IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 10, 2019, accepted February 17, 2019, date of publication February 21, 2019, date of current version March 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900662

Network Embedding via Community Based Variational Autoencoder

WEI SHI[1,2], LING HUANG[1,2], CHANG-DONG WANG[1,2], JUAN-HUI LI[1,2], YONG TANG[3], AND CHENGZHOU FU[3]

[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]Guangdong Province Key Laboratory of Computational Science, Guangzhou 510275, China
[3]School of Computer Science, South China Normal University, Guangzhou 510631, China

Corresponding author: Ling Huang (huanglinghl@hotmail.com)

This work was supported in part by the NSFC under Grant 61876193, Grant U181120009, and Grant 61772211, in part by the Guangdong Natural Science Funds for the Distinguished Young Scholar under Grant 2016A030306014, in part by the Tip-top Scientific and Technical Innovative Youth Talents of the Guangdong Special Support Program under Grant 2016TQ03X542, in part by the Science and Technology Planning Project of Guangzhou Municipal Colleges and Universities under Grant 2012A164, and in part by the Guangzhou Innovative Entrepreneurship Project under Grant 2019PT204.

ABSTRACT In recent years, network embedding has attracted more and more attention due to its effectiveness and convenience to compress the network structured data. In this paper, we propose a community-based variational autoencoder (ComVAE) model to learn network embedding, which consists of a community detection module and a deep learning module. In the proposed model, both community information and deep learning techniques are utilized to learn low-dimensional vertex representations. First, community information reveals an implicit relationship between vertices from a global view, which can be a supplement to local information and help to improve the embedding quality. To obtain the community information, community detection algorithms are utilized as a module and the modularization design makes the model more flexible. Second, deep learning techniques can not only integrate and preserve the information from both local and global views efficiently but also strengthen the robustness of vertex representations. To demonstrate the performance of our model, extensive experiments are conducted in four downstream tasks, namely, network reconstruction, node classification, link prediction, and visualization. The experimental results show that our model outperforms the state-of-the-art approaches to real-world datasets.

INDEX TERMS Network embedding, community detection, variational autoencoder.

I. INTRODUCTION

With the development of information technology, more and more network structured data has been generated and recorded, e.g., protein-protein interaction networks, social networks and citation networks. It is vital to mine latent and valuable information from the networks [1]–[4]. However, the sparsity of the traditional network representation makes it difficult to generalize in statistical learning [5]. Therefore, network embedding becomes a necessity for network analysis, which can compress sparse and high-dimensional representations into dense and low-dimensional embeddings in an unsupervised manner [6]. In particular, learning a discriminative embedding space can be helpful for various applications [7], [8]. By maintaining certain characteristics of the original network, the embeddings can be applied to downstream tasks such as network reconstruction [9], node classification [10], [11], link prediction [12] and visualization [13]. In other words, for better performance in various downstream tasks, how to extract the most vital information effectively from the original network is both a target and a challenge for network embedding.

Many efforts have been made in developing network embedding approaches in different ways. For example, DeepWalk [5] and node2vec [14] utilize the random walk and the skipgram model to embed the network. TADW [15] learns representations of the network with rich text information via matrix factorization. With outstanding performance in a wide variety of research fields such as natural language processing [16], [17] and computer vision [18], [19], neural networks have also become popular in unsupervised feature learning [20]. For network embedding, autoencoder [21] is

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

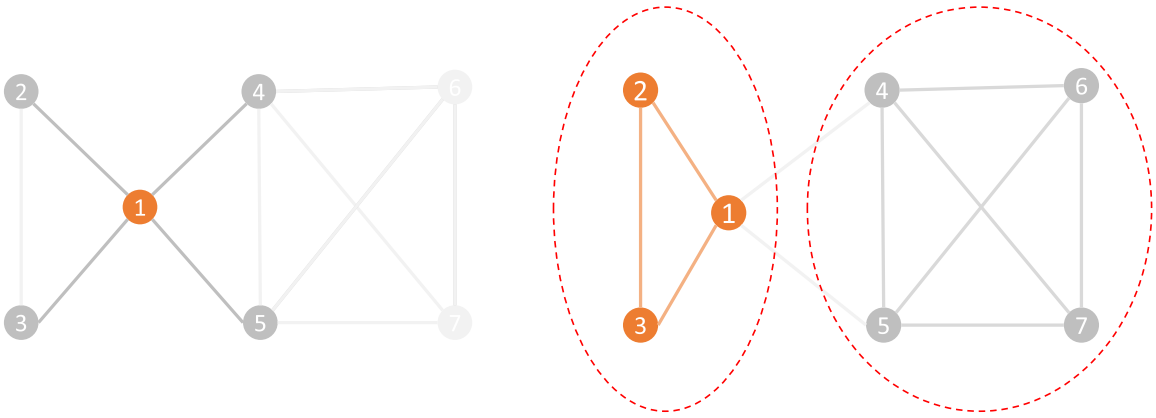


FIGURE 1. An example to show the difference between the first-order proximity information and the community information. (a) From the local view, the relationships between the target vertex 1 and its neighbors are treated equally; first-order proximity information. (b) From the global view, the network can be divided into two communities. According to community information, the relationships between the target vertex 1 and its neighbors in the same community can be treated as stronger ones.

used in SAE [22], SDNE [23] and MVC-DNE [24]. Generative adversarial network is utilized in ANE [25] and GraphGAN [26].

Compared to the random walk models which are regarded as learning representations for linear sequences, the deep learning models are able to capture non-linear information conveyed by the graph [27]. Compared to the matrix factorization based algorithms, the computational complexity of deep learning models mainly depends on the depth of neural network, instead of the eigendecomposition or singular value decomposition of a matrix, which is time and space consuming [28]. In other words, if we design the neural network as shallow as possible under the premise of ensuring performance, deep learning models will be more efficient for less training time. Therefore, in this paper, deep learning techniques are adopted due to its capability of capturing the non-linear information in an effective and efficient way. In particular, we use the conditional variational autoencoder [29], [30] as a module to learn vertex representations for three reasons. Firstly, compared to the traditional autoencoder, the sampling procedure of the variational autoencoder leads to more robust outcomes. Secondly, it is easier to train the variational autoencoder and tune the parameters, since it does not suffer from the problems such as mode collapse. Thirdly, the conditional variational autoencoder can integrate both local information and community information in a unified module, which will be detailed below. Though variational autoencoder has been utilized in [31] and [32] to respectively embed attributed networks and networks with multimodal contents, the proposed model is designed for homogeneous networks without extra information and utilizes variational autoencoder from a different perspective.

