Self-Training Enhanced: Network Embedding and Overlapping Community Detection With Adversarial Learning

Junyang Chen, Zhiguo Gong, *Senior Member, IEEE*, Jiqian Mo, Wei Wang, Wei Wang, Cong Wang, Xiao Dong, Weiwen Liu, and Kaishun Wu

***Abstract*— Network embedding (NE) aims to encode the relations of vertices into a low-dimensional space. After NE, we can obtain the learned vectors of vertices that preserve the proximity of network structures for subsequent applications, e.g., vertex classification and link prediction. In existing NE models, they usually exploit the skip-gram with a negative sampling method to optimize their objective functions. Generally, this method learns the vertex representation only from the local connectivity of vertices (i.e., neighbors). However, there is a larger scope of vertex connectivity in real-world scenarios: a vertex may have multifaceted aspects and should belong to overlapping communities. Taking a social network as the overlapping example, a user may subscribe to the channels of politics, economy, and sports simultaneously, but the politics share more common attributes with the economy and less with the sports. In this article, we propose an adversarial learning approach (ACNE) for modeling overlapping communities of vertices. Specifically, we map the association between communities and vertices into an embedding space. Moreover, we take further research on enhancing our ACNE with the following two operations. First, in the initialization stage, we adopt a walking strategy with perception to obtain paths containing more possible boundary vertices to improve overlapping community detection. Then, after representation learning with ACNE, we use soft community assignments from a simple classifier as supervision to update the weights of ACNE. This self-training mechanism referred to as ACNE-ST can help ACNE to achieve better performance. Experimental results demonstrate that the proposed methods, including ACNE and ACNE-ST, can outperform the state-of-the-art models on the subsequent tasks of vertex classification and overlapping community detection.**

***Index Terms*— Adversarial learning, network embedding (NE), overlapping community detection, self-training.**

Manuscript received April 19, 2020; revised October 19, 2020, February 18, 2021, and April 12, 2021; accepted May 20, 2021. This work was supported in part by the National Key Research and Development (R&D) Program of China under Grant 2019YFB1600704; in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B111103001; in part by the National Natural Science Foundation of China under Grant U2001207, Grant 61872248, and Grant 61902249; and in part by the Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08x603. *(Corresponding authors: Zhiguo Gong; Kaishun Wu.)*

Junyang Chen and Kaishun Wu are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518057, China (e-mail: junyangchen@szu.edu.cn; wu@szu.edu.cn).

Zhiguo Gong and Jiqian Mo are with the State Key Laboratory of Internet of Things for Smart City, Department of Computer Information Science, University of Macau, Taipa, Macau (e-mail: fstzgg@umac.mo; yb87431@um.edu.mo).

Wei Wang is with the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518107, China (e-mail: wangw328@mail.sysu.edu.cn).

Wei Wang is with the School of Information Science and Engineering, Dalian University of Technology, Dalian 116000, China (e-mail: wwlovetransfer@mail.dlut.edu.cn).

Cong Wang is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: supercong94@gmail.com).

Xiao Dong is with the School of Artificial Intelligence, Sun Yat-sen University, Shenzhen 518107, China (e-mail: dx.icandoit@gmail.com).

Weiwen Liu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: wwliu@cse.cuhk.edu.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3083318.

Digital Object Identifier 10.1109/TNNLS.2021.3083318

I. INTRODUCTION

IN REAL-WORLD scenarios, graph structures are ubiquitous, e.g., citation networks and social networks. To learn relations among vertices in a graph, the network embedding (NE) method is proposed. Generally, it aims to map vertices into a low-dimensional space where similar ones are assigned to nearby areas [1]. After NE, we can obtain learned embeddings for subsequent applications, e.g., vertex classification [2] and link prediction [3]. Up until now, a lot of works on preserving the proximity structure of networks have been devoted to NE. For example, starting from DeepWalk [2] performing truncated random walks to explore network structures, LINE [4] extends it by adopting the strategies of breadth-first search (BFS) and depth-first search (DFS). After that, node2vec [3] takes both these strategies into account and further designs a biased random walk procedure to explore diverse neighbors. To sum up, there is a common pattern for these methods: after searching out neighbors of vertices with various strategies, they all adopt skip-gram with negative sampling [5], [6] to learn vertex representations. Specifically, skip-gram is a language model that learns word embeddings by maximizing the probability of word co-occurrences (corresponding to vertex neighbors in graphs) within a sliding window. Thus, the purpose of NE aims to make a target vertex being close to its neighbors and meanwhile being far away from its negative samples.

Nevertheless, most of the current NE models only consider the local connectivity of vertices' neighbors during the learning while ignoring the global pattern, which is regarded as communities in complex graphs. In many practical scenarios, vertices of networks may contain disparate aspects [8], [9],

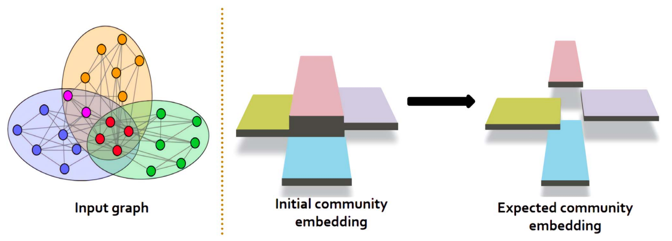


Fig. 1. Instance of overlapping communities in one given graph. Here, we take an online video website as an example from which users are represented as vertices and edges are denoted as the relations of subscribing to common video genres (this figure is referred to [7]). The purpose of embedding learning aims to obtain the embeddings that preserve both the vertex and community proximities in a low-dimensional space.

i.e., different paths expanding from a vertex to its n-step neighbors may result from the expression of its aspects (i.e., communities). For instance, we provide an example about a social network of an online video website in Fig. 1, where users are represented as vertices and edges are denoted as the relations of subscribing to common video genres. In practice, a user may simultaneously subscribe to politics, economy, and sports channels (we call them communities in the following). The politics community usually shares more users with the economy and less with sports. As such, if we ignore community information in NE, the learned embeddings of users and their communities may be indistinguishable in the embedding space. In general, the community structure is an important pattern of networks, which is expected to benefit both NE and overlapping community detection.

There are several challenges in overlapped community-aware NE. The first one is how to determine the vertex communities. The second is how to transfer the community assignments of vertices from a discrete space to a continuous one. The last is how to optimize a loss function to guild the embeddings of vertices and their assigned communities being close to each other while being far from irrelevant communities. In recent years, generative adversarial networks (GANs) [10] have drawn great attention for their success in different applications [11]. It is worth mention that GANs can learn a map of an input from a simple distribution to a complicated one (i.e., mapping into an embedding space) [12]. There have been some studies on integrating GANs into NE. For example, GraphGAN [13] is proposed to unify generative and discriminative models for embedding learning to boost the performance of NE. Inspired by DeepWalk [2], AIDW [14] proposes an improved version with a GAN-based regularization method. A-RNE [15] leverages GANs with triplet ranking loss to generate high-quality negative vertices during the training. ProGAN [12] uses GANs to discovers ordinary underlying vertex proximities for benefiting NE. However, these above models employ standard GANs for generating vertex samples while ignoring community structures. Therefore, it still remains to be tackled for incorporating community information into GANs.

