i.e., the mapping mechanisms sampled from the negative sample generator). The discriminator outputs a single scalar which represents the probability that the representation mechanism came from the autoencoder with mutual information regularity rather than negative sample generator. We train the discriminator so as to maximize the probability of assigning the correct label to both positive representation mechanism $(X, Z)$ and the negative representation mechanism $(X', Z')$. In other words, the discriminator attempts to discriminate the negative representation mechanism samples as 0, and the positive representation mechanism samples as 1. It is worth noting that the larger the objective function of the discriminator, the better. This is different from the objective functions mentioned in the previous section, such as the objective function of autoencoder with mutual information regularity and the objective function of negative sample generator, which are the smaller the better. Then, we give the definition of the objective function of the discriminator under the direct mapping representation mechanism:

$$\begin{aligned} V_{DX'} =& \mathbb{E}_{x\sim p_{data}(x)}\left[\log D\left(x, E(x)\right)\right] \\ &+ \mathbb{E}_{x'\sim p_{x'}(x')}\left[\log\left(1 - D\left(x', E'(x')\right)\right)\right] \end{aligned} \tag{10}$$

where $E(x)$ and $E'(x')$ represent the outputs of the encoder and the negative encoder, i.e., embedding $z$ and $z'$. $D(\cdot)$ represents the output of the discriminator.

*Mutual information representation mechanism.* When using this way to implement the negative representation mechanism, we concatenate the real node attribute $X$ and the output $Z'$ of the negative sample generator based on "fake" node attributes $X'$, i.e., $(X, Z')$, as the negative mapping mechanism. Then, we give the definition of the objective function of the discriminator under this representation mechanism:

$$\begin{aligned} V_{DX} =& \mathbb{E}_{x\sim p_{data}(x)}\left[\log D\left(x, E(x)\right)\right] \\ &+ \mathbb{E}_{x'\sim p_{x'}(x')}\left[\log\left(1 - D\left(x, E'(x')\right)\right)\right] \end{aligned} \tag{11}$$

where $E(x)$ and $E'(x')$ represent the outputs of the encoder and the negative encoder respectively, i.e., embedding $z$ and $z'$. $D(\cdot)$ represents the output of the discriminator. Here we maximize this objective function so as to maximize the probability of assigning the correct label to both positive representation mechanism and the negative representation mechanism (hence the larger the better).

From another perspective, the combination of real node attributes $X$ and embedding $Z$ (i.e., $E(X)$) generated by

encoder can be seen as the joint distribution, and the combination of real node attributes $X$ and embedding $Z'$ generated by the negative encoder based on "fake" node attributes $X'$ can be seen as the product of the marginals. According to [37], Eq. (11) can be seen as a measure of the JensenShannon (JS) divergence (one of the f-divergence introduced in Section 4.2) between the joint and the product of the marginals, which actually can be seen as an estimate of the mutual information based on f-divergence representation between node attributes $X$ and embedding $Z$. Thus, we call it mutual information representation mechanism.

Mutual information [41] can capture the inherent dependence and maximal relevance between the real node attributes $X$ and embedding $Z$. And encoder actually is a compression mechanism which compresses high-dimensional data into low-dimension data. The above two points encourage the encoder to retain the most representative information in the low-dimensional embedding $Z$. ArmGAN with mutual information representation mechanism is equivalent to adding another mutual information constraint which is implemented by GAN framework. This mutual information constraint is another type of mutual information which is based on the f-divergence representation and is different from the mutual information regularity used in Sections 4.2 and 4.3, which is based on DV-representation. The model with two mutual information constraint terms can strengthen the power of the model for extracting more representative features. Therefore, it is more suitable for the tasks of node classification and node clustering. In the ArmGAN with direct mapping representation mechanism, the negative representation mechanism is represented by "fake" node attribute $X'$ and its corresponding embedding $Z'$. The embedding $Z'$ is generated based on a "fake" network with perturbed node attribute $X'$ and real network topology $A$, which is equivalent to moving the attributes of high-order (or remote) neighbor nodes to the positions of direct (first-order) neighbors. Thus, the negative sample generator can collect the information of high-order or remote neighbor nodes, and further pass this type of information to the positive sample generator by generative adversarial learning. Therefore, the embedding $Z$ contains more information (not only information of local neighbors but also that of remote neighbors) which is more conducive for measuring the similarity of two nodes and further predicting whether there is an edge between these two nodes. ArmGAN with direct mapping representation mechanism is more suitable for the task of link prediction. This is also demonstrated by the link prediction experiments.

### 4.5 Adversarial Representation Mechanism Learning (ArmGAN)

In this section, we will give the whole ArmGAN model. The whole training process of ArmGAN contains two parts, i.e., the generation process and discrimination process. In the generation process, we train the autoencoder with mutual information regularity and negative sample generator together. In the discrimination process, we only train the representation mechanism discriminator.

In the generation process, we first show how to train the autoencoder with mutual information regularity. Before doing so, let us first review its role in the whole model. The task of autoencoder with mutual information regularity is to help the discriminator to realize its discriminative task, i.e., to help the discriminator to correctly predict the positive representation mechanism samples $(X, Z)$ as 1. The goal of the encoder and the goal of the discriminator are consistent. We can define the adversarial training criterion of encoder from discriminator, which is based on the feedback of discriminator and is the same as the first term of the objective function of the discriminator:

$$V_{ED} = \mathbb{E}_{x \sim p_{data}(x)} \left[ \log \left( D \left( x, E(x) \right) \right) \right] \tag{12}$$

In fact, in the whole model, the encoder is not only affected by discriminator through generative adversarial learning but also affected by the decoder and mutual information regularity through two constraint terms, which are expressed in Eq. (6). As the goal of the encoder is consistent with that of the discriminator, Eq. (12) should be maximized, while as we mentioned in Section 4.2 Eq. (6) should be minimized. To minimize both, we add a negative sign to Eq. (12), then we get the overall objective function of the encoder:

$$\mathcal{L}_E = -V_{ED} + \alpha_1 \mathcal{L}_{AE} - \beta_1 I_\Theta(X, Z). \tag{13}$$

In Eq. (13), we add a hyperparameter $\alpha_1$ to the reconstruction loss which represents the proportion of the reconstruction loss to the total objective function $\mathcal{L}_E$. And $\beta_1$ represents the proportion of mutual information regularity to the total objective function $\mathcal{L}_E$.