Among the existing network embedding algorithms, many of them capture the proximity between vertices only from a local view. For instance, LINE [33] only utilizes the first-order and second-order proximity, while DNGR [27] also utilizes the higher-order proximity at the same time. For these algorithms, the proximities only utilize local information within a fixed number of hops from each vertex, ignoring the information from a global view. To address this problem, community information, which is obtained by analyzing the entire topological structure of the network [34], can be regarded as global information for network embedding, because communities are mainly detected via integrating connection information throughout the whole network. For example, the Fast Unfolding algorithm [35] merges communities via a greedy strategy globally. The Girvan-Newman algorithm [36] utilizes the shortest path between every pair of vertices to calculate betweenness in a global manner. Figure 1 illustrates the difference between the first-order proximity information and the community information. What needs to be emphasized is that we do not insist global information is more useful than local information, but it will improve the embedding quality if both of them can be integrated properly.

Recently, some efforts have been made in utilizing community structure for learning vertex representations, e.g. M-NMF [37], GNE [38] and ANEM [39]. However, most of them redesign the target function based on some community metrics such as modularity [40], which makes them not flexible enough for different networks. Therefore, the proposed model directly utilizes the existing community detection algorithms as a module for two reasons. Firstly, community detection is a hot research topic and many effective algorithms have been proposed. Secondly, the modularization design makes it convenient to choose different community detection algorithms for different networks in order to obtain better performance, which makes the proposed model more flexible.

According to the above discussion, in this paper, we propose a community based variational autoencoder

model (ComVAE) for network embedding, utilizing deep learning techniques and taking community information as a supplement to local information. Firstly, deep learning techniques can not only integrate the information from both local and global views efficiently, but also strengthen the robustness of representations. Secondly, community information reveals implicit relationship between vertices from a global view, which can help to improve the embedding quality as well as local information. To the best of our knowledge, it is the first attempt to embed networks via a variational autoencoder which integrates both local information and community information. Extensive experiments are conducted on two types of real-world datasets to demonstrate the performance of ComVAE via four downstream tasks, namely network reconstruction, node classification, link prediction and visualization. The experimental results confirm the effectiveness of ComVAE and demonstrate the obvious improvements brought by utilizing both community information and deep learning techniques in our model.

The rest of this paper is organized as follows. In Section II we formally illustrate the notations and definitions in this work. In Section III we detail the proposed model ComVAE. In Section IV we introduce how we conduct experiments and show the experimental results. In Section V we conclude this work.

II. NOTATIONS AND DEFINITIONS

In this section, we formally introduce notations and definitions in this work. Given a network $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i, i = 1, 2, \ldots, n\}$ is the set of n vertices and $\mathcal{E} = \{e_{i,j}\}$ is the set of edges, while $e_{i,j} = e_{j,i}$ for undirected network. The goal of network embedding is compressing the representation of each vertex $v_i \in \mathcal{V}$ from the original representation $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|}$ into a new representation $\boldsymbol{y} \in \mathbb{R}^d$, where $d \ll |\mathcal{V}|$. Generally, the adjacency matrix can be obtained from the network, which is represented as $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. The task of network embedding is to reduce dimension and learn a new representation matrix as $Y \in \mathbb{R}^{|\mathcal{V}|\times d}$. In addition to dimension reduction, some proximity between vertices should be maintained during the procedure of network embedding. In the proposed model, both local information and community information are expected to be preserved.

Definition 1 (Local Information): Given a network G, the matrix $X = A$ is denoted as local information because each row vector $\boldsymbol{x}$ in X represents the first-order neighbor relationship of each vertex. With local information, the relationship between interconnected vertices is expected to be maintained after network embedding.

Definition 2 (Community Information): Given a network G, the community information matrix $C \in \mathbb{R}^{|\mathcal{V}|\times k}$, where k is the number of communities, should be obtained through community detection. Each row vector $\boldsymbol{c}$ in matrix C represents the possibility of the corresponding vertex belonging to each community. For non-overlapping community detection, the vector $\boldsymbol{c}$ is a binary and normalized vector. Community information can be regarded as one type of global proximity throughout the network, which is a supplement to local information.

III. THE PROPOSED MODEL

In this section, first of all, an overview of the proposed model will be introduced. Then the reasons for selecting community detection algorithms will be explained. Afterwards, the architecture of neural network will be described along with analyzing its rationality for network embedding. Finally, the entire model will be summarized.

A. OVERVIEW

The proposed model is designed for homogeneous network embedding and contains two main modules, namely a community detection module and a deep learning module. The community detection module is used to extract the community information from network via community detection algorithms. The deep learning module can integrate both local information and community information to learn vertex embeddings. Therefore, the proposed model takes the network data as input and outputs the corresponding embedding result via these two modules.

B. COMMUNITY DETECTION MODULE

As analyzed above, community information can reveal some hidden relationship in network from a global view. To obtain community information, community detection should be executed. In our model, two classical and effective algorithms, namely Label Propagation Algorithm (LPA) [41] and Infomap [42], are applied as candidate community detection modules.

1) **Label Propagation Algorithm (LPA) [41]**:
After allocating each vertex with a unique community label as initialization, LPA merges community label of each vertex through voting by neighbors until convergence. Therefore, the community information propagates globally from one-hop relationship to higher hop relationship via label voting and community merging. Besides, LPA is a nearly linear time algorithm to detect communities. Due to its information propagating mode and high efficiency, LPA is employed as one type of community detection module and a model called ComVAE (LPA) is constructed.
2) **Infomap [42]**:
Different from LPA, Infomap focuses on encoding vertex sequences with the shortest length based on information theory, and detects communities through a deterministic greedy search strategy. To obtain vertex sequences, random walk is used as a strategy to collect higher-order information. Besides, the greedy search strategy integrates information globally and merges communities. Because random walk and greedy search are popular strategies for obtaining community information, Infomap is employed as another type of community detection module and a model called ComVAE (Infomap) is constructed.