To solve this problem, we introduce an adversarial learning method (dubbed as ACNE) for NE and overlapping community detection. More concretely, both vertices and communities are represented as learnable embeddings. First, we sample a community from the walking paths of each vertex. Next, we need to map the relation of a discrete vertex community assignment into a continuous vertex–community embedding space. To achieve this purpose, we exploit a discriminator to train the embeddings that jointly maximize the predicting probabilities of the assigned community and the context vertices for a target vertex. Through this fashion, the correlation of vertex–vertex from local connectivity and vertex–community from network structure can be uniformly preserved in our method. After that, to obtain distinguishable embeddings, we employ a softmax generator for constructing high-quality negative communities instead of using a uniform sampling method. As such, the pairs of a given target vertex and its negative communities with higher probabilities computed in the discriminator will be promoted to be sampled. Finally, to evaluate if simultaneously taking vertex connectivity and community structure into account can actually benefit the NE performance, we carry out experiments including vertex classification and overlapping community detection on several real-world datasets to compare the performance of our methods with state-of-the-art models.

Moreover, we further study ACNE to improve its performance by considering the following two aspects.

1) Though the proposed adversarial learning method can outperform the baseline models on the aforementioned tasks, one concern may prevent it from achieving better performance. Because ACNE leverages random walk to sample the context of vertices, the generated paths may fall into local loops: it is very likely for a vertex walking back to previously visited nodes when moving to its neighbors uniformly at random [16], [17]. To avoid the paths being trapped into local communities and benefit from overlapping community detection, we adopt a walking strategy with perception to obtain paths containing more possible boundary vertices.
2) Besides, after performing representation learning with ACNE, we apply soft community assignments from a simple classifier as supervision to update the weights of ACNE. The proposed self-training method, referred to as ACNE-ST, can fine-tune ACNE to learn more discriminative representations.

The contributions can be summarized as follows.

1) We introduce an adversarial learning method dubbed as ACNE that can incorporate overlapping community information into network representation learning.
2) The proposed generator can yield high-quality negative communities that can help the discriminator learn more discriminative representations.
3) We employ a walking strategy with perception to benefit overlapping community detection that can generate paths containing more possible boundary vertices. Moreover, we further introduce a self-training method named ACNE-ST to achieve better performance results.
4) We empirically evaluate both ACNE and ACNE-ST with subsequent analysis tasks, including vertex classification and overlapping community detection. Our experiments show that the proposed methods can achieve more superior performance than state-of-the-art models.

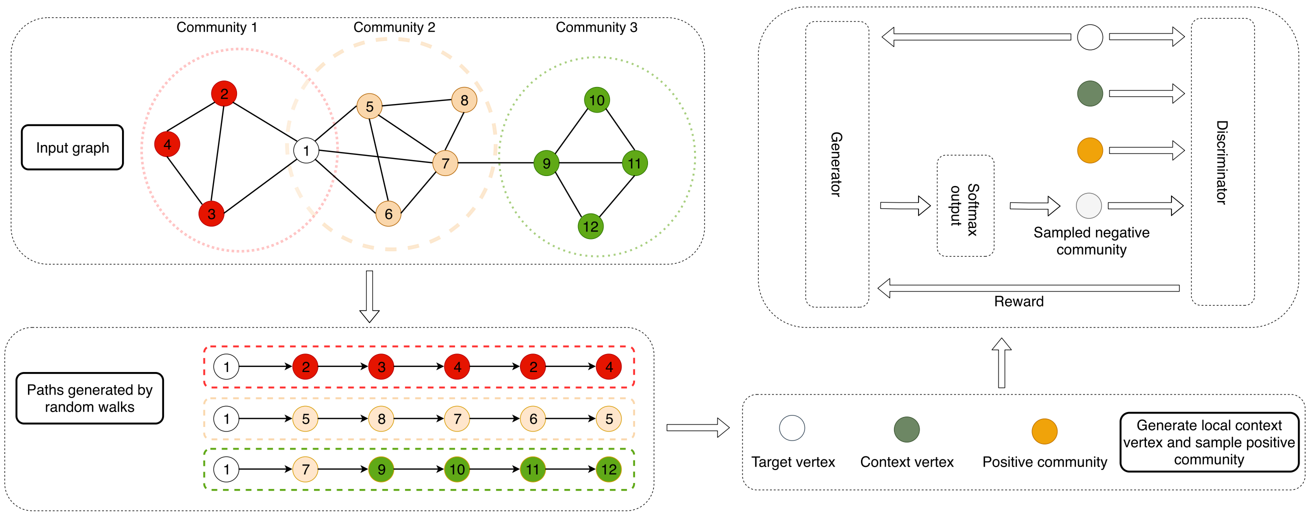


Fig. 2. Overall training process of ACNE.

The rest sections are organized as follows. In Section II, we firstly present the core idea of our proposed ACNE model and then demonstrate the self-training enhanced ACNE-ST model. After that, we present their complexity analysis in Section III. We analyze experimental results in Section IV and introduce related work in Section V. Section VI summarizes our work.

II. Proposed Models

In this part, we will give the problem formulation and notations. Then, we will introduce an overview of the ACNE framework, as shown in Fig. 2, and provide detailed descriptions of its components.

A. Problem Formulation With Notations

In general, we aim to perform overlapping community detection and NE learning; thus, the problem can be formulated as follows.

1) Network Embedding: A network can be presented as $G = (V, E)$, where V denotes a set of vertices and $E \subseteq V \times V$ is a set of edges. Generally, NE aims to preserve the network proximity of each vertex $v \in V$ in a low-dimensional embedding, where $\mathbf{v} \in \mathbb{R}^d$ and $d \ll |V|$ is the embedding dimension.

2) Community Embedding: We denote the number of communities as $|C|$. Because mapping vertex–vertex and vertex–community relations into the same embedding space is conducive to integrate the information of local connectivity and network structure, we also learn a low-dimensional representation $\mathbf{c} \in \mathbb{R}^d$ for each community, which has the same dimension as vertex embeddings. In this way, we can measure the similarity of communities and vertices by calculating the inner product of their corresponding embeddings, which will benefit overlapping community detection.

3) Random Walk: In a given network, the random walk method is used for exploring the neighborhood of vertices [2], [14], [18], [19]. Then, we can obtain a set of vertex sequences that contain semantic relations among vertices. These sequences can be denoted as $S = \{s_1, \ldots, s_N\}$, where N is the total number of walk sequences. Each sequence has $s = \{v_1, \ldots, v_{|s|}\}$.

B. Walking With Perception

To explore the neighbors of vertices, we first generate paths from a network G. For example, as shown on the left-hand side of Fig. 2, starting with vertex 1, multiple paths can be generated with random walks. However, it is very likely for a vertex walking back to previously visited ones when moving at random [16], [17]. The first path is trapped into community 1 (marked in red). Therefore, we aim to generate the paths containing more possible boundary vertices to benefit overlapping community detection. We adopt a walking strategy with perception [16] in the path generation. Given a current vertex u, the selection probability of a next-hop vertex v is defined as follows:

$$q_{uv} = 1 - \frac{Com_{\mathrm{uv}}}{\min\{\deg(u), \deg(v)\}} \tag{1}$$

where q_{uv} is the probability of v being selected as the next-hop vertex, Com_{uv} denotes the number of common neighbors between u and v, and $\deg(\cdot)$ represents the number of degrees. From (1), we can see that the current vertex u is more likely to choose the one with less common neighbors as the next-hop vertex. This walking strategy can discover more possible boundary vertices, which may benefit the overlapping community detection. Note that the rejection probability would be defined as $1 - q_{uv}$. Then, more paths, such as the third one in the lower left of Fig. 2 (containing communities marked in light-yellow and green), can be generated.