Next we will show how to train the negative sample generator. The goal of the negative sample generator is to deceive the discriminator so that the discriminator wrongly predicts the negative representation mechanism samples $(X', Z')$ or $(X, Z')$ as 1. Obviously, the goal of the negative encoder is opposite to that of the discriminator. We can define the adversarial training criterion of negative encoder from discriminator, which is opposite to the second term of the objective function of discriminator. As the discriminator is to maximize this term, the negative encoder aims to minimize this term, which is just opposite with the discriminator:

$$\begin{gathered} \mathcal{L}_{E'_d D} = \mathbb{E}_{x' \sim p_{x'}(x')} \left[ \log \left( 1 - D \left( x', E' \left( x' \right) \right) \right) \right] \\ \text{or} \\ \mathcal{L}_{E'_m D} = \mathbb{E}_{x \sim p_{data}(x), x' \sim p_{x'}(x')} \left[ \log \left( 1 - D \left( x, E'(x') \right) \right) \right] \end{gathered} \tag{14}$$

where $\mathcal{L}_{E'_d D}$ is for using direct mapping representation mechanism, and $\mathcal{L}_{E'_m D}$ is for using mutual information representation mechanism.

However, according to [20], Eq. (14) may not provide gradient that is large enough to update the negative encoder. Concretely, in the early stage of its learning, when negative encoder is poor, the discriminator can reject samples with high confidence because they are clearly different from the positive samples which are generated by autoencoder with mutual information regularity. In this case, $\log \left( 1 - D \left( x', E' \left( x' \right) \right) \right)$ or $\log \left( 1 - D \left( x, E'(x') \right) \right)$ saturates and, as a result, their gradient will be close to zero. In order to solve this problem, we redefine the adversarial training

criterion according to the improved strategy proposed by [20]:

$$\mathcal{L}_{E'_d D} = -\mathbb{E}_{x' \sim p_{x'}(x')} \left[\log\left(D\left(x', E'(x')\right)\right)\right] \quad \text{or} \quad \mathcal{L}_{E'_m D} = -\mathbb{E}_{x \sim p_{data}(x), x' \sim p_{x'}(x')} \left[\log\left(D\left(x, E'\left(x'\right)\right)\right)\right] \tag{15}$$

where $\mathcal{L}_{E'_d D}$ and $\mathcal{L}_{E'_m D}$ are the adversarial criterion of the negative encoder for using the *direct mapping representation mechanism* and the *mutual information representation mechanism*. This objective function results in the same fixed point of the dynamics of negative encoder and discriminator but provides much larger gradients early in learning. This redefined adversarial criterion of negative encoder also should be minimized.

In the whole ArmGAN model, like the positive encoder, the negative encoder is not only affected by discriminator but also by the negative decoder and negative mutual information regularity. The constraint terms from the negative decoder and negative mutual information regularity are defined in Eq. (9), which also should be minimized like the adversarial criterion for the negative encoder. Then, we can define the overall objective function of the negative encoder as:

$$\mathcal{L}_{E'_d} = \mathcal{L}_{E'_d D} + \alpha_2 \mathcal{L}_{AE'} - \beta_2 I^{'}_{\Theta'}(X', Z') \quad \text{or} \quad \mathcal{L}_{E'_m} = \mathcal{L}_{E'_m D} + \alpha_3 \mathcal{L}_{AE'} - \beta_3 I^{'}_{\Theta'}(X', Z') \tag{16}$$

where $\mathcal{L}_{E'_d}$ and $\mathcal{L}_{E'_m}$ are the objective functions of the negative encoder using the *direct mapping representation mechanism* and the *mutual information representation mechanism*. We add hyperparameters $\alpha_2$ (and $\alpha_3$) to the reconstruction loss which represents the proportion of the reconstruction loss to the total objective function $\mathcal{L}_{E'_d}$ (and $\mathcal{L}_{E'_m}$). $\beta_2$ (and $\beta_3$) represents the proportion of negative mutual information regularity to the total objective function $\mathcal{L}_{E'_d}$ (and $\mathcal{L}_{E'_m}$).

Now, we can give the overall objective function of the generation process as follows:

$$\min_{E,T,E',T'} \left(\mathcal{L}_E + \mathcal{L}_{E'_d}\right) \quad \text{or} \quad \min_{E,T,E',T'} \left(\mathcal{L}_E + \mathcal{L}_{E'_m}\right) \tag{17}$$

In the discrimination process, we train the discriminator to maximize the probability of assigning the correct label to both positive samples from autoencoder with mutual information regularity and negative samples from negative sample generator. As mentioned in Section 4.4, the objective function of the discriminator using the *direct mapping representation mechanism* and *mutual information representation mechanism* are defined by Eq. (10) and Eq. (11). Then the objective function of the discrimination process is given by

$$\max_D V_{DX'} \quad \text{or} \quad \max_D V_{DX} \tag{18}$$

The definitions of $V_{DX'}$ and $V_{DX}$ are in Eq. (10) and Eq. (11) in Section 4.4.

To sum up, in the whole process of adversarial learning, the autoencoder with mutual information regularity, the negative sample generator and the discriminator are trained alternately. That is, when training the autoencoder with mutual information regularity and the negative sample generator, the discriminator is fixed. Alternatively, when training the discriminator, the autoencoder with mutual information regularity and the negative sample generator are fixed. This iterative process continues until convergence. The detailed algorithm description of ArmGAN is provided in Appendix.

## 5 Experiments

In this section, we first give the experimental setup, then compare the new approach ArmGAN with some state-of-the-art methods on four network analysis tasks, i.e., node classification, node clustering, link prediction and network visualization. Next, we give the parameter analysis and perturbation strategy analysis. Last we provide the convergence analysis.

### 5.1 Experimental setup

**Datasets.** Seven publicly available datasets[1] with varying sizes and characteristics are used, which are representative of two types of networks: webpage networks (Cornell, Texas, Washington and Wisconsin from the WebKB dataset) and citation networks (Citeseer, Cora and Pubmed). The WebKB dataset is made up of webpages from four universities (Cornell, Texas, Washington and Wisconsin), in each of which the nodes are partitioned into five groups. For the citation network dataset, the nodes are articles, edges are citations, and the articles are partitioned into different research areas. Other information of seven datasets is summarized in Table 1.

TABLE 1: Datasets Information.

| Dataset | Nodes | Edges | Classes | Attributes |
|---|---|---|---|---|
| Cornell | 195 | 304 | 5 | 1,703 |
| Texas | 183 | 328 | 5 | 1,703 |
| Washington | 217 | 446 | 5 | 1,703 |
| Wiscosin | 262 | 530 | 5 | 1,703 |
| Citeseer | 3,312 | 4,732 | 6 | 3,703 |
| Cora | 2,708 | 5,429 | 7 | 1,433 |
| Pubmed | 19,717 | 44,338 | 3 | 500 |

**Baseline methods.** We compare the proposed methods against the 10 state-of-the-art network representation methods. As we mentioned in the method part, our ArmGAN model has two versions, i.e., 1) the discrimination process using direct mapping representation mechanism and 2) the discrimination process using mutual information representation mechanism, which are called as $\text{ArmGAN}_{\text{d}}$ and $\text{ArmGAN}_{\text{m}}$, respectively.

1) DeepWalk [1] is a unsupervised method that adopts random walk and Skip-Gram to learn node representation.
2) node2vec [14] is a variant of DeepWalk and designs a biased random walk to learn node representation.