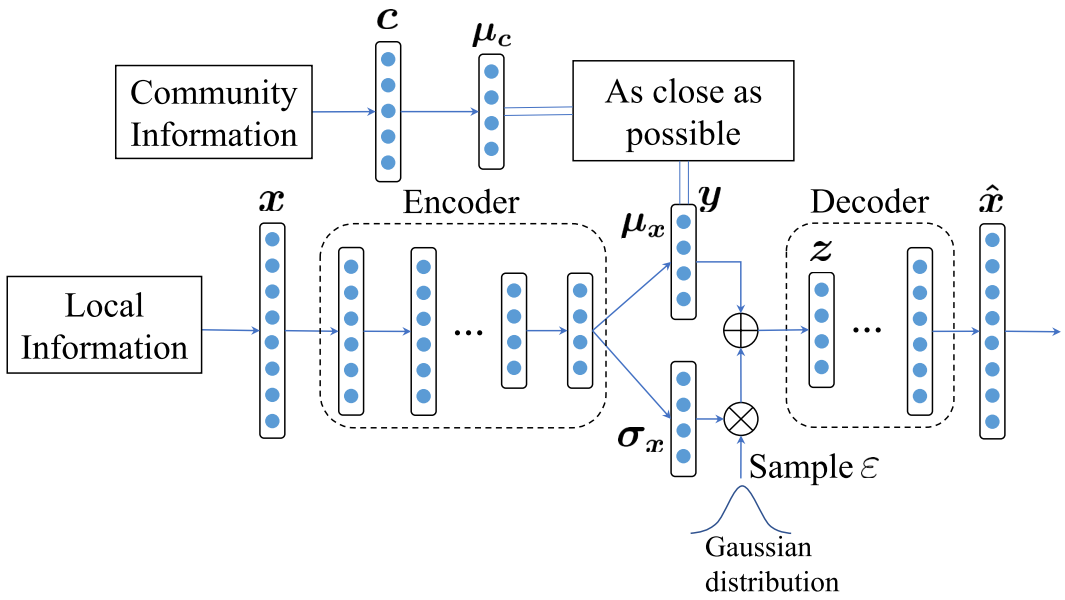


FIGURE 2. Architecture of the deep learning module in ComVAE. Local information x and community information c are inputs of the module while y is the vertex representation learned after training. $\hat{x}$ is the reconstructed vector of the original data. z is the vector to be decoded, which is obtained by reparameterization. μ_x and σ_x can be respectively regarded as the mean and the standard deviation of data distribution, while μ_c can be regarded as the mean of community information distribution. ε is sampled from the normalized Gaussian distribution. In the encoder, there is always a dropout layer following a full-connected layer. Other layers in the module are fully connected.

C. DEEP LEARNING MODULE

In this section, we introduce in detail the architecture of deep learning module in ComVAE and explain how the conditional variational autoencoder can integrate local information with community information to learn vertex representations.

1) ARCHITECTURE

Deep learning module is the core part of our model, the architecture of which is shown in Figure 2.

In the module, most layers are fully connected in order to extract information as much as possible from the input vector. Therefore, for those full-connected layers, given an input vector $\boldsymbol{v}$, the corresponding output vector $\boldsymbol{u}$ can be calculated as follows:

$$\boldsymbol{u} = h(W\boldsymbol{v} + \boldsymbol{b}) \tag{1}$$

where W is the weight matrix, $\boldsymbol{b}$ is the bias vector and $h(\cdot)$ is the activation function. Notice that the activation functions in both encoder and decoder should be non-linear functions for learning non-linear mappings from the original vertex space to the embedding space.

In the encoder architecture, in order to avoid overfitting, we add a dropout layer after every full-connected layer. The dropout layer disables a portion of neurals randomly in the training phase, which can strengthen the robustness of the network embedding performance.

2) LOSS FUNCTION

Based on the variational autoencoder, the loss function in this module consists of the reconstruction loss and the KL divergence loss.

The reconstruction loss is generated by data compression and recovery. In essence, the neural network of our deep learning module is an autoencoder, which contains a reconstruction procedure. Autoencoder is the architecture with an encoder and a decoder. The encoder ($f_{\theta_1}(\cdot)$) aims to compress the original vector $\boldsymbol{x}$ into a low-dimensional representation, while the decoder ($g_{\theta_2}(\cdot)$) aims to generate a new vector $\hat{\boldsymbol{x}}$ with the same dimension as $\boldsymbol{x}$ from the low-dimensional representation. In order to guarantee the close relationship between interconnected vertices to be maintained in the low-dimensional space, it is expected that the local information in $\boldsymbol{x}$ should change as little as possible after dimension compression and recovery. Aiming at minimizing the loss of local information, the reconstruction loss used in our deep learning module is:

$$\begin{aligned} Loss_r &= Loss(\boldsymbol{x}, g_{\theta_2}(f_{\theta_1}(\boldsymbol{x}))) \\ &= Loss(\boldsymbol{x}, \hat{\boldsymbol{x}}) \\ &= -\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}[\boldsymbol{x}_i \log \hat{\boldsymbol{x}}_i + (1-\boldsymbol{x}_i)\log(1-\hat{\boldsymbol{x}}_i)], \end{aligned} \tag{2}$$

where $|\mathcal{V}|$ is the number of vertices, and the function $Loss(\cdot)$ calculates the cross entropy between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$.

The KL divergence loss is the particular loss of the variational autoencoder compared to the traditional autoencoder. As shown in Figure 2, two layers denoted as $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x}$ can be regarded as the mean and the standard deviation of data distribution respectively. The latent variable vector $\boldsymbol{z}$ can be obtained by sampling from the Gaussian distribution determined by $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x}$. To avoid the issue of the randomness from different sampling results, the model is expected

to learn robust embedding representations with always relatively small reconstruction loss. So this sampling procedure can be regarded as another way to strengthen the robustness, in addition to the dropout technique. For back propagation of the sampling procedure, reparameterization trick is adopted. Therefore, the distribution of $p(\boldsymbol{z}|\boldsymbol{x})$ is $N(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x}^2)$. According to the assumption of the variational autoencoder, the distribution of $p(\boldsymbol{z})$ is $N(\boldsymbol{0}, \boldsymbol{I})$. To make these two distributions as close as possible, the KL divergence loss is:

$$\begin{aligned} Loss_{kl'} &= KL(p(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \\ &= KL(N(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x}^2)||N(\boldsymbol{0}, \boldsymbol{I})) \\ &= \frac{1}{2}\sum_{i=1}^{d}(\boldsymbol{\mu_x}_i^2 + \boldsymbol{\sigma_x}_i^2 - \log \boldsymbol{\sigma_x}_i^2 - 1), \end{aligned} \tag{3}$$

where $KL(\cdot)$ is the KL divergence for measuring difference between two distributions and d is the dimension of embedding representation.