To give a more detailed demonstration of the effects of walking with perception, we provide Fig. 3 to show the probabilities of Vertex 1 choosing the next nodes. The perception method has a higher probability (12/37) to access the boundary Vertex 6 comparing with the random one (1/5).

Note that the normalization of (1) in [16] aims to avoid backtracking because returning to the currently visiting node may introduce extra overhead and slow down the convergence. In practice, we find that it does not make too much difference

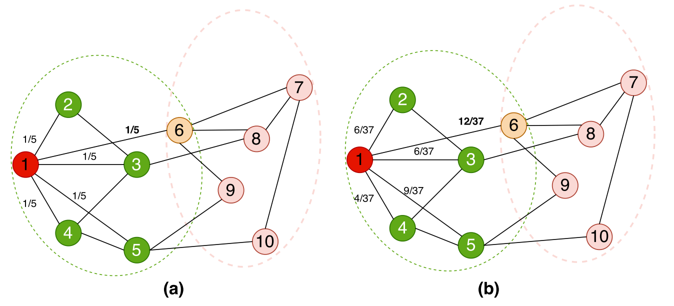


Fig. 3. Comparison of (a) randomly walking and (b) walking with perception. The perception method has a higher probability to access the boundary vertex 6.

in the performance of the downstream tasks. We conjecture that: 1) the probability of returning to the revisited node is small and 2) the revisited node does not hinder discovering more possible boundary vertices for the model. As such, we can save the computation of the normalization step in our algorithm.

C. Context-Aware Community Assignment

After conducting the walking strategy, we can obtain paths (or call walk sequences), as shown at the lower left-hand side of Fig. 2. Besides, as mentioned in the introduction, paths stretching out from a vertex to its n-step neighbors can reflect the expression of its communities in network structures. Thus, the vertex community assignment in a generated sequence is related to its context vertices. As such, with the Gibbs sampling method [20], we can estimate the conditional probability of community assignments for each given vertex and its associated sentence as follows:

$$p(c|v,s) \propto p(c|s_{\neg v}) \prod_{\hat{v}\neq v,\hat{v}\in s} p(c|\hat{v}) \tag{2}$$

where $s_{\neg v}$ represents the sequence s except vertex v, $p(c|s_{\neg v})$ denotes the conditional probability of a community c given a sequence s, and $p(c|\hat{v})$ is the vertex–community distribution of v (we assume it as prior knowledge generated from global network structures, and the details will be given later).

Then, we can further derive $p(c|s_{\neg v})$ of (2) as follows:

$$p(c|s_{\neg v}) = \frac{N(c,s_{\neg v})}{\sum_{\hat{c}}^{C} N(\hat{c},s_{\neg v})} \tag{3}$$

where $N(c,s_{\neg v})$ denotes the number of vertices belonging to community c in sequence s except current vertex v, and C represents the whole set of communities.

Next, we will introduce how we learn the vertex–community distributions mentioned before. First, we use $\mathbf{M} \in \mathbb{R}^{V\times V}$ to denote the symmetric adjacency matrix of a network. Then, we can employ nonnegative matrix factorization (NMF) to learn the vertex–community distribution [21], [22], which is formulated by

$$\min_{\mathbf{W}\geq 0} ||\mathbf{M} - \mathbf{W}\cdot\mathbf{W}^T||_F^2 + \alpha||\mathbf{W}||_F^2 \tag{4}$$

where $||\cdot||_F$ represents the Frobenius norm of a matrix, $\mathbf{W} \in \mathbb{R}^{V\times C}$ denotes the learned vertex–community distributions that encode the global understanding of network structures, and α is a harmonic factor for regularization.

After that, the last part of (2), i.e., $p(c|\hat{v})$, the probability of vertex $\hat{v}$ assigned to community c, can be computed by

$$p(c|\hat{v}) = \frac{\mathbf{W}_{\hat{v},c}}{\sum_{\hat{c}}^{C} \mathbf{W}_{\hat{v},\hat{c}}}. \tag{5}$$

Note that here we do not directly use $\mathbf{W}$ as the final results for community detection or NEs. Instead, when learning the representations of vertices and communities, we aim to preserve the correlation of vertex–community from global network structure and vertex–vertex from local connectivity into the same embedding space. To achieve this goal, we first sample a community from its context vertices (or called a vertex sequence of walking paths). Then, we adopt a discriminator to learn the vertex embeddings through maximizing the probabilities of predicting both discrete assigned community and context vertices. The experimental results demonstrate that such a jointly learning manner can achieve better performance than the separate learning ones (for comparison, we will introduce the results of jointly modeling and the simple NE, respectively). The details will be shown in Section IV-D.

D. Adversarial Learning for Vertices and Communities

Though we can map the discrete results learned from generative models into a continuous embedding space with the GAN techniques, the standard GANs is generally designed to generate samples of vertices rather than communities. Therefore, how to generate the samples of underlying communities is still a challenging problem.

Generator G: Given a target vertex, we design the generator of ACNE to generate high-quality negative communities. More concretely, we use a softmax function on a set of negative vertices. The formulation is given as follows:

$$G(c_n|v_t;\theta_G) = \frac{\exp(\mathbf{c}_n\cdot\mathbf{v}_t^{\mathrm{T}})}{\sum_{\hat{c}\in C}\exp(\hat{\mathbf{c}}\cdot\mathbf{v}_t^{\mathrm{T}})} \tag{6}$$

where c_n is the possible negative community, v_t denotes a given target vertex, θ_G represents the parameters of vertices and communities in the generator, and $|C|$ denotes a predefined size of the communities. Since we usually set $|C| \ll |V|$, the summation term of the denominator in (6) only takes slight expense of computation. Generally speaking, the designed generator aims to sample high-quality negative communities with the probability calculation $G(c_n|v_t;\theta_G)$ instead of uniform sampling that possibly generates entirely unrelated communities: the gradient calculated by the inner product of the embeddings could be a very small number even close to zero. With (6), the loss function of the generator can be further formulated by

$$\mathcal{L}_G = \sum_{v_t\in\mathcal{B}} \mathbb{E}_{c_n\sim G(\cdot|v_t;\theta_G)} D(c_n,v_t;\theta_D) \tag{7}$$

where $\mathcal{B}$ represents a batch of target vertices, θ_D denotes the parameters of vertices and communities in the discriminator, and $D(\cdot)$ indicates the sigmoid function $\sigma(\cdot)$ where $D(c_n,v_t;\theta_D) = \sigma(\mathbf{c}_n\cdot\mathbf{v_t}^{\mathrm{T}}) = (1/(1+\exp(-\mathbf{c}_n\cdot\mathbf{v_t}^{\mathrm{T}})))$.