1. https://linqs.soe.ucsc.edu/data

3) LINE [2] is also a popular unsupervised method that preserves the first-order and second-order proximity among nodes in the graph.
4) GraRep [3] adopts matrix factorization method to learn node representation.
5) AANE [19] models and incorporates node attribute proximity into network embedding.
6) TriDNR [18] learns node representations by coupling multiple neural network models to jointly exploit the network structure, node-content correlation, and label-content correspondence.
7) SNE [17] preserves the structure proximity and attribute proximity of social networks and generates nodes embedding.
8) VGAE [22] is the variational graph autoencoder for graph embedding with both topological and content information.
9) ARVGA [21] uses adversarially regularized variational autoencoder algorithm to learn the embedding.
10) DGI [13] is a recently proposed unsupervised GNN method which learns node representation by maximizing the mutual information between patch representation and global representation.

**Parameter settings.** For all the algorithms compared, the final embedding dimension is set to 128 for WebKB dataset, and 256 for Cora, Citeseer and Pubmed as the citation networks are much larger than the webpage networks. For the hyperparameters of our model, we set the $\alpha_1$, $\alpha_2$, $\alpha_3$, $\beta_1$, $\beta_2$, $\beta_3$ to 1, 1, 1, 0,01, 0.01, 0.01 respectively, which are often stable and give good results. To ensure fairness, the parameters of the baseline methods were set as what were used by their authors. In our approach ArmGAN, we set the number of samples $s$ as the number of nodes as used in most mutual information estimation methods [13], [35]. For both encoder and negative encoder, we use the classic two-layer GCN and use $\mathrm{ReLU}(\cdot)$ as the activation function. For the representation mechanism discriminator and neural networks of mutual information regularity and the negative mutual information regularity, we use three-layer fully connected neural network and use $\mathrm{ReLU}(\cdot)$ as the activation function. We apply the pytorch deep learning tools and use Adam optimizer to learn the model for 400 epochs. Both the learning rate and discriminator learning rate are set to 0.001.

## 5.2 Node classification

In node classification, each node is assigned with one label. A fraction of nodes and their labels are observed, and the labels of the remaining nodes need to be predicted. Therefore the performance of node classification can reveal the distinguishability of nodes under different network representation learning methods. After getting the network embedding, we adopt the LibSVM and LibLINEAR software packages in Weka to classify these nodes with ground-truth. For each network, we used 10-fold cross-validation, and report accuracy (AC) [42] with mean value and standard deviation.

The experimental results are presented in Table 2. The results show that our proposed ArmGAN with mutual information mechanism ($\mathrm{ArmGAN_m}$) outperforms all the baselines on all datasets. On average, $\mathrm{ArmGAN_m}$ outperformed the state-of-the-art network representation method DGI in node classification by 8.37% using LibSVM and by 6.34% using LibLINEAR. In addition, $\mathrm{ArmGAN_m}$ outperformed the GAN based method ARVGA (which matches the distribution of representation to Gaussian distribution) in node classification by 8.64% using LibSVM and by 6.06% using LibLINEAR on seven networks on average. This also validates applying the adversarial learning strategy on the representation mechanism is better than applying the adversarial strategy on representation results.

TABLE 2: Comparison on node classification with mean value and standard deviation in terms of AC (%). The best result is marked in bold and the second best is underlined.

| Packages | Methods | Cornell | Texas | Washington | Wisconsin | Cora | Citeseer | Pubmed |
|---|---|---|---|---|---|---|---|---|
| LibSVM | DeepWalk | 38.97±0.17 | 49.18 ±0.18 | 55.30±0.14 | 49.24±0.22 | 82.57±0.01 | 52.52±0.06 | 78.79±0.02 |
| | node2vec | 35.90±0.19 | 50.27±0.18 | 47.47±0.14 | 46.56±0.10 | 79.98±0.05 | 61.63±0.13 | 80.30±0.22 |
| | LINE | 43.59±0.06 | 56.28±0.12 | 59.91±0.11 | 54.58±0.15 | 30.20±0.05 | 41.07±0.11 | 75.47±0.008 |
| | GraRep | 53.33±0.15 | 62.68±0.11 | 52.07±0.19 | 59.16±0.24 | 73.41±0.22 | 54.28±0.24 | 80.64±0.19 |
| | AANE | 51.80±0.17 | 56.28±0.11 | 64.06±0.09 | 43.13±0.07 | 30.20±0.10 | 24.70±0.03 | 78.63±0.10 |
| | TriDNR | 37.95±0.16 | 48.09±0.10 | 47.01±0.15 | 40.46±0.20 | 43.27±0.09 | 54.47±0.08 | 79.07±0.12 |
| | SNE | 48.21±0.29 | 57.92±0.18 | 54.38±0.11 | 59.54±0.15 | 49.00±0.01 | 44.74±0.04 | 78.37±0.09 |
| | VGAE | 45.12±0.09 | 55.00±0.12 | 54.38±0.06 | 53.82±0.12 | 81.05±0.19 | 65.97±0.05 | 83.42±0.11 |
| | ARVGA | 42.56±0.12 | 56.28±0.16 | 58.99±0.10 | 49.26±0.19 | 80.42±0.06 | 65.10±0.05 | 80.64±0.008 |
| | DGI | 42.56±0.23 | 56.28±0.04 | 47.47±0.04 | 45.42±0.11 | 84.15±0.02 | 71.84±0.06 | 79.19±0.007 |
| | $\mathrm{ArmGAN_m}$ | **54.35**±0.05 | **64.48**±0.09 | **65.89**±0.08 | **60.69**±0.05 | **88.95**±0.04 | **73.07**±0.05 | **86.31**±0.004 |
| | $\mathrm{ArmGAN_d}$ | 51.28±0.08 | 60.65±0.07 | 61.75±0.09 | 59.16±0.09 | 88.77±0.02 | 72.88±0.04 | 85.99±0.007 |
| LibLINEAR | DeepWalk | 38.46±0.03 | 48.09±0.17 | 53.92±0.14 | 49.62±0.07 | 82.04±0.02 | 48.42±0.05 | 78.36±0.03 |
| | node2vec | 37.95±0.19 | 50.27±0.16 | 45.62±0.14 | 46.95±0.17 | 80.79±0.11 | 52.44±0.17 | 80.08±0.01 |
| | LINE | 44.10±0.17 | 53.39±0.16 | 56.22±0.08 | 54.96±0.18 | 50.25±0.01 | 40.56±0.08 | 74.92±0.02 |
| | GraRep | 47.17±0.08 | 59.40±0.06 | 51.15±0.09 | 60.31±0.23 | 79.83±0.13 | 53.61±0.28 | 80.37±0.15 |
| | AANE | 41.54±0.17 | 53.01±0.09 | 61.75±0.10 | 38.93±0.05 | 27.03±0.04 | 22.24±0.02 | 77.99±0.09 |
| | TriDNR | 34.87±0.13 | 42.08±0.08 | 43.32±0.17 | 41.60±0.23 | 53.39±0.08 | 52.91±0.09 | 78.40±0.008 |
| | SNE | 45.64±0.10 | 59.02±0.16 | 55.76±0.29 | 59.92±0.25 | 54.46±0.03 | 44.35±0.07 | 77.20±0.04 |
| | VGAE | 45.64±0.10 | 51.91±0.05 | 54.84±0.03 | 54.49±0.11 | 79.13±0.05 | 69.25±0.05 | 83.81±0.02 |
| | ARVGA | 41.54±0.11 | 59.02±0.06 | 60.37±0.09 | 56.11±0.18 | 81.24±0.06 | 66.71±0.03 | 80.59±0.007 |
| | DGI | 43.08±0.13 | 56.28±0.09 | 52.07±0.03 | 48.47±0.18 | 85.22±0.02 | 73.85±0.05 | 84.22±0.09 |
| | $\mathrm{ArmGAN_m}$ | **51.79**±0.09 | **62.30**±0.07 | **62.21**±0.09 | **61.07**±0.07 | **88.55**±0.09 | **75.99**±0.03 | **86.10**±0.003 |
| | $\mathrm{ArmGAN_d}$ | 50.76±0.09 | 61.20±0.04 | 60.82±0.08 | 59.54±0.12 | 87.92±0.02 | 75.02±0.06 | 85.83±0.003 |