To utilize community information in the deep learning module, a layer is added to train the vector $\boldsymbol{\mu_c}$ with the same dimension as $\boldsymbol{\mu_x}$, which is regarded as the mean of community information distribution. Similar to the case of local information, vertices with similar community information are expected to be embedded nearly in the low-dimensional space. Thus, the distribution of $p(\boldsymbol{z})$ is assumed to follow different Gaussian distributions according to different community information, by which the community information can be considered in the loss function. The new KL divergence loss is designed as follows:

$$\begin{aligned} Loss_{kl} &= KL(N(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x}^2)||N(\boldsymbol{\mu_c}, \boldsymbol{I})) \\ &= \frac{1}{2}\sum_{i=1}^{d}[(\boldsymbol{\mu_x} - \boldsymbol{\mu_c})_i^2 + \boldsymbol{\sigma_x}_i^2 - \log \boldsymbol{\sigma_x}_i^2 - 1], \end{aligned} \tag{4}$$

where $KL(\cdot)$ and d have the same meanings as Eq. (3).

Combining the reconstruction loss and the KL loss, the total loss we need to minimize is:

$$Loss_{total} = Loss_r + Loss_{kl}. \tag{5}$$

After the training phase, instead of taking the vector $\boldsymbol{z}$ as the generation result of the traditional variational autoencoder, the vector $\boldsymbol{\mu_x}$ is taken as the embedding representation $\boldsymbol{y}$ for two reasons. Firstly, $\boldsymbol{\mu_x}$ directly interacts with $\boldsymbol{\mu_c}$, reflecting on the KL divergence loss function, which can preserve the influence of community information in the embedded vertex representations. Secondly, after training the model, $\boldsymbol{\mu_x}$ can generate a stable embedding result with the fixed parameters and weights of the neural network, avoiding the randomness brought by the sampling procedure.

D. MODEL SUMMARY

As described above, our ComVAE model consists of a community detection module and a deep learning module. In ComVAE, we first obtain local information from network and community information by community detection. Then, we utilize both local information and community information simultaneously to train the deep learning module, obtaining embedding representation of every vertex in the network. As a conclusion, the ComVAE model is effective due to the utilization of deep learning techniques and information from different views, robust due to the dropout and sampling techniques, efficient due to the relatively shallow layers with good performance, which will be confirmed in the experiments.

IV. EXPERIMENT

In this section, four downstream tasks, namely network reconstruction, node classification, link prediction and visualization, are conducted as extensive experiments to verify the performance of ComVAE, compared with other state-of-the-art models. The code of our model is publicly available at https://github.com/PP8818/ComVAE.

A. EXPERIMENTAL SETTINGS

1) DATASETS

In experiments, eight real-world networks in two major types with different scales are used in different downstream tasks.

- Type I: WebKB.[1] The WebKB networks consist of subnetworks from four universities, including Texas with 187 vertices, Cornell with 195 vertices, Wisconsin with 265 vertices and Washington with 230 vertices. Each subnetwork is taken as an independent dataset.
- Type II: Social Networks.[2] The datasets are the Facebook networks formed by users from four universities, including Amherst41 with 2235 vertices, Hamilton46 with 2314 vertices, Mich67 with 3748 vertices and Rochester38 with 4563 vertices [43]. Each subnetwork is taken as an independent dataset.

2) COMPARING MODELS

Because ComVAE is designed for homogeneous network, other types of network such as attributed network and multi-view network are not suitable for our proposed model. For evaluating the performance of ComVAE, five state-of-the-art algorithms designed for homogeneous network are adopted as comparing models in the following downstream tasks. For classical models, DeepWalk [5] utilizes random walk and skipgram model to learn vertex embeddings. Node2vec [14] improves DeepWalk through adding two parameters p and q in random walk procedure, which can control the tendency of walking following breath-first searching or depth-first searching. LINE [33] combines the first-order and second-order proximity in embedding representations. Because community information is taken as the global information in our model, M-NMF [37], aiming at preserving community structures during network embedding, is selected as another comparing model. Due to the deep learning techniques used in our model, SDNE [23] is also adopted as a representative model, which utilizes the first-order and second-order proximity to train an autoencoder.

[1] http://linqs.cs.umd.edu/projects/projects/lbc/
[2] https://escience.rpi.edu/data/DA/fb100/

TABLE 1. Dimension of encoding layers.

Datasets	Dimension
Taxas Cornell Wisconsin Washington	input-128-64-32
Amherst41 Hamilton46 Mich67 Rochester38	input-1024-256-64

3) PARAMETER SETTINGS

In the experiments, the code of all the comparing models is obtained from the open source website Github.[3] The embedding dimension of the four datasets in WebKB is set as 32, while the dimension of the other four datasets in Social Networks is set as 64. For the intermediate layers of deep learning based models including ComVAE and SDNE, the neural numbers in encoding layers are not intentionally selected and just set as some common numbers, which is listed in Table 1. According to the conventional design of autoencoder, the neural number of every decoding layer is set the same as that of the corresponding encoding layer. To train the ComVAE model, after selecting community detection method, inputting network data and tuning parameters, the vertex embeddings can be obtained as output.

The detailed settings of other parameters are listed as follows:

- ComVAE. With different community detection modules, ComVAE(Infomap) and ComVAE(LPA) are respectively evaluated in experiments. In these two models, Sigmoid is set as the activation function in the last decoding layer while ReLU is the activation function in other layers. The standard deviation of Gaussian distribution in the sampling procedure is set as 1. Besides, the dropout rate is set as 0.1 and early stopping is also adopted in order to avoid overfitting.
- DeepWalk.[4] The parameters *walk length*, *window size* and *walk number* are set as 40, 5 and 10 respectively.
- node2vec.[5] The parameters *walk length*, *window size* and *walk number* are set as 80, 10 and 10 respectively. Because parameters p and q control the tendencies of random walker, we select different values and construct two comparing models, namely node2vec(p0.5q1) and node2vec(p1q0.5), to test the performance.
- LINE.[6] The parameter *order* is set as 3, which means both the first-order proximity and the second-order proximity are taken into consideration.
- M-NMF.[7] The parameters α, β and λ are set as 1, 1 and 10^9 respectively. To construct the community indicator matrix and community representation matrix used in M-NMF, the parameter k is set as the real number of communities in each dataset.
- SDNE.[8] The parameters α, γ and *reg* in the loss function are set as 100, 1 and 1 respectively.