Moreover, note that the output of the generator [see (6)] is a discrete index of the communities. As such, the stochastic gradient descent (SGD) method cannot be directly utilized for the optimization. Based on [23], [24], the policy gradient-based reinforcement learning method can be used to optimize the generator as follows:

$$\begin{aligned}\nabla_{\theta_G}\mathcal{L}_G &= \nabla_{\theta_G}\sum_{v_t\in\mathcal{B}}\mathbb{E}_{c_n\sim G(\cdot|v_t;\theta_G)}D(c_n,v_t;\theta_D)\\ &= \sum_{v_t\in\mathcal{B}}\mathbb{E}_{c_n\sim G(\cdot|v_t;\theta_G)}D(c_n,v_t;\theta_D)\nabla_{\theta_G}\log G(c_n|v_t;\theta_G)\end{aligned}\tag{8}$$

where the gradient of $\mathcal{L}_G$ is an expected summation of $\nabla_{\theta_G}\log G(c_n|v_t;\theta_G)$ and its weights $D(\cdot)$. Here, we assume $D(\cdot)$ of (8) as the reward function, and we aim to maximize the expected rewards. In general, the policy gradient loss is adopted to maximize the margin between the target vertex and its assigned negative communities. Specifically, for each pair of (c_n, v_t), this policy will punish trivial negative communities by lowering down their co-occurring probabilities. At the same time, the policy will encourage the discriminator to sample high-quality negative communities, i.e., pair (c_n, v_t) with higher similarity parameterized by θ_D is encouraged to be generated.

Besides, the reinforcement-based methods usually suffer from unstable performance and, thus, have high variance results [25] in practice. Therefore, we refer to [26] and [27] to alleviate this problem by adding a baseline function to the reward term $D(\cdot)$. More concretely, we replace it by

$$D(c_n,v_t;\theta_D)+\frac{\sum_{\mathcal{B}\in\mathcal{P}}\sum_{v_t\in\mathcal{B}}\mathbb{E}_{c_n\sim G(\cdot|v_t;\theta_G)}D(c_n,v_t;\theta_D)}{|\mathcal{P}|}\tag{9}$$

where the second term of (9) is the baseline function, i.e., the average reward in the training process that can be used to reduce learning volatility [26], [27], and $\mathcal{P}$ represents the overall training batches.

Discriminator D: We design a discriminator to achieve the following two goals: 1) the first goal is to make the context vertices being close in the embedding space while being far from their negative vertices (similar to the Skip-gram method [5], [6]) and 2) the second one is to satisfy $D(c_p, v_t) \gg D(c_n, v_t)$, where c_p, v_t, and c_n indicate positive community, target vertex, and negative community, respectively.

To fulfill the above two goals, we aim to learn the target vertex representation by jointly maximizing the probabilities of predicting its context vertices and assigned community. As shown on the right-hand side of Fig. 2, the learning of each target vertex v_t and its associated tuple $\{v_c, v_n, c_p, c_n\}$ in a given sequence $s \in \mathcal{B}$ can be formalized by

$$\begin{aligned}\mathcal{L}_D = \sum_{s\in\mathcal{B}}\sum_{v_t\in s}&\mathbb{E}_{c_n\sim G(\cdot|v_t;\theta_G)}\\ &\times\big[\log D(v_t,v_c)\\ &\quad+\log D(-v_t,v_n)+\log D(v_t,c_p)+\log D(-v_t,c_n)\\ &\quad+\lambda\big(||\mathbf{v}_t||_F^2+||\mathbf{v}_c||_F^2+||\mathbf{v}_n||_F^2+||\mathbf{c}_p||_F^2+||\mathbf{c}_n||_F^2\big)\big]\end{aligned}\tag{10}$$

Algorithm 1 ACNE-ST Training Process

```
Input: Graph $G = (V, E)$, training batch $\mathcal{B}$ of vertex
       pairs, dimension $d$ of embeddings, community set
       $C$
Result: Parameters $\theta_D$ and $\theta_G$ in the discriminator and
        the generator
1  begin
2      Randomly initialize $\theta_D$ and $\theta_G$;
3      Using the walking with perception method [16] to
       obtain walk sequences $S$;
4      For each vertex in the sequences, we sample its
       community assignment using Eq. (2);
5      Start the first run of ACNE;
6      while not converge do
7          Generate a batch $\mathcal{B}$ from walk sequences $S$;
8          for n-steps in the generator do
9              For each target vertex $v_t$, we sample negative
               communities with Eq. (6);
10             Using policy gradient with the formulated loss
               Eq. (8) to update the parameters $\theta_G$;
11         end
12         for n-steps in the discriminator do
13             For each target vertex $v_t$, we compute $D$ loss
               with Eq. (10);
14             Using gradient descent to update the parameters
               $\theta_D$;
15         end
16     end
17     Build up a classifier to obtain the probabilities of
       community assignments of vertices;
18     For each vertex in the sequences, its community
       assignment would be either the prediction of the
       classifier if the probability is larger than $\delta$, otherwise
       we sample a community assignment using Eq. (2);
19     Go back to Step 6 for a further run;
20 end
```

where v_c and v_t form a vertex pair, v_n is the negative vertex of v_t sampled by negative sampling [5], c_p is the positive community generated by (2), c_n is its negative community sampled by (6), $||\cdot||_F$ is the Frobenius norm of embeddings, and λ is a harmonic factor used for regularization. Note that, here, $\{\mathbf{v}_t, \mathbf{v}_c, \mathbf{v}_n, \mathbf{c}_p, \mathbf{c}_n\}$ are the parameters of the discriminator (i.e., θ_D), which can be optimized with the general gradient descent technique.

E. Proposed ACNE-ST Training Process

Our proposed ACNE-ST training process can be summarized in Algorithm 1. To begin with, we generate walk sequences S via the walking with perception method [16] (Step 3). Then, for each vertex in a sequence, we first sample its community assignment using (2) (Step 4). After that, we start the first run of ACNE (from step 6 to step 16). Similar to the learning process in [13], [28], and [29], we first fix the parameters θ_D of the discriminator and train the generator with the training set. The purpose of the generator

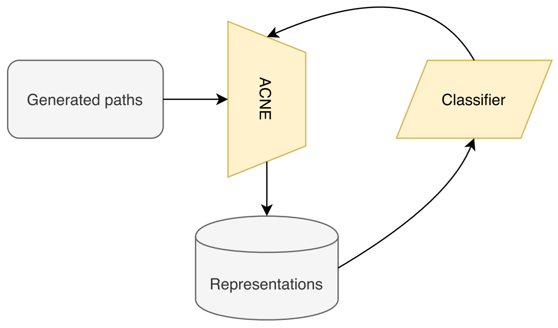


Fig. 4. Self-training enhanced ACNE-ST model.

is to search for high-quality negative communities as the inputs of the discriminator. After that, we fix the parameters θ_G of the generator and train the discriminator. Note that both the generator and the discriminator are trained with the Adam gradient descent method [30]. We use L_2 regularization in the training. After converging, we adopt the parameters learned by the discriminator as the final vertex and community embeddings. Besides, we build up a simple classifier by using Liblinear package [31] with its default settings to obtain the probabilities of community assignments of vertices (step 17). For each vertex in a sequence, we adopt its community assignment when the probability is larger than a threshold δ (we set it as 0.9 in experiments). Otherwise, we sample its community assignment using (2) (Step 18). Finally, we go back to Step 6 for the second run. In general, we totally run twice of ACNE. The diagram of the self-training method ACNE-ST is shown in Fig. 4. We adopt it to fine-tune ACNE to learn more discriminative representations. The detailed experimental settings are demonstrated in Section IV.