Although the ArmGAN with direct mapping representation mechanism ($\mathrm{ArmGAN_d}$) is not as good as $\mathrm{ArmGAN_m}$, $\mathrm{ArmGAN_d}$ outperforms all the baselines on 3 out of 7 datasets in LibSVM and 5 out of 7 datasets in LibLINEAR. As mentioned in the Section 4.4, $\mathrm{ArmGAN_m}$ has two mutual information constrain terms: one is based on DV-representation and the other is based on f-divergence representation implemented by adversarial representation mechanism learning. This can further help the encoder to extract more representative information which may be more suitable for the node classification task. Therefore, $\mathrm{ArmGAN_m}$ can perform better than $\mathrm{ArmGAN_d}$ in node classification.

## 5.3 Node clustering

In node clustering, we aim to assign distinct cluster to each node with no supervision. To conduct the experiment, we first train all the algorithms to obtain the network embedding. After that, we applied $k$-means algorithm to the embedding results of nodes to cluster them into different classes. For node clustering (a.k.a., community detection), besides accuracy [42], we also use normalized mutual information (NMI) [43] as an additional accuracy metric since NMI has been more often used in node clustering.

The experimental results are shown in Table 3. As shown, our proposed $\mathrm{ArmGAN_m}$ performs the best on 6 out of 7 datasets in terms of both AC and NMI. $\mathrm{ArmGAN_d}$ performs the best on 1 out of 7 networks in terms of both AC and

TABLE 3: Comparison on node clustering in terms of AC(%) and NMI(%).

| Metrics (%) | Methods | Cornell | Texas | Washington | Wisconsin | Cora | Citeseer | Pubmed |
|---|---|---|---|---|---|---|---|---|
| AC | DeepWalk | 36.05 | 46.72 | 40.76 | 38.76 | 45.61 | 36.21 | 64.84 |
| | node2vec | 33.85 | 47.54 | 37.33 | 49.62 | 56.30 | 40.76 | 65.56 |
| | LINE | 39.49 | 53.38 | 52.68 | 45.43 | 30.72 | 25.01 | 43.11 |
| | GraRep | 31.79 | 36.72 | 31.36 | 33.24 | 48.29 | 31.20 | 54.43 |
| | AANE | 37.28 | 30.49 | 41.57 | 30.53 | 18.51 | 21.76 | 34.55 |
| | TriDNR | 38.21 | 47.54 | 43.59 | 43.70 | 31.56 | 34.44 | 59.29 |
| | SNE | 41.08 | 41.53 | 48.80 | 55.30 | 39.44 | 31.17 | 65.13 |
| | VGAE | 36.72 | 48.35 | 43.73 | 43.28 | 57.06 | 53.46 | 58.64 |
| | ARVGA | 38.21 | 41.48 | 43.66 | 42.81 | 64.08 | 43.50 | 58.76 |
| | DGI | 38.87 | 53.55 | 48.75 | 44.27 | 71.93 | 68.76 | 65.21 |
| | $\text{ArmGAN}_\text{m}$ | **54.36** | **60.66** | **60.83** | 56.49 | **76.11** | **70.18** | **71.55** |
| | $\text{ArmGAN}_\text{d}$ | 48.20 | 56.28 | 60.82 | **58.01** | 74.04 | 67.77 | 70.96 |
| NMI | DeepWalk | 7.06 | 6.16 | 5.66 | 7.65 | 31.51 | 10.58 | 25.55 |
| | node2vec | 6.65 | 4.49 | 2.94 | 7.86 | 42.02 | 12.99 | 25.02 |
| | LINE | 9.27 | 18.16 | 18.95 | 9.39 | 10.13 | 5.62 | 7.17 |
| | GraRep | 8.80 | 12.43 | 5.18 | 8.02 | 35.46 | 9.61 | 17.76 |
| | AANE | 9.55 | 3.52 | 13.19 | 2.86 | 0.40 | 1.19 | 0.01 |
| | TriDNR | 7.20 | 4.32 | 8.10 | 6.60 | 12.19 | 9.59 | 19.28 |
| | SNE | 11.11 | 12.63 | 17.43 | 18.94 | 16.28 | 7.31 | 25.61 |
| | VGAE | 7.77 | 8.52 | 9.03 | 9.31 | 42.92 | 27.93 | 17.83 |
| | ARVGE | 10.26 | 7.28 | 12.60 | 11.92 | 44.95 | 22.72 | 18.40 |
| | DGI | 13.99 | 14.05 | 13.93 | 13.22 | 56.52 | 44.32 | 25.77 |
| | $\text{ArmGAN}_\text{m}$ | **21.07** | **18.42** | **25.91** | 19.72 | **58.43** | **44.56** | **33.70** |
| | $\text{ArmGAN}_\text{d}$ | 15.24 | 13.55 | 25.82 | **19.94** | 58.22 | 42.89 | 32.91 |

TABLE 4: Comparison on link prediction in terms of AUC and AP.