B. NETWORK RECONSTRUCTION

After network embedding, the embedded vertex representations can be regarded as some discrete points in the embedding space. An intuitive way to test the performance is reconstructing a network based on these discrete points, so as to investigate the similarity between the reconstructed network and the original network. To measure the similarity between networks, connectivity between vertices is a key consideration. In other words, if the connectivity can be precisely predicted based on the vertex embeddings, the original network can be well reconstructed.

To predict the connectivity, a Logistic Regression classifier is trained, which can judge whether a link exists between a vertex pair. Given a concatenated vertex embedding pair as input feature, connected vertex pairs in the original network are used as positive instances, while the same number of disconnected ones are randomly sampled as negative instances. For each type of instances, 60% vertex pairs are selected as the training set and the remaining ones are the testing set. AUC [44] is utilized to evaluate the prediction performance.

The results are shown in Table 2 and Table 3, where the best performance of each dataset among all models is represented in bold. From the results, one of the ComVAE models obtains the highest AUC on seven out of the eight datasets, while the AUC values obtained by M-NMF and SDNE are also not bad. Based on the results, the analysis can be drawn as follows. Firstly, network reconstruction aims to measure how well the first-order proximity is maintained after network embedding. Though all the compared models take the first-order proximity into account, from the results, deep learning based models such as ComVAE and SDNE are able to extract and preserve the proximity information more precisely. Secondly, if community information is considered in models such as ComVAE and M-NMF, a vertex can be embedded closer to some group of embedded vertices which have close relationship with it in the original network, resulting in more accurate relationship prediction after network embedding and hence higher AUC in this task. Therefore, the ComVAE models can perform well and show the effectiveness on network reconstruction.

C. NODE CLASSIFICATION

After obtaining the embedded vertex representations, node classification is a common downstream task to evaluate the performance of network embedding, via allocating labels for vertices in the original network and measuring the accuracy. In the experiment, the LIBLINEAR [45] package is utilized to train the classifiers. For each dataset, different portions of vertices are randomly selected as the training set and the rest vertices are the testing set, while the training portions vary from 10% to 90% with 10% interval. Due to the page limit, only the results on datasets in Social Networks in terms of

[3]https://github.com
[4]https://github.com/phanein/deepwalk
[5]https://github.com/thunlp/OpenNE
[6]https://github.com/thunlp/OpenNE
[7]https://github.com/AnryYang/M-NMF
[8]https://github.com/suanrong/SDNE

TABLE 2. Network reconstruction results on WebKB in terms of AUC.

Models	Texas	Cornell	Wisconsin	Washington
DeepWalk	0.6556	0.6532	0.6400	0.6801
node2vec(p0.5q1)	0.6719	0.6593	0.6576	0.6912
node2vec(p1q0.5)	0.6269	0.6891	0.6207	0.7192
LINE	0.6834	0.6334	0.6188	0.6646
M-NMF	0.8080	0.7886	0.7328	**0.8129**
SDNE	0.7982	0.7629	0.7550	0.7988
ComVAE(Infomap)	**0.8481**	**0.8298**	**0.7821**	0.6410
ComVAE(LPA)	0.7939	0.7340	0.6903	0.7205

TABLE 3. Network reconstruction results on Social Networks in terms of AUC.

Models	Amherst41	Hamilton46	Mich67	Rochester38
DeepWalk	0.7006	0.6770	0.7347	0.6756
node2vec(p0.5q1)	0.6853	0.6535	0.7007	0.6336
node2vec(p1q0.5)	0.6801	0.6591	0.6920	0.6299
LINE	0.7081	0.6929	0.7102	0.7207
M-NMF	0.7496	0.7421	0.7593	0.7522
SDNE	0.7531	0.7385	0.7913	0.7556
ComVAE(Infomap)	0.7817	0.7706	0.8114	0.7840
ComVAE(LPA)	**0.7831**	**0.7740**	**0.8118**	**0.7847**

TABLE 4. Node classification results on Amherst41 in terms of accuracy.

Models	Training rate (%)								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	0.7813	0.8031	**0.8185**	0.8121	0.8202	0.8177	0.8241	**0.8322**	**0.8170**
node2vec(p0.5q1)	0.7868	0.7925	0.8121	0.8128	0.8193	0.8233	0.8212	0.8233	0.8125
node2vec(p1q0.5)	0.7729	0.8020	0.8141	0.8166	**0.8229**	0.8244	**0.8301**	0.8300	**0.8170**
LINE	0.7719	0.7880	0.7962	0.7964	0.7996	0.8110	0.8077	0.8188	0.7902
M-NMF	0.7386	0.7735	0.8115	0.8151	0.8202	0.8166	0.8241	0.8300	0.8036
SDNE	0.7475	0.7550	0.7668	0.7703	0.7764	0.7752	0.7809	0.7852	0.7589
ComVAE(Infomap)	0.7858	0.8054	0.8089	0.8121	0.8148	**0.8311**	**0.8301**	0.8300	0.7991
ComVAE(LPA)	**0.7903**	**0.8087**	0.8153	**0.8180**	**0.8229**	0.8221	0.8182	0.8233	0.8036

TABLE 5. Node classification results on Hamilton46 in terms of accuracy.

Models	Training rate (%)								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	0.8032	0.8175	0.8216	0.8193	0.8211	0.8326	0.8288	0.8294	0.8147
node2vec(p0.5q1)	0.7998	0.8051	0.8185	0.8243	0.8271	0.8413	0.8403	0.8337	0.8190
node2vec(p1q0.5)	0.7902	0.8099	0.8148	0.8179	0.8133	0.8283	0.8288	0.8294	0.8017
LINE	0.7691	0.7862	0.7981	0.8092	0.8081	0.8251	0.8144	0.8143	0.7888
M-NMF	0.7792	0.8051	0.8080	0.8157	**0.8289**	0.8413	0.8417	0.8359	0.8147
SDNE	0.7460	0.7716	0.7815	0.7862	0.7822	0.7970	0.7885	0.7797	0.7845
ComVAE(Infomap)	**0.8080**	**0.8229**	0.8253	0.8272	0.8237	0.8348	**0.8489**	**0.8402**	0.8362
ComVAE(LPA)	0.7998	0.8121	**0.8296**	**0.8287**	0.8202	**0.8445**	0.8417	0.8380	**0.8405**

TABLE 6. Node classification results on Mich67 in terms of accuracy.