III. Complexity Analysis

Our methods, including ACNE and ACNE-ST, aim to learn the following embeddings: θ_G and θ_D that represent the parameters of vertex and community representations in the generator and discriminator, respectively. In general, the optimization complexity of ACNE is $O(2L|V||C|wl)$, where L is the loop size, $|V|$ is the number of vertices, $|C|$ is the community size, w is the random walk sequences of each vertex, and l is the sequence length. The complexity of ACNE-ST is $O(2L|V||C|wl + |V|d^2)$, where d is the dimension size and $O(|V|d^2)$ is the complexity of vertex classification.

IV. Experiments

In this section, we aim to evaluate the learned vertex and community embeddings on several real-world datasets in terms of the subsequent tasks including vertex classification and overlapping community detection.

A. Experimental Datasets

We carry out experiments on four widely used datasets where the statistics are shown in Table I.

Cora[1] is a citation network collected in [32]. It has 2708 nodes, 5278 edges, and seven labels.

[1]https://people.cs.umass.edu/∼mccallum/data.html

TABLE I
Statistics of Data

Datasets	$\vert E\vert$	$\vert V\vert$	$\vert L\vert$
Cora	5,278	2,708	7
Citeseer	4,551	3,264	6
Wiki	12,761	2,405	19
DBLP_C4	52,914	17,725	4

Citeseer[2] is another widely used citation network that has 3264 nodes, 4551 edges, and six labels.

Wiki[3] is a language network collected by LBC[4] project. It has 2405 nodes, 12 761 edges, and 19 groups.

DBLP_C4[5] is a citation network on computer science field that is collected by Tang *et al.* [33]. We choose four subfields, including data mining, database, CV, and AI for experiments.

B. Baselines

The baseline models can be introduced based on the following four groups.

1) General NE: DeepWalk [2] proposes a two-step method: employ truncated random walks on a given network to obtain vertex sequences; then adopt the Skip-gram method [5] to learn vertex representations. Node2vec [3] extends DeepWalk with a biased random walk to search the network. LINE [4] introduces the first- and second-order vertex proximity preservation methods during the embedding learning. SDNE [34] is the first work to utilize a deep neural network. GraRep [35] exploits a matrix factorization technique (SVD) in network representation learning.

2) GAN-Based NE: GraphGAN [13] proposes a unified method that incorporates the generative and discriminative models to boost NE performance. AIDW [14] introduces a regularized GAN-based method that can be considered as an inductive version of DeepWalk. ARNE [15] aims to construct high-quality negative vertices to achieve better results in the downstream tasks.

3) General Community Detection: LC [36] introduces linkage relations among communities. SCP [37] proposes to search for adjacent cliques to detect communities. MDL [38] presents a minimum description length way to carry out clustering. BigCLAM [39] exploits an NMF for overlapping and hierarchically nested community detection. Similarly, NMF [21] also uses matrix factorization to obtain vertex–community distribution in a global understanding of network structures. Note that we employ NMF as one of the baselines because we use it to obtain prior knowledge [refer to (2)].

4) Jointly Modeling: MNMF [40] proposes a matrix factorization method that can jointly detect nonoverlapping communities and learn NEs. ComE [41] also introduces a jointly learning manner by using multivariate Gaussian distributions to detect communities. More recently, PolyDeepwalk [42] presents a polysemous embedding approach that can model

[2]https://github.com/wonniu/AdvT4NE_WWW2019
[3]https://github.com/albertyang33/TADW
[4]https://linqs.soe.ucsc.edu/
[5]http://arnetminer.org/citation (V4 version is used)

TABLE II
VERTEX CLASSIFICATION ACCURACY (%) ON CORA

% Label Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	64.60	69.85	74.21	76.68	77.59	77.68	78.63	79.35	79.23
LINE	66.59	66.06	72.25	73.94	74.03	74.65	75.12	75.30	75.76
Node2vec	73.96	78.04	80.07	81.62	82.16	82.25	82.85	84.02	84.91
SDNE	70.97	75.08	76.90	77.82	78.26	79.11	79.37	79.46	79.37
GraRep	74.98	77.48	78.57	79.38	79.53	79.68	79.75	80.89	80.74
AIDW	73.83	77.93	79.43	81.16	81.79	82.27	82.93	84.11	83.69
GraphGAN	76.43	79.14	81.62	81.91	82.12	82.83	83.28	84.65	84.93
ARNE	68.09	72.86	75.14	75.83	76.97	77.30	79.22	78.90	78.43
MNMF	75.08	77.85	79.05	79.53	79.82	80.21	79.98	80.11	79.41
ComE	76.72	79.25	80.73	80.97	81.53	82.10	82.19	82.42	82.65
PolyDeepwalk	76.00	79.51	80.49	81.06	81.46	81.73	84.02	84.76	83.66
CNE	78.17	81.67	82.70	82.89	83.97	**84.68**	**85.91**	85.60	85.97
ACNE	**78.84**	**81.82**	**83.49**	**83.69**	**84.05**	**84.59**	85.85	**86.35**	**88.19**
ACNE-ST	**78.87**	**83.20**	**83.39**	**83.94**	**84.44**	84.41	**85.93**	**86.72**	**88.30**

TABLE III
VERTEX CLASSIFICATION ACCURACY (%) ON CITESEER

% Label Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	45.53	50.98	53.79	55.25	56.05	56.84	57.36	58.15	59.11
LINE	47.03	50.09	52.71	53.52	54.20	55.42	55.87	55.93	57.22
Node2vec	50.78	55.89	57.93	58.60	59.44	59.97	60.32	60.75	61.04
SDNE	47.35	51.10	52.45	53.20	53.70	54.20	54.79	55.26	54.46
GraRep	50.60	53.56	54.63	55.44	55.20	55.07	56.04	55.48	56.39
AIDW	50.77	54.82	56.96	58.04	59.65	60.03	60.99	61.18	62.84
GraphGAN	53.68	56.28	57.77	59.52	59.74	59.34	58.06	57.58	56.27
ARNE	54.12	56.09	56.46	56.92	56.99	57.81	57.55	56.97	52.60
MNMF	51.62	53.80	55.47	56.94	56.81	57.04	57.05	57.00	57.22
ComE	54.71	57.70	58.84	59.67	59.93	60.30	61.12	61.62	61.11
PolyDeepwalk	52.25	53.75	56.46	56.92	57.48	57.19	58.43	58.66	58.98
CNE	**56.33**	**59.57**	**62.88**	63.75	65.01	66.84	67.02	67.32	64.22
ACNE	55.34	58.46	62.80	**65.08**	**65.63**	**67.08**	**67.14**	**67.99**	**68.50**
ACNE-ST	**56.40**	**59.53**	**64.63**	**66.72**	**67.22**	**68.30**	**68.78**	**69.06**	**69.64**

multiple facets of vertices by mapping each facet to an embedding.

Moreover, we take CNE as a variant of ACNE for an ablation study. Specifically, CNE omits the generator component in adversarial training, so as to sample negative communities with a uniform distribution.

C. Experimental Settings and Metrics

We apply random walks on the datasets for the preprocessing where the walk length, window size, and the number of walks are set to 30, 10, and 50, respectively. For the embedding dimension setting, we employ a grid-search method by varying the dimension $d \in \{128, 200, 256, 300, 400\}$ for reference. For other trivial parameters in the baselines, we adopt their preferred settings in the articles. Moreover, we utilize the Adam optimizer [30] to optimize our methods. The initial rate of Adam is set to 1e-3. For the parameter λ of (10) and δ mentioned in Section II-E, we set them as 1e-5 and 0.9, respectively. In the subsequent tasks of vertex classification, we build a classifier by using the Liblinear package [31] with its default settings and use accuracy [14] as metrics. Besides, we employ a modified modularity [43] to evaluate the results of overlapping community detection.