| Metrics (%) | Methods | Cornell | Texas | Washington | Wisconsin | Cora | Citeseer | Pubmed |
|---|---|---|---|---|---|---|---|---|
| AUC | DeepWalk | 52.34 | 49.15 | 54.44 | 62.32 | 83.10 | 80.50 | 84.40 |
| | node2vec | 70.99 | 55.30 | 56.63 | 69.43 | 77.39 | 67.31 | 78.03 |
| | LINE | 63.05 | 50.52 | 57.51 | 57.29 | 63.27 | 57.88 | 66.02 |
| | GraRep | 43.87 | 44.42 | 45.57 | 45.43 | 55.31 | 69.03 | 46.33 |
| | AANE | 52.38 | 46.68 | 45.86 | 53.88 | 50.64 | 52.27 | 50.30 |
| | TriDNR | 50.14 | 48.32 | 49.98 | 58.53 | 81.25 | 76.78 | 86.00 |
| | SNE | 52.99 | 51.57 | 49.89 | 54.07 | 84.68 | 83.09 | 75.52 |
| | VGAE | 82.94 | 80.88 | 75.54 | 83.30 | 92.38 | 91.44 | 94.46 |
| | ARVGA | 83.92 | 76.45 | 77.00 | 68.78 | 92.80 | 92.41 | 96.11 |
| | DGI | 85.92 | 86.50 | 79.13 | 86.44 | 92.96 | 94.95 | 95.80 |
| | $\text{ArmGAN}_\text{m}$ | 88.56 | 89.16 | 80.06 | 89.64 | 94.29 | 95.46 | 95.64 |
| | $\text{ArmGAN}_\text{d}$ | **91.32** | **89.70** | **81.47** | **91.01** | **94.99** | **96.81** | **96.70** |
| AP | DeepWalk | 63.21 | 51.87 | 55.28 | 69.21 | 85.00 | 83.60 | 84.10 |
| | node2vec | 72.63 | 57.31 | 68.39 | 70.75 | 74.61 | 68.09 | 76.97 |
| | LINE | 64.42 | 53.95 | 60.07 | 58.53 | 70.87 | 66.13 | 69.61 |
| | GraRep | 47.42 | 47.35 | 47.89 | 47.73 | 52.89 | 64.11 | 48.26 |
| | AANE | 56.92 | 50.84 | 50.99 | 54.88 | 51.80 | 52.37 | 54.34 |
| | TriDNR | 55.44 | 52.65 | 54.32 | 66.71 | 85.65 | 80.93 | 85.64 |
| | SNE | 51.51 | 50.98 | 49.51 | 52.23 | 76.85 | 75.30 | 78.73 |
| | VGAE | 85.99 | 85.71 | 80.55 | 85.68 | 93.51 | 92.66 | 94.86 |
| | ARVGA | 85.54 | 81.06 | 83.66 | 76.25 | 92.99 | 93.48 | 96.29 |
| | DGI | 87.27 | 88.90 | 81.42 | 87.60 | 92.18 | 95.05 | 95.38 |
| | $\text{ArmGAN}_\text{m}$ | 91.90 | 92.09 | **86.25** | 90.03 | 94.55 | 96.13 | 95.64 |
| | $\text{ArmGAN}_\text{d}$ | **92.55** | **93.23** | 85.98 | **92.97** | **95.26** | **96.79** | **96.34** |

NMI and performs the second best in 5 out of 7 datasets in terms of AC and 4 out of 7 datasets in terms of NMI. On average, $\text{ArmGAN}_\text{m}$ outperforms DGI which is well-known for its high accuracy in node clustering by 8.40% in terms of AC and 5.71% in terms of NMI on all seven networks. In addition, $\text{ArmGAN}_\text{m}$ performs better than the classical GAN based method ARVGA by 16.81% in terms of AC and 13.38% in terms of NMI on all 7 networks on average. The superior performance of our ArmGAN over the state-of-the-art methods validates the effectiveness of the new approach, and further demonstrates the superiority of adversarial representation mechanism over adversarial representation results.

### 5.4 Link Prediction

In the task of link prediction, our goal is to predict whether there exists an edge between two give nodes. This task shows the performance of edge predictability of different network representation learning methods. All methods are trained on an incomplete version of these datasets where some of the edges have been removed, while all node attributes are kept. After training, we obtain the representation vectors for all nodes and use inner product method to predict the probability of edge existence for a given node pair. The test set consists of the 10% removed edges (node pairs) in the original network as the positive samples and randomly selected disconnected node pairs as negative samples, where the number of positive and negative samples are the same. The validation set contains 5% edges, which is used for fine tuning the hyperparameters. The remaining 85% edges of the original network are taken as the training set. For link prediction experiments, we report the area under the ROC curve (AUC) [22] and average precision (AP) [22] scores for each method on the test set. We conduct each experiment 10 times and report the mean values as the final scores.

The experimental results on link prediction are shown in Table 4. As shown, $\text{ArmGAN}_\text{d}$ performs the best on all 7 datasets in terms of AUC and 6 out of 7 datasets in terms of AP. $\text{ArmGAN}_\text{m}$ performs the best on 1 out 7 datasets in terms of AP and performs the second best on 6 out of 7 datasets in terms of AUC and 5 out of 7 datasets in terms of AP. For all 7 datasets, $\text{ArmGAN}_\text{d}$ is on average 2.90% and 7.79% more accurate in terms of AUC and 3.61% and 6.26% more accurate in terms of AP than DGI and ARVGA, respectively.

It is worth noting that $\text{ArmGAN}_\text{d}$ performs better than $\text{ArmGAN}_\text{m}$ in link prediction in almost all dataset, which is different from the experiments on node classification and node clustering. This is because $\text{ArmGAN}_\text{d}$ can capture not only the information of local neighbors but also that of remote neighbors, which is more conducive for measuring the similarity of two nodes and further doing link prediction. Therefore, $\text{ArmGAN}_\text{d}$ has a better performance in the task of link prediction.

### 5.5 Visualization

To further illustrate that the embedding from our method is an accurate representation, we also visualize the embedding results of all methods in the Cora dataset as an example. We use the t-SNE [7] tool to down scale the result of embedding representation to two dimensions and draw a color for each categorical label. Therefore a desirable visualization result refers to that nodes belonging to the same category (in same color) should be close to each other. The result of visualization is given in Fig. 2.

As shown in Fig. 2, the results of DeepWalk, LINE, node2vec and AANE are less satisfactory since the points of different categories are mixed with each other. The results of VGAE, ARVGA, DGI are relatively better as the clusters of points with the same color can be observed to form segmented groups, but the borders are not very clear. We observe that our proposed $\text{ArmGAN}_\text{m}$ and $\text{ArmGAN}_\text{d}$ can make relatively clear separation between different categories. In other words, nodes in the same color are roughly gathered together. This visualization validates that our model can obtain a better representation, and further demonstrates the superiority of our proposed model.