Models	Training rate (%)								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	0.4321	0.4682	0.4665	0.4629	0.4787	0.4933	0.4676	0.4747	0.4747
node2vec(p0.5q1)	0.4247	0.4532	0.4638	0.4713	0.4664	0.4687	0.4640	0.4707	0.4560
node2vec(p1q0.5)	0.4262	0.4558	0.4535	0.4633	0.4642	0.4740	0.4640	0.4720	0.4667
LINE	0.4191	0.4461	0.4466	0.4518	0.4498	0.4587	0.4391	0.4387	0.4560
M-NMF	0.3945	0.4305	0.4436	0.4513	0.4525	0.4560	0.4569	0.4667	0.4693
SDNE	0.3912	0.4185	0.4409	0.4455	0.4482	0.4400	0.4329	0.4480	0.4400
ComVAE(Infomap)	**0.4478**	0.4628	**0.4825**	**0.4878**	0.4824	0.4820	0.4676	0.4773	0.4800
ComVAE(LPA)	0.4404	**0.4758**	0.4726	0.4820	**0.4920**	**0.4960**	**0.4800**	**0.4827**	**0.5013**

accuracy are presented in Table 4-7, where the best performance of each training rate among all models is represented in bold.

From the results in Table 4-7, our proposed models obtain the highest accuracy in most cases (88.9%), which demonstrates the effectiveness of ComVAE on node classification. Especially, one of our ComVAE models performs the best under any training rate on the Mich67 and Rochester38 datasets. By conducting community detection based on the original network, some implicit relationship can be revealed from a global view and similar vertices tend to be divided into the same community. Because the community

TABLE 7. Node classification results on Rochester38 in terms of accuracy.

Models	Training rate (%)								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	0.7244	0.7529	0.7565	0.7706	**0.7691**	0.7760	0.7736	0.7766	0.7856
node2vec(p0.5q1)	0.7278	0.7420	0.7549	0.7652	0.7651	0.7722	0.7765	0.7831	0.7943
node2vec(p1q0.5)	0.7295	0.7445	0.7505	0.7630	0.7621	0.7782	0.7721	0.7711	0.7812
LINE	0.7022	0.7160	0.7299	0.7411	0.7384	0.7519	0.7495	0.7645	0.7702
M-NMF	0.6937	0.7141	0.7214	0.7407	0.7568	0.7809	0.7750	0.7766	0.8009
SDNE	0.6409	0.6431	0.6585	0.6581	0.6591	0.6665	0.6633	0.6572	0.6740
ComVAE(Infomap)	**0.7492**	0.7538	0.7643	0.7677	**0.7691**	0.7804	0.7779	0.7809	0.8053
ComVAE(LPA)	0.7414	**0.7573**	**0.7700**	**0.7743**	0.7660	**0.7837**	**0.7845**	**0.7963**	**0.8074**

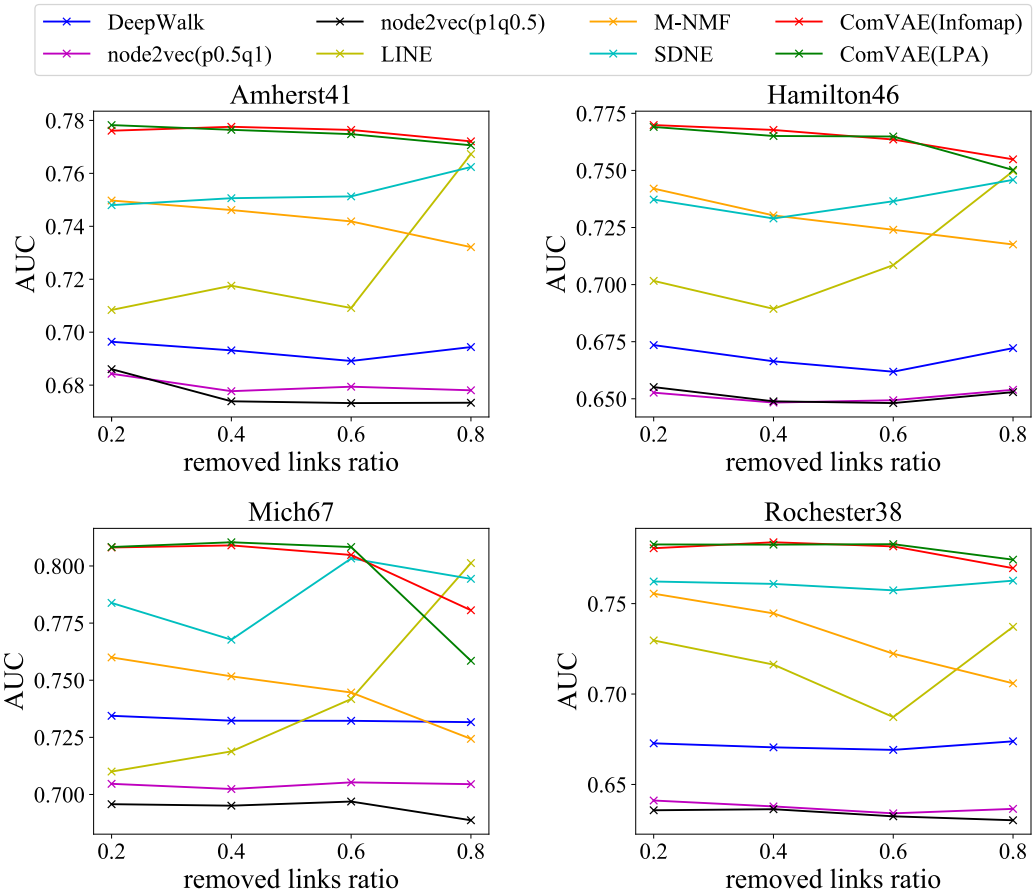


FIGURE 3. Link prediction results on Social Networks in terms of AUC.

information is preserved well via deep learning techniques during the network embedding procedure, the embedded vertices are more discriminative and hence suitable for classification, which helps to improve the accuracy in most cases. Therefore, the proposed ComVAE models perform better in the node classification task.