D. Evaluation of Vertex Classification

In this part, we report the results of classification accuracy performance for all baselines with various training ratios. The details are shown in Tables II–VI, where the highest and second-place scores are highlighted in boldface. Note that, here, we do not include the baselines in the general community detection group since they are not specially designed for NE. We omit NMF because its accuracy scores are much lower than the others. From these tables, we can observe the following.

1) Our proposed methods can consistently achieve better performance than the state-of-the-art models on all datasets. These results validate the effectiveness of our methods for integrating overlapping community information into NE. Moreover, the proposed self-train method, ACNE-ST, can achieve better performance than ACNE and CNE, indicating that the NE can further benefit from the incorporation of the walking strategy and the soft community assignments.
2) More concretely, ACNE outperforms the jointly learning methods (i.e., MNMF, ComE, and PolyDeepwalk) with our proposed adversarial learning targets. Especially, for the GAN-based learning methods (i.e., AIDW, GraphGAN, and ARNE), ACNE also achieves

TABLE IV
VERTEX CLASSIFICATION ACCURACY (%) ON WIKI

% Label Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	46.60	54.48	59.05	62.70	64.66	65.95	66.98	68.37	68.78
LINE	57.88	61.08	63.50	64.68	66.29	66.91	67.43	67.46	68.61
Node2vec	55.94	59.67	61.11	64.21	65.08	65.58	66.76	67.19	68.73
SDNE	52.42	57.34	60.15	62.35	63.18	64.21	64.71	65.63	65.60
GraRep	58.57	61.91	63.58	63.77	64.68	65.39	65.92	65.18	67.05
AIDW	57.32	61.84	63.54	64.90	65.58	66.54	65.59	66.58	68.02
GraphGAN	57.97	62.57	63.79	65.39	66.01	66.67	67.83	68.23	68.87
ARNE	58.43	60.45	62.23	62.44	62.59	62.89	62.47	62.79	62.66
MNMF	54.76	58.82	60.43	61.66	62.74	63.23	63.46	63.45	64.77
ComE	59.11	62.46	64.38	65.45	65.98	67.38	67.49	67.92	67.89
PolyDeepwalk	56.44	61.90	62.59	63.76	64.09	64.35	64.96	65.90	66.49
CNE	58.89	63.38	**65.20**	**66.87**	67.31	67.85	68.28	**69.23**	68.46
ACNE	**59.68**	**63.41**	64.45	66.11	**67.41**	**68.09**	**68.98**	**69.65**	**69.71**
ACNE-ST	**59.77**	**63.93**	**65.20**	**67.50**	**67.58**	**68.08**	**69.11**	69.19	**69.21**

TABLE V
MICRO-F1 AND MACRO-F1 (%) OF VERTEX CLASSIFICATION ON CORA

Method	30%		50%		70%		90%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GraphGAN	80.30	80.72	81.19	80.84	82.83	83.02	83.78	81.86
PolyDeepwalk	79.51	80.12	80.34	80.80	83.65	84.13	82.14	81.55
ACNE	82.38	81.56	82.20	81.72	85.61	84.51	**86.72**	**86.09**
ACNE-ST	**82.81**	**82.30**	**83.60**	**82.06**	**85.63**	**84.56**	85.61	84.63

TABLE VI
MICRO-F1 AND MACRO-F1 (%) OF VERTEX CLASSIFICATION ON CITESEER

Method	30%		50%		70%		90%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GraphGAN	57.35	52.27	58.83	53.14	57.56	51.82	56.25	52.14
PolyDeepwalk	57.02	52.25	58.52	53.10	57.24	51.82	55.96	52.08
ACNE	63.28	59.42	66.12	62.27	**68.47**	**65.66**	66.06	63.17
ACNE-ST	**65.21**	**60.85**	**66.79**	**63.22**	66.73	63.99	**67.75**	**69.08**

improvements, which we conjecture the reason is that they ignore community information in the learning.

3) Besides, to make a more comprehensive comparison, we adopt the metrics of Micro-F1 and Macro-F1 to evaluate the classification performance. For space limitation, we select GraphGAN and PolyDeepWalk that perform well in Tables II and III among the baselines for further comparison. From Tables V and VI, we can observe that our methods consistently outperform the baselines, obtaining about 3.7% and 14.1% improvements on average in Cora and Citeseer, respectively.

In addition, for DBLP_C4, we adopt smaller training ratios, so as to evaluate the performance of ACNE under sparse scenes and accelerate the training speed of classifiers. As shown in Fig. 5, we select PolyDeepwalk, Node2vec, and GraphGAN that achieve the best performance from each baseline group for further presentation. In general, ACNE can still obtain improvements in sparse situations.

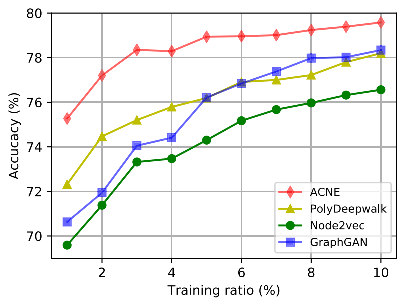


Fig. 5. Vertex classification accuracy of DBLP_C4.

E. Evaluation of Community Detection

In this part, we evaluate the performance of community detection by comparing it with the methods in jointly modeling and general community detection groups. The results are reported in Table VII where we can observe the following.

TABLE VII
MODULARITY RESULTS OF COMMUNITY DETECTION

Datasets	LC	SCP	BigCLAM	MDL	MNMF	NMF	PolyDeepwalk	ComE	CNE	**ACNE**	**ACNE-ST**
Cora	0.544	0.142	0.796	0.771	0.832	0.759	0.964	0.912	1.317	**1.334**	**1.338**
Citeseer	0.457	0.068	0.649	0.469	1.123	1.037	1.151	1.192	1.242	**1.276**	**1.318**
Wiki	0.480	0.113	0.522	0.550	1.413	1.303	1.576	1.614	1.705	**1.734**	**1.742**
DBLP_C4	0.586	0.067	0.616	0.685	0.687	0.651	0.693	0.701	0.757	**0.779**	**0.780**

TABLE VIII
ABLATION STUDY OF WALKING PERCEPTION

Method	Modularity		
	Cora	Citeseer	Wiki
ACNE-ST w/o	1.289	1.255	1.660
walking perception	-3.66%	-4.78%	-4.71%

1) The proposed methods significantly outperform the other baselines, which demonstrates that our methods can detect more meaningful communities. These results also verify the efficacy of the designed generator in generating high-quality negative communities. In addition, ACNE-ST is able to obtain better detection results than ACNE, which demonstrates its fine-tuned capability.
2) ACNE can obtain better performance than NMF, which verifies the superiority of ACNE comes beyond the prior knowledge learning from NMF (referring to Section II-C). These results conform to our assumption: the vertex community assignment is associated with its context.

To sum up, the tasks of vertex classification and overlapping community detection demonstrate the efficacy of our methods for jointly modeling vertex connectivity and community structures with the GAN technique.

F. Analysis on Walking With Perception

To evaluate the impact of walking strategy on perception, we carry out an ablation study to show the benefits of detecting overlapping communities. As shown in Table VIII, the community detection performance decreases by 3.66%, 4.78%, and 4.71% on Cora, Citeseer, and Wiki, respectively, proving the effectiveness of the walking strategy.