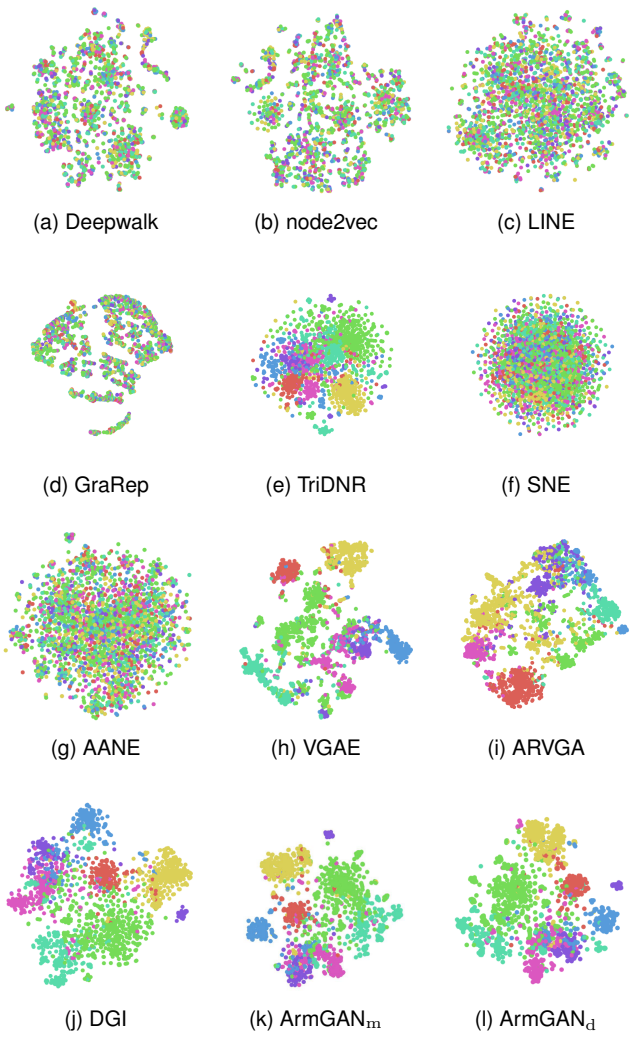



Fig. 2: Visualization of different network embedding methods on the Cora dataset

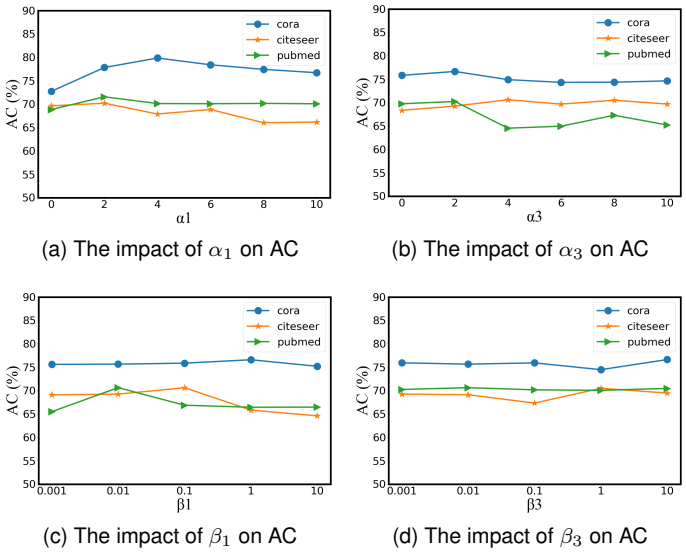



Fig. 3: The impacts of hyperparameters $\alpha_1$, $\alpha_3$, $\beta_1$, $\beta_3$ of ArmGAN$_{\mathrm{m}}$ in node clustering performance on Cora, Citeseer and Pubmed datasets.

### 5.6 Parameter Analysis

In this section, we will analyze the impact of the hyperparameters of our ArmGAN model using node clustering task as an example. The role of hyperparameters of ArmGAN$_{\mathrm{m}}$ and ArmGAN$_{\mathrm{d}}$ are similar. For simplicity, here we only analyze the hyperparameters of ArmGAN$_{\mathrm{m}}$. The hypterparameters of ArmGAN$_{\mathrm{m}}$ are $\alpha_1$, $\alpha_3$, $\beta_1$ and $\beta_3$, where $\alpha_1$ and $\alpha_3$ weight the reconstruction losses for the autoencoder with mutual information regularity and negative sample generator in the objective function. $\beta_1$ and $\beta_3$ weight mutual information regularity and negative mutual information regularity in the objective function. To study the impact of the individual parameter on the clustering result, we just vary the target parameter, with other three parameters are fixed. The results are shown in Fig. 3. In each figure, X axis denotes different values of hyperparameter and Y axis represents the clustering accuracy, i.e., AC, and the results of NMI which is similar with AC are provided in Appendix.

As is shown in Fig. 3, the AC scores on these datasets are rather steady with the changing values of the hyperparameters. This indicates that these hyperparameters have little impact on the performance of the proposed approach. The results show that our proposed method is not so sensitive to changes of the hyperparameters, and further demonstrate the robustness of our proposed ArmGAN.

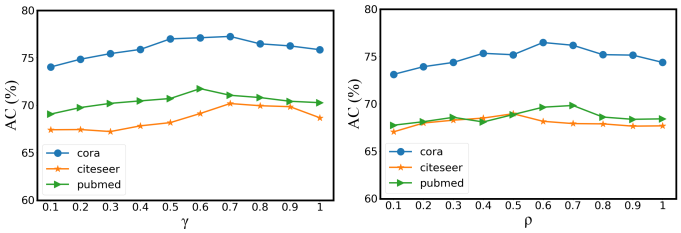
### 5.7 Perturbation Strategy

In this section, we investigate how different perturbation strategies and different degrees of perturbation influence the performance of the proposed approach. Here we take node clustering as an example. As the perturbation for ArmGAN$_{\mathrm{m}}$ and ArmGAN$_{\mathrm{d}}$ are similar, here we only use ArmGAN$_{\mathrm{m}}$ for illustration.

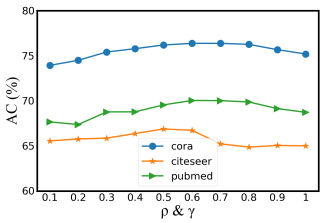
First, we test the impact of perturbation on network topology, attribute information and their combination, respectively. We first consider perturbation on attribute information, which is implemented by randomly corrupting the attributes $X$ via row-vise shuffling. In this case, while the fake network has the same topology with the original one, they have different features. We called this perturbation strategy as ArmGAN$_{\mathrm{X}}$. Then we consider perturbation on network topology which preserves the original features $X$ but adds or removes edges from the adjacency matrix ($A' \neq A$) with a certain probability. In specific, we first generate a random graph $M$ which has the same nodes as the original graph and the edge existence probability between any node pair is defined as $\rho$ (here we let $\rho$ be the inverse of the number of nodes). We then obtain the corrupted adjacency matrix $A' = A \oplus M$ where $\oplus$ is the XOR (exclusive OR) operation. This strategy produces a fake network with the same features, but different connectivity. We called this perturbation strategy as ArmGAN$_{\mathrm{A}}$. At last, we consider simultaneous feature shuffling ($X' \neq X$) and adjacency matrix perturbation ($A' \neq A$), which are implemented using the ways described above. We called it ArmGAN$_{\mathrm{XA}}$. The results for using these three perturbation strategies are shown in Table 5. The results show that ArmGAN using

TABLE 5: Node clustering performance with different perturbation strategies.