D. LINK PREDICTION

Link prediction is another common downstream task to evaluate the performance of network embedding. Similar to network reconstruction, they are both connectivity prediction tasks. However, link prediction aims to predict nonexistent links which are about to generate. To obtain the ground-truth, some ratios of links are randomly removed from the original network and the new networks are embedded by models. Specifically, the removed links ratios are respectively set as 0.2, 0.4, 0.6 and 0.8, resulting in four new networks for each dataset. In addition, the removed links are used as the corresponding testing set. To guarantee the connectivity of new networks after links removed, only datasets in Social Networks are selected in this experiment. Besides, we adopt the same method introduced in Section IV-B to train the classifier and predict the links. AUC is still used as the evaluation criterion in this task.

From the link prediction results shown in Figure 3, two conclusions can be drawn as follows. Firstly, the AUC values obtained by M-NMF, SDNE and ComVAE are always higher than those obtained by other models, which indicates the

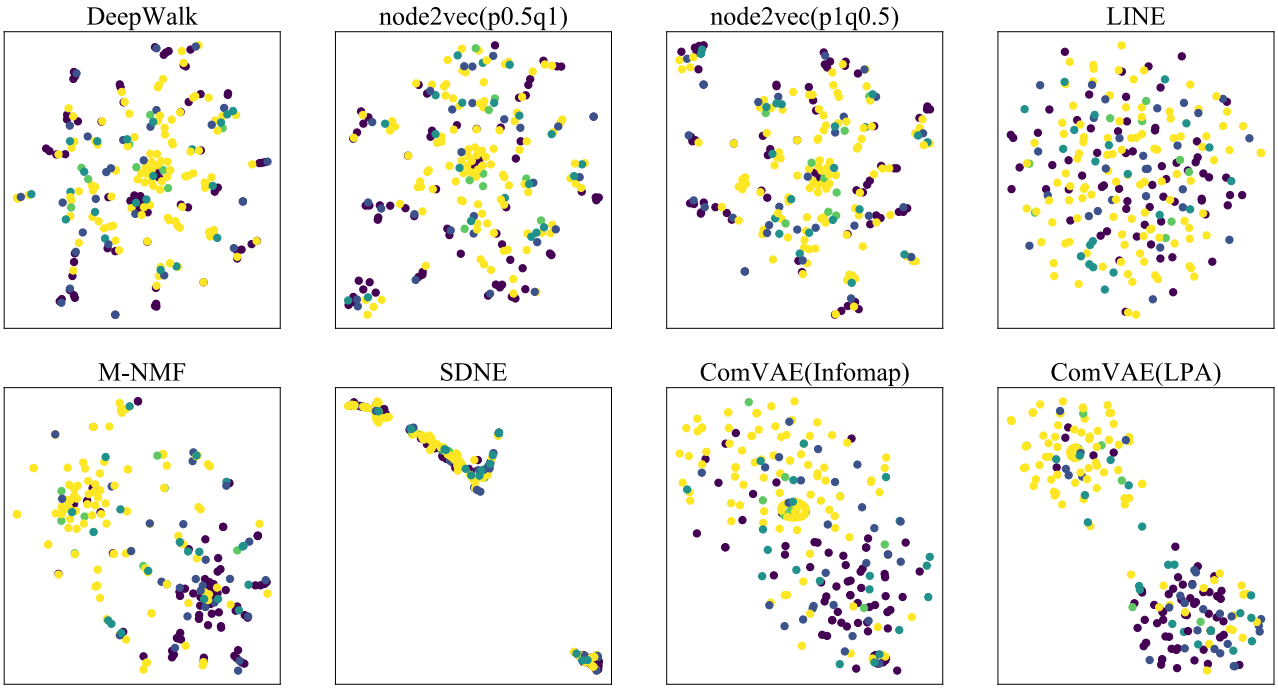


FIGURE 4. Visualization results on the Washington dataset.

considerations of community information and deep learning techniques are helpful in improving the performance in the link prediction task. Secondly, the AUC values obtained by two ComVAE models are close to each other and outperform other models in most cases, which demonstrates the robustness of ComVAE and the further performance improvements achieved by integrating both of the two aforementioned considerations.

E. VISUALIZATION

Visualization is another intuitive way to demonstrate the quality of network embedding. In this downstream task, t-SNE [46] is used to map vertex representations into a 2D space and then the points in the 2D space are visualized. By taking the Washington dataset as an example, the visualization results are shown in Figure 4.

In the Washington dataset, there are five communities with different colors. From the results, the embeddings of no model can be visualized with highly cohesive in the same community and highly distinguishable between different communities. For this phenomenon, the main reason is considered as the information loss during transformation from vertex embedding space to 2D space. To compare the visualization results among models, we mainly focus on the two largest communities colored by yellow and purple. In the visualization results of M-NMF and ComVAE, the yellow points are mainly distributed in the upper left corner while the purple points are mainly in the lower right corner, compared with the irregular color distributions of other models. Therefore, the huge role of preserving community information in the embedded representations emerges again. More precisely, with the greater spacing and clearer boundary between yellow and purple points, the ComVAE(LPA) model outperforms other models in this task.

V. CONCLUSION

In this paper, we have proposed a community based variational autoencoder model for network embedding. The proposed model not only utilizes the deep learning techniques, but also integrates local information with community information. Extensive experiments have been conducted on two types of real-world datasets in four downstream tasks. The experimental results have confirmed the superiority of our proposed models over several state-of-the-art models.

REFERENCES

[1] J.-H. Li, C.-D. Wang, P.-Z. Li, and J.-H. Lai, "Discriminative metric learning for multi-view graph partitioning," *Pattern Recognit.*, vol. 75, pp. 199–213, Mar. 2018.

[2] L. Huang, C.-D. Wang, and H.-Y. Chao, "Overlapping community detection in multi-view brain network," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Madrid, Spain, Dec. 2018, pp. 655–658.

[3] Y. Long *et al.*, "Learning heterogeneous network embedding from text and links," *IEEE Access*, vol. 6, pp. 55850–55860, 2018.