G. Influences of Parameters

In this section, we aim to investigate the influences of the key parameters on the performance of the proposed methods in terms of vertex classification and community detection.

1) Dimension d: We first vary the number of the dimension setting in $\{128, 200, 256, 300, 400\}$ to evaluate the ACNE performance, where the experimental results are shown in Fig. 6. Specifically, Fig. 6(a) shows that ACNE can obtain relatively stable performance for classification accuracy on the Cora, Citeseer, and Wiki datasets, respectively. The curve of Citeseer first increases with dimension size at the beginning and then becomes comparatively smooth after $d = 256$. Fig. 6(b) reports the modularity performance on the dimension setting,

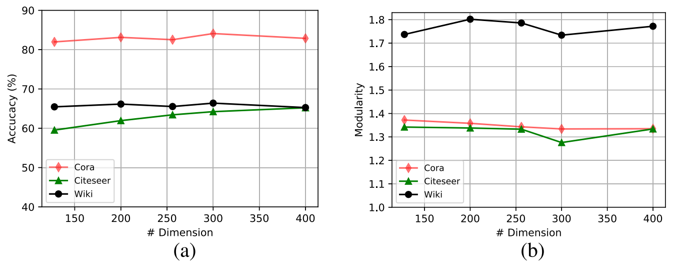


Fig. 6. Sensitivity analysis of dimension in ACNE. (a) Accuracy versus dimension. (b) Modularity versus dimension.

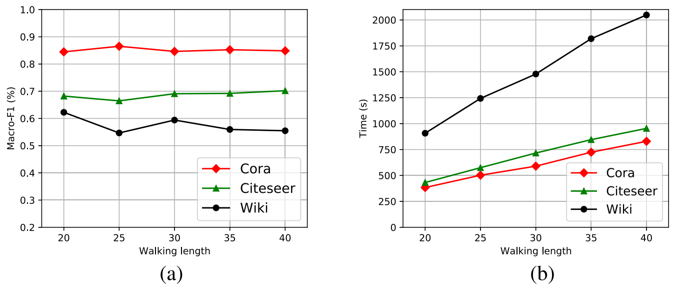


Fig. 7. Sensitivity analysis of walking length in ACNE-ST. (a) Macro-F1 versus walking length. (b) Time versus walking length.

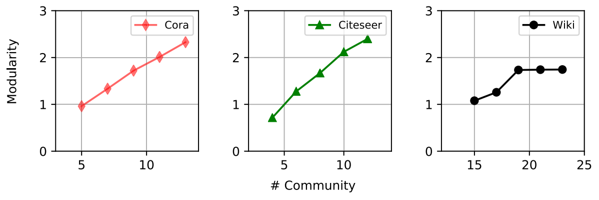


Fig. 8. Influence of modularity on community setting.

where we can observe that the curve of the dataset fluctuates slightly (about 0.03 points on Cora, 0.05 points on Citeseer, and 0.08 points on Wiki).

2) Walking Length: We conduct a sensitivity analysis of the walking length on classification performance and running time, as shown in Fig. 7. We vary the length of random walk in $\{20, 25, 30, 35, 40\}$. From Fig. 7(a), we can observe that the classification performance of ACNE-ST is relatively robust to the walking length on the Cora, Citeseer, and Wiki datasets. Besides, Fig. 7(b) shows that the running time is linear with the walking length, which is expected from the complexity analysis mentioned in Section III.

3) Community C: We estimate the influences of the community number C on the ACNE performance. Concretely, we vary C in the range of $\{5, 7, 9, 11, 13\}$, $\{4, 6, 8, 10, 12\}$, and $\{15, 17, 19, 21, 23\}$ on Cora, Citeseer, and Wiki, respectively.

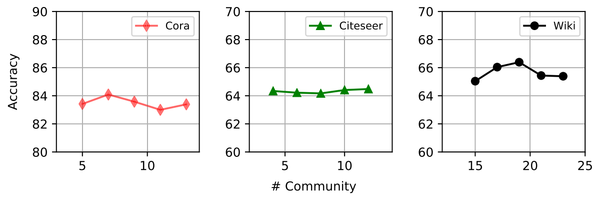


Fig. 9. Influence of accuracy on community setting.

Generally, Fig. 9 demonstrates that ACNE achieves the best performance when C is set to 7 and 19 on Cora and Wiki correspondingly. On the Citeseer dataset, ACNE can gain stable performance when C is around 7. The above results of community settings generally match with the ground-truth numbers in the datasets, which implies that our methods can dynamically detect the community number of the network. Moreover, Fig. 8 shows that the modularity curves grow with the number of communities. We conjecture that more fine-grained communities can be found when setting larger C, while the accuracy decreases for ACNE falling into the local optimum, as shown in Fig. 9. Thus, this is a tradeoff between learning more discriminative vertex embeddings and detecting more fine-grained communities.

V. Related Work

This work is based on the following categories: network representation learning, overlapping community detection, and jointly modeling.

A. Network Representation Learning

It is also known as NE that has received great attention in recent years. Ordering the work in time, Perozzi *et al.* [2] propose DeepWalk that first carries out truncated random walks to obtain vertex sequences and then applies the Skip-gram method [5] to learn vertex embeddings. After that, Tang *et al.* [4] advance LINE that utilizes breadth-first or DFS strategies to perform random walks. Then, node2vec [3] is introduced to combine both of these strategies by designing biased random walks. Besides, inspired by the development of GANs [10] that has achieved promising performances in various tasks [11], some GAN-based models are proposed for NE. For example, GraphGAN [13] first incorporates both generative and discriminative models into embedding learning. Then, AIDW [14] takes an inductive version of DeepWalk by adopting an adversarial method to regularize the learned embeddings. ARNE [15] leverages GANs to generate high-quality negative vertices to boost learning performance, whereas these mentioned models only learn the local connectivity of vertices while ignoring the global pattern, i.e., the communities in networks.

B. Community Detection

It is one of the critical tasks in social science [44]. The traditional methods usually only detect nonoverlapping communities; as such, they do not conform to real-world scenarios where each vertex may be related to various communities as it may play multiple roles. To solve this problem, some methods are proposed for overlapping community detection. For instance, SCP [37] advances a sequential algorithm for the detection. Then, LC [36] is proposed to employ a link-clustering algorithm that partitions the links of vertices for community detection. After that, MDL [38] introduces a minimum description way for overlapping group detection. Afterward, both NMF [21] and BigCLAM [39] utilize an NMF technique to learn vertex–community distributions and perform community assignments. However, all these methods are not specially designed for NE and ignore the local vertex connectivity.

C. Jointly Modeling

It aims to achieve two tasks simultaneously. MNMF [40] proposes a matrix factorization-based method that can jointly detect communities and learn NEs. Then, ComE [41] exploits a multivariate Gaussian approach to model communities in the learning. PolyDeepwalk [42] introduces a polysemous embedding method that regards multiple facets of vertices as the communities and maps the facets into embeddings. However, these mentioned models only consider the intra-community relations that the embeddings of vertices and their assigned communities are close to each other while neglecting the intercommunity relations that the embeddings of vertices and their irrelevant communities are far away. The proposed generator would generate high-quality negative communities that can help the discriminator to learn more discriminative vertex and community embeddings.