| Metrics (%) | Methods | Cornell | Texas | Washington | Wisconsin | Cora | Citeseer | Pubmed |
|---|---|---|---|---|---|---|---|---|
| AC | $\text{ArmGAN}_{\text{X}}$ | **54.36** | **60.66** | **60.83** | **56.49** | **76.11** | **70.18** | **71.55** |
| | $\text{ArmGAN}_{\text{A}}$ | 50.76 | 59.56 | 57.14 | 52.29 | 74.85 | 67.41 | 68.77 |
| | $\text{ArmGAN}_{\text{XA}}$ | 45.12 | 60.65 | 58.98 | 52.29 | 74.66 | 65.94 | 70.43 |
| NMI | $\text{ArmGAN}_{\text{X}}$ | **21.07** | **18.42** | **25.91** | **19.72** | **58.43** | **44.56** | **33.70** |
| | $\text{ArmGAN}_{\text{A}}$ | 15.14 | 15.56 | 18.34 | 15.86 | 57.08 | 41.26 | 30.54 |
| | $\text{ArmGAN}_{\text{XA}}$ | 14.60 | 17.86 | 19.77 | 17.73 | 58.17 | 41.44 | 32.91 |

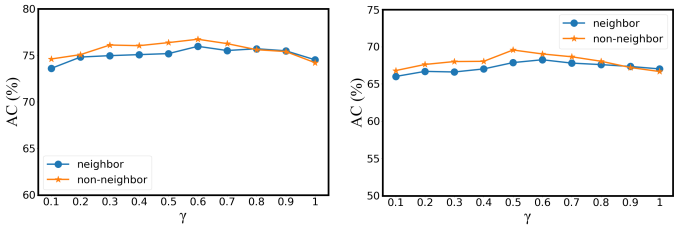



(a) $\text{ArmGAN}_{\text{X}}$ (b) $\text{ArmGAN}_{\text{A}}$

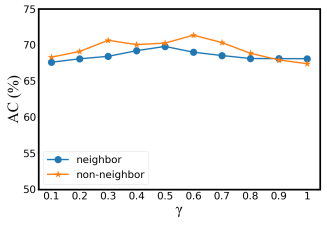



(c) $\text{ArmGAN}_{\text{XA}}$

Fig. 4: The impacts of different degrees of perturbation in node clustering. $\gamma$ represents the ratio of the perturbed attributes and $\rho$ represents the reverse probability of each element in adjacency matrix $A$.




(a) Cora (b) Citeseer




(c) Pubmed

Fig. 5: The impacts of shuffling neighboring nodes or non-neighboring nodes in node clustering on Cora, Citeseer and Pubmed.

the first strategy performs better than using the other two strategies on these datasets. This may be because network topology is more reliable than attribute information for network related tasks. Thus, using real network topology in the "fake" network can provide more useful information for network embedding.

Then, we investigate the influence of different degrees of perturbation under each strategy. Specifically, for perturbation on attribute information, we vary the proportion of perturbed nodes from 10% to 100% with increment of 10%. For perturbation on network topology, we vary the probability $\rho$ from 0.1 to 1 with increment of 0.1 (the probability $\rho$ means that each element in adjacency matrix $A$ is reversed with probability $\rho$, i.e., if there is an edge between two nodes, they will be disconnected with probability $\rho$, and vice versa). For perturbation on both attribute information and network topology, the perturbation ratio of attributes and the reversed probability change simultaneously from 10% (0.1) to 100% (1). In order to reduce the influence of randomness, we generate 20 network instances randomly for each degree of perturbation and calculate the mean of the performance on these networks. The results are shown in Fig. 4. As shown, in the beginning, when the degree of perturbation is small, the results are relatively poor and they gradually become better as perturbation degree increases. And then, the results slightly drop, as the degree continues increasing. One possible explanation for the above observation is that, when the degree of perturbation is small (such as close to 0), the fake network is very much similar to the original network. It is difficult for the discriminator to learn. Conversely, when the degree of perturbation is very large (such as close to 1), the fake network is very much different from the original network. Thus the model may not necessarily learn the meaningful representation to distinguish the two. The results show that for perturbation on attribute information, the algorithm performs the best when the perturbation ratio is between 60% and 70% and for perturbation on network topology, the algorithm performs the best when the probability $\rho$ is between 0.5 and 0.6. For perturbation on both of them, the algorithm performs the best when they are between 60% (0.6) and 70% (0.7).

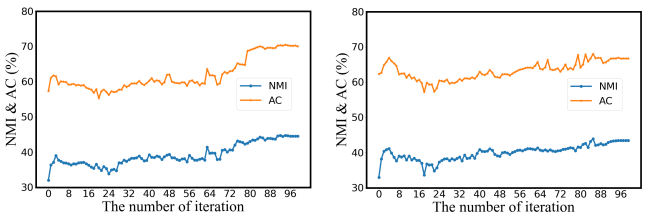
Furthermore, for attribute shuffling strategy, as shuffling attributes of neighboring nodes or non-neighboring nodes may be different, we conduct additional experiments to validate this. We randomly select part of nodes (the percentage of selected nodes varies from 0.1 to 1 which is similar to the approach discussed above). For each selected node, we swap its attribute with one of its neighbors or non-neighbors. We also generate 20 network instances for each specific percentage value and report the mean performance result. The results are shown in Fig. 5. As shown, in most cases shuffling non-neighboring nodes can achieve better performance than shuffling neighboring nodes.

### 5.8 Convergence Analysis

Finally, we investigate the convergence of ArmGAN. The experimental result is shown in Fig. 6, where the X axis represents the iteration numbers and Y axis represents the NMI and AC scores evaluated in node clustering task. Here we choose Citeseer, which is relatively large among all the 7 networks, as the dataset for convergence analysis experiments. As shown in Fig. 6, the AC and NMI have similar changing trends. They fluctuate in the very beginning, and gradually increase as training proceeds. In the end, when the iteration number is close to 100, the AC and NMI converge to a steady state.

## 6 Conclusion and Discussion

In this paper, we propose a new generative adversarial framework for network embedding called ArmGAN, which uses adversarial learning strategy on the representation




(a) ArmGAN$_{\mathrm{m}}$ (b) ArmGAN$_{\mathrm{d}}$

Fig. 6: Results of convergency analysis in node clustering on Citeseer dataset

mechanism rather than on embedding results so as to better utilize the essential advantage of GAN on network embedding. The new generative adversarial framework contains three players: the autoencoder with mutual information regularity which hosts the positive representation mechanism, the negative sample generator which hosts the negative representation mechanism, and the discriminator. This is different from the existing GAN framework for network embedding that includes only two players, and only one player realizes the representation mechanism. Furthermore, the goal of the autoencoder with mutual information regularity is consistent with the goal of discriminator, i.e., helping the discriminator to identify its samples as real representation mechanism, while the goal of the negative sample generator is opposite to that of the discriminator, i.e., deceiving the discriminator. The method proposed is evaluated on 7 real datasets with different scales for different network analysis tasks. Experimental results show that the new method significantly outperforms the state-of-the-art methods including a typical GAN based method and a mutual information based method.