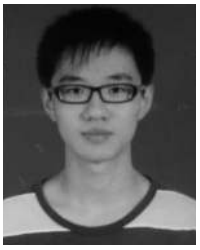
[4] L. Huang, C.-D. Wang, and H.-Y. Chao, "A harmonic motif modularity approach for multi-layer network community detection," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Singapore, Nov. 2018, pp. 1043–1048.

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

[6] C. Li *et al.*, "PPNE: Property preserving network embedding," in *Proc. 22nd Int. Conf. Database Syst. for Adv. Appl. (DASFAA)*, 2017, pp. 163–179.

[7] J. Han *et al.*, "Representing and retrieving video shots in human-centric brain imaging space," *IEEE Trans. Image Process.*, vol. 22, no. 7, pp. 2723–2736, Jul. 2013.

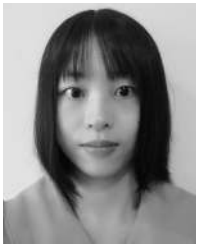
[8] D. Zhang, D. Meng, and J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 5, pp. 865–878, May 2017.
[9] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1105–1114.
[10] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 817–826.
[11] C. Li, Z. Li, S. Wang, Y. Yang, X. Zhang, and J. Zhou, "Semi-supervised network embedding," in *22nd Int. Conf. Database Syst. Adv. Appl. (DASFAA)*, Suzhou, China, Mar. 2017, pp. 131–147.
[12] T. Li, J. Zhang, S. Y. Philip, Y. Zhang, and Y. Yan, "Deep dynamic network embedding for link prediction," *IEEE Access*, vol. 6, pp. 29219–29230, 2018.
[13] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepWalk: Discriminative learning of network representation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 3889–3895.
[14] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[15] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.
[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova. (2018). "BERT: Pre-training of deep bidirectional transformers for language understanding." [Online]. Available: https://arxiv.org/abs/1810.04805
[17] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1746–1751.
[18] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, Dec. 2016.
[19] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of co-salient objects by looking deep and wide," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 215–232, 2016.
[20] M. Bhuiyan and M. Al Hasan, "Representing graphs as bag of vertices and partitions for graph classification," *Data Sci. Eng.*, vol. 3, no. 2, pp. 150–165, 2018.
[21] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 3–10.
[22] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1293–1299.
[23] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.
[24] D. Yang, S. Wang, C. Li, X. Zhang, and Z. Li, "From properties to links: Deep network embedding on incomplete graphs," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, Singapore, Nov. 2017, pp. 367–376.
[25] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. .
[26] H. Wang *et al.*, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[27] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.
[28] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.
[29] D. P. Kingma and M. Welling. (2013). "Auto-encoding variational Bayes." [Online]. Available: https://arxiv.org/abs/1312.6114
[30] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.
[31] H. Li, H. Wang, Z. Yang, and M. Odagaki, "Variation autoencoder based network representation learning for classification," in *Proc. ACL Student Res. Workshop*, 2017, pp. 56–61.
[32] F. Huang, X. Zhang, C. Li, Z. Li, Y. He, and Z. Zhao, "Multimodal network embedding via attention based multi-view variational autoencoder," in *Proc. ACM Int. Conf. Multimedia Retr.*, 2018, pp. 108–116.
[33] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[34] M. Newman, *Networks*. London, U.K.: Oxford Univ. Press, 2018.
[35] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, 2008, Art. no. P10008.
[36] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Apr. 2002.
[37] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.
[38] L. Du, Z. Lu, Y. Wang, G. Song, Y. Wang, and W. Chen, "Galaxy network embedding: A hierarchical community structure preserving approach," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2079–2085.
[39] J.-H. Li, C.-D. Wang, L. Huang, D. Huang, J.-H. Lai, and P. Chen, "Attributed network embedding with micro-meso structure," in *Proc. 23rd Int. Conf. Database Syst. for Adv. Appl. (DASFAA)*, 2018, pp. 20–36.
[40] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.
[41] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 76, no. 3, p. 036106, 2007.
[42] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 2, pp. 1118–1123, 2008.
[43] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, to be published.
[44] M. Gao, L. Chen, X. He, and A. Zhou, "BINE: Bipartite network embedding," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2018, pp. 715–724.
[45] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
[46] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.



WEI SHI received the bachelor's degree from Sun Yat-sen University, in 2018, where he is currently a graduate student. His current research interest includes data mining.

LING HUANG received the bachelor's and master's degrees from the South China University of Technology, in 2009 and 2013, respectively. She is currently pursuing the Ph.D. degree with Sun Yat-sen University. She has published several papers in international journals and conferences, such as *Pattern Recognition*, IEEE Access, *Information Sciences*, *Knowledge-Based Systems*, AAAI, IEEE ICDM, IEEE BIBM, and DASFAA. Her research interest includes data mining.

CHANG-DONG WANG received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2013. He was a Visiting Student with the University of Illinois at Chicago, in 2012. In 2013, he joined the School of Mobile Information Engineering, Sun Yat-sen University, as an Assistant Professor, where he is currently an Associate Professor with the School of Data and Computer Science. He has published over 100 scientific papers in international journals and conferences, such as IEEE TPAMI, IEEE TKDE, IEEE TCYB, IEEE TSMC-C, *Pattern Recognition*, IEEE Access, KAIS, AAAI, ICDM, CIKM, SDM, DASFAA, and BIBM. His current research interests include machine learning and data mining. His ICDM 2010 paper won the Honorable Mention for Best Research Paper Awards. He also received the 2012 Microsoft Research Fellowship Nomination Award. He was awarded 2015 Chinese Association for Artificial Intelligence Outstanding Dissertation.



JUAN-HUI LI received the bachelor's degree from Sun Yat-sen University, in 2017, where she is currently a graduate student. Her current research interests include network embedding and community detection.

YONG TANG received the B.S. degree in computer science from Wuhan University, in 1985, and the Ph.D. degree from the University of Science and Technology of China, in 2001. He is currently a Professor and the Dean of the School of Computer Science, South China Normal University, and also serves as the Director of the Services Computing Engineering Research Center of Guangdong Province. His research interests include database and cooperative software, temporal information processing, social networks, and big data analytics. He has completed more than 30 research and development projects, and has authored or co-authored more than 100 publications in these areas.

CHENGZHOU FU received the master's degree in software engineering and the Ph.D. degree in computer science from South China Normal University, China, in 2012 and 2017, respectively. His current research interests include social networks and data mining.