VI. Conclusion

In general, we propose: 1) an adversarial training model named ACNE for NE and overlapping community detection and 2) a self-training enhanced method ACNE-ST that can fine-tune ACNE to achieve better representation learning results. Most of the GAN-based methods for NE only exploit standard GANs to sample vertices instead of communities. As such, it is still a challenge to utilize community information in GANs. To solve this problem, in ACNE, we first generate different paths expanded from a vertex to its n-step neighbors, which may represent its communities. After that, we sample communities for vertices by using a context-aware community assignment method. At the same time, we design a softmax generator to obtain high-quality negative communities for learning. Finally, we employ a discriminator for jointly learning the community and vertex embeddings. After the representation learning, we apply a self-training mechanism called ACNE-ST by adopting soft community assignments from a simple classifier as supervision to update the weights of ACNE. Our experiments show that: 1) both ACNE and ACNE-ST can achieve better performance than state-of-the-art models and 2) ACNE-ST can fine-tune ACNE to obtain more discriminative representations.

Acknowledgment

This work aims to extend Adversarial Learning for Overlapping Community Detection and Network Embedding, Proceedings of 24th European Conference on Artificial Intelligence (ECAI), 2020.

References

[1] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.

[2] 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.

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

[4] 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.

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

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

[7] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[8] A. Perer, I. Guy, E. Uziel, I. Ronen, and M. Jacovi, "Visual social network analytics for relationship discovery in the enterprise," in *Proc. IEEE Conf. Vis. Analytics Sci. Technol. (VAST)*, Oct. 2011, pp. 71–79.

[9] J. Reisinger and R. J. Mooney, "Multi-prototype vector-space models of word meaning," in *Proc. Human Lang. Technol., Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 109–117.

[10] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[11] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," 2019, *arXiv:1906.01529*. [Online]. Available: http://arxiv.org/abs/1906.01529

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

[13] H. Wang *et al.*, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[14] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[15] Q. Dai, Q. Li, L. Zhang, and D. Wang, "Ranking network embedding via adversarial learning," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2019, pp. 27–39.

[16] Y. Li *et al.*, "Walking with perception: Efficient random walk sampling via common neighbor awareness," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 962–973.

[17] Z. Zhou, N. Zhang, Z. Gong, and G. Das, "Faster random walks by rewiring online social networks on-the-fly," *ACM Trans. Database Syst.*, vol. 40, no. 4, pp. 1–36, Feb. 2016.

[18] Q. Dai, X. Shen, L. Zhang, Q. Li, and D. Wang, "Adversarial training methods for network embedding," in *Proc. World Wide Web Conf.*, May 2019, pp. 329–339.

[19] C. Tu *et al.*, "A unified framework for community detection and network representation learning," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 6, pp. 1051–1065, Jun. 2019.

[20] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Acad. Sci. USA*, vol. 101, no. 1, pp. 5228–5235, 2004.

[21] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, Apr. 2012, pp. 106–117.

[22] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2003, pp. 267–273.

[23] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3528–3536.

[24] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017.

[25] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

[26] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[27] J. Wang *et al.*, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 515–524.

[28] L. Cai and W. Y. Wang, "KBGAN: Adversarial learning for knowledge graph embeddings," 2017, *arXiv:1711.04071*. [Online]. Available: http://arxiv.org/abs/1711.04071

[29] P. Wang, S. Li, and R. Pan, "Incorporating GAN for negative sampling in knowledge representation learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] 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, Aug. 2008.

[32] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of Internet portals with machine learning," *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[33] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 990–998.

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

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

[36] Y. Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 7307, p. 761, Jun. 2010.

[37] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, "Sequential algorithm for fast clique percolation," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 78, no. 2, Aug. 2008, Art. no. 026109.

[38] T. P. Peixoto, "Model selection and hypothesis testing for large-scale network models with overlapping groups," *Phys. Rev. X*, vol. 5, no. 1, Mar. 2015, Art. no. 011033.

[39] J. Yang and J. Leskovec, "Community-affiliation graph model for overlapping network community detection," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 1170–1175.

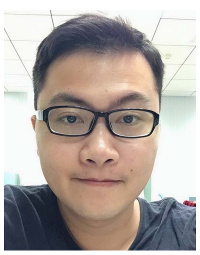
[40] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017.

[41] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 377–386.

[42] N. Liu, Q. Tan, Y. Li, H. Yang, J. Zhou, and X. Hu, "Is a single vector enough? Exploring node polysemy for network embedding," 2019, *arXiv:1905.10668*. [Online]. Available: http://arxiv.org/abs/1905.10668

[43] H. Zhang, I. King, and M. Lyu, "Incorporating implicit link preference into overlapping community detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 29, no. 1, 2015.

[44] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, Feb. 2010.

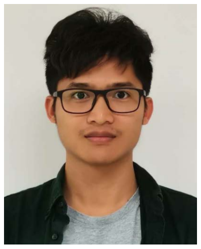


Junyang Chen received the Ph.D. degree in computer and information science from the University of Macau, Taipa, Macau, in 2020.

He is currently an Assistant Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include graph neural networks, text mining, and recommender systems.

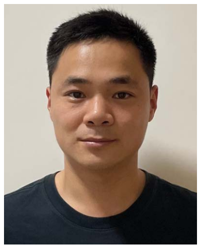
Zhiguo Gong (Senior Member, IEEE) received the Ph.D. degree in computer science from the Institute of Mathematics, Chinese Academy of Sciences, Beijing, China, in 1998.

He is currently a Professor with the Faculty of Science and Technology, University of Macau, Taipa, Macau. His current research interests include machine learning, data mining, database, and information retrieval.



Jiqian Mo received the B.S. degree in software engineering and the M.S. degree in computer science from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Internet of Things for Smart City and the Department of Computer and Information Science, University of Macau, Taipa, Macau.

His research interests include deep learning, traffic prediction, and video prediction.



Wei Wang received the Ph.D. degree in software engineering from the Dalian University of Technology, Dalian, China, in 2018.

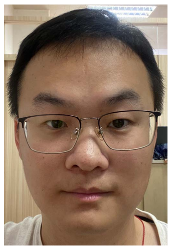
He is currently an Associate Professor with the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou, China. His research interests include computational social science, data mining, the Internet of Things, and artificial intelligence.

Wei Wang received the B.S. degree from the School of Science, Anhui Agricultural University, Hefei, China, in 2015, and the M.S. degree from the School of Computer Science and Technology, Anhui University, Hefei, in 2018. He is currently pursuing the Ph.D. degree with the School of Software Technology, Dalian University of Technology, Dalian, China.

His major research interests include transfer learning and zero-shot learning.

Cong Wang received the bachelor's degree in mathematics and applied mathematics from Inner Mongolia University, Hohhot, China, in 2017, and the master's degree in computational mathematics from the Dalian University of Technology, Dalian, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

His research interests include computer vision and deep learning.



Xiao Dong received the bachelor's degree in computer science and technology from the Fujian University of Technology, Fuzhou, China, in 2014, and the M.Sc. degree from Shandong Normal University, Jinan, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Artificial Intelligence, Sun Yat-sen University, Guangzhou, China.

His research interests include computer vision, machine learning, and signal processing.

Weiwen Liu received the B.S. degree in computer science from the South China University and Technology, Guangzhou, China, in 2016, and the Ph.D. degree in computer science from The Chinese University of Hong Kong, Hong Kong, in 2020.

Kaishun Wu received the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2011.

He is currently a Distinguished Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China.