In real world, for some networks, their topological information and attributes information are inconsistent. We plan to use multi-channel graph convolutional networks and attention mechanism to extend our method so as to automatically learn and combine the reliable information of topology and attributes information and form effective and robust representation mechanism in the future.

## Acknowledgment

This work was supported in part by the Natural Science Foundation of China (61876128, 61832014, 61772361), the George Washington University Facilitating Fund (UFF) FY21, and the NSF under grants III-1763325, III-1909323, and SaTC-1930941.

## References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[2] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077.

[3] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[4] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2663–2671.

[5] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011, pp. 1169–1174.

[6] J. Tang, C. Aggarwal, and H. Liu, "Node classification in signed social networks," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, 2016, pp. 54–62.

[7] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[8] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, 2014, pp. 283–292.

[9] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, "Learning deep representations for graph clustering." in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014, pp. 1293–1299.

[10] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-Margin DeepWalk: Discriminative learning of network representation," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 3889–3895.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[12] P. Zhang, B. Chai, J. Zhang, and W. Li, "Semi-supervised representation learning method for attributed networks," *Computer Engineering and Applications*, 2019.

[13] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[14] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[15] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding." in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 203–209.

[16] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[17] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[18] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-Party deep network representation," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 1895–1901.

[19] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017, pp. 633–641.

[20] I. Goodfellow, J. PougetAbadie, M. Mirza, B. Xu, D. WardeFarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[21] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2609–2615.

[22] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proceedings of the 5th International Conference on Learning Representations*, 2016.

[23] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2475–2487, 2019.

[24] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Proceedings of the 32nd Association for the Advancement of Artificial Intelligence*, 2018, pp. 2167–2174.

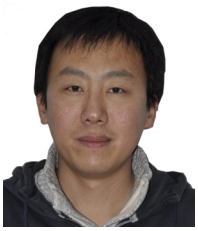
[25] D. Fu, Z. Xu, B. Li, H. Tong, and J. He, "A view-adversarial framework for multi-view network embedding," in *Proceedings of the 29th International Conference on Information and Knowledge Management*, 2020, pp. 2025–2028.

[26] H. Hong, X. Li, and M. Wang, "GANE: A generative adversarial network embedding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2325–2335, 2019.
[27] 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 *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 2508–2515.
[28] S. Zhu, J. Li, H. Peng, S. Wang, P. S. Yu, and L. He, "Adversarial directed graph embedding," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.
[29] H. Gao, J. Pei, and H. Huang, "ProGAN: Network embedding via proximity generative adversarial network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1308–1316.
[30] L. Yang, Y. Wang, J. Gu, C. Wang, X. Cao, and Y. Guo, "JANE: Jointly adversarial network embedding," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 1381–1387.
[31] M. Khajehnejad, A. A. Rezaei, M. Babaei, J. Hoffmann, M. Jalili, and A. Weller, "Adversarial graph embeddings for fair influence maximization over social networks," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 4306–4312.
[32] S. Bandyopadhyay, L. N, S. V. Vivek, and M. N. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 25–33.
[33] Q. Liu, C. Long, J. Zhang, M. Xu, and P. Lv, "TriATNE: Tripartite adversarial training for network embeddings," *IEEE Transactions on Cybernetics*, 2021.
[34] C. Wang, W. Shi, L. Huang, K. Lin, D. Huang, and P. S. Yu, "Node pair information preserving network embedding based on adversarial networks," *IEEE Transactions on Cybernetics*, 2020.
[35] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.
[36] M. D. Donsker and S. S. Varadhan, "Asymptotic evaluation of certain markov process expectations for large time, i," *Communications on Pure and Applied Mathematics*, vol. 28, no. 1, pp. 1–47, 1975.
[37] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.
[38] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Estimating divergence functionals and the likelihood ratio by convex risk minimization," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5847–5861, 2010.
[39] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "MINE: Mutual information neural estimation," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.
[40] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proceedings of the 5th International Conference on Learning Representations*, 2016.
[41] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.
[42] H. Liu, Z. Wu, X. Li, D. Cai, and T. S. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1299–1311, 2012.
[43] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 09, p. P09008, 2005.

**Dongxiao He** received her Ph.D. degrees in computer science from Jilin University, China, in 2014. She is an Associate Professor in the College of Intelligence and Computing, Tianjin University, China. She has published 16 top-tier journal and conference papers. Her current research interests include data mining and network analysis. She serves as senior program committees for conferences AAAI and IJCAI.

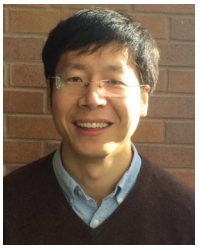
**Tao Wang** is currently a master student of College of Intelligence and Computing from Tianjin University, Tianjin, China. His research interests mainly related to community detection and social network analysis.

**Lu Zhai** received the M.S. degree in College of Intelligence and Computing, Tianjin University, China. Her research interests are mainly related to community detection, social network analysis and machine learning.

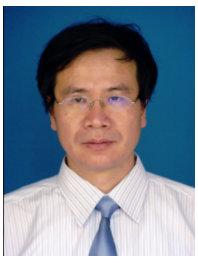


**Di Jin** received his Ph.D. degrees in computer science from Jilin University, China, in 2012. He is currently an Associate Professor in the College of Intelligence and Computing, Tianjin University, China. He got WWW-21 Best Paper Award Runner-up and published more than 20 papers in top-tier journals and conferences. His research interests include social network analysis and data mining. He serves as senior program committees for conferences AAAI and IJCAI.

**Liang Yang** received the Ph.D. degree in computer science from the Institute of Information Engineering, Chinese Academy of Sciences, China, in 2016. He is an Assistant Professor in the School of Artificial Intelligence, Hebei University of Technology, China. His current research interests include network analysis, low-rank modeling, and data mining. He serves as senior program committees for conference IJCAI.



**Yuxiao Huang** is an Assistant Professor of Data Science in the Columbian College of Arts & Sciences at George Washington University (U.S.A.). His research interest is Machine Learning. His work has been published on more than 20 international journals and conferences. He serves as as program committee for conferences including NeurIPS, ICML, AAAI and IJCAI. He earned a PhD in Computer Science from Jilin University (China).



**Zhiyong Feng** received his Ph.D. degree in Tianjin University. He is currently a full professor in the School of Computer Science and Technology, Tianjin University, China. He is the author of one book, more than 130 articles, and 39 patents. His research interests include knowledge engineering, services computing, and security software engineering. He is a member of the IEEE and ACM.

**Philip S. Yu** is a Distinguished Professor in Computer Science at the University of Illinois at Chicago. His research interest is on big data, including data mining, data stream, database and privacy. He has published more than 920 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and of the IEEE. He is on the steering committee of IEEE Conference on Data Mining and was a member of the IEEE Data Engineering steering committee. Dr. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University.