A Comprehensive Survey on Community Detection With Deep Learning

Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, *Senior Member, IEEE*, Jian Yang, *Member, IEEE*, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, Quan Z. Sheng, *Member, IEEE*, and Philip S. Yu, *Fellow, IEEE*

***Abstract*— Detecting a community in a network is a matter of discerning the distinct features and connections of a group of members that are different from those in other communities. The ability to do this is of great significance in network analysis. However, beyond the classic spectral clustering and statistical inference methods, there have been significant developments with deep learning techniques for community detection in recent years—particularly when it comes to handling high-dimensional network data. Hence, a comprehensive review of the latest progress in community detection through deep learning is timely. To frame the survey, we have devised a new taxonomy covering different state-of-the-art methods, including deep learning models based on deep neural networks (DNNs), deep nonnegative matrix factorization, and deep sparse filtering. The main category, i.e., DNNs, is further divided into convolutional networks, graph attention networks, generative adversarial networks, and autoencoders. The popular benchmark datasets, evaluation metrics, and open-source implementations to address experimentation settings are also summarized. This is followed by a discussion on the practical applications of community detection in various domains. The survey concludes with suggestions of challenging topics that would make for fruitful future research directions in this fast-growing deep learning field.**

***Index Terms*— Community detection, deep learning, graph neural network, network representation, social networks.**

Manuscript received May 24, 2021; revised October 8, 2021; accepted November 23, 2021. This work was supported by the Australian Research Council through the DECRA Project under Grant DE200100964. *(Corresponding author: Jia Wu.)*

Xing Su, Fanzhen Liu, Jia Wu, Jian Yang, and Quan Z. Sheng are with the School of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: xing.su2@students.mq.edu.au; fanzhen.liu@students.mq.edu.au; jia.wu@mq.edu.au; jian.yang@mq.edu.au; michael.sheng@mq.edu.au).

Shan Xue is with the School of Computing, Macquarie University, Sydney, NSW 2109, Australia, and also with the CSIRO Data61, Sydney, NSW 2015, Australia (e-mail: emma.xue@mq.edu.au; emma.xue@data61.csiro.au).

Chuan Zhou is with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100093, China (e-mail: zhouchuan@amss.ac.cn).

Wenbin Hu is with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: hwb@whu.edu.cn).

Cecile Paris and Surya Nepal are with the CSIRO Data61, Sydney, NSW 2015, Australia (e-mail: cecile.paris@data61.csiro.au; surya.nepal@data61.csiro.au).

Di Jin is with the School of Computer Science and Technology, Tianjin University, Tianjin 300350, China (e-mail: jindi@tju.edu.cn).

Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: psyu@uic.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3137396.

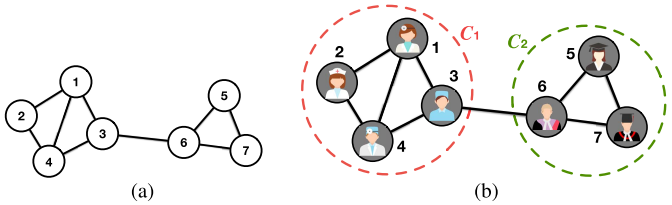
Digital Object Identifier 10.1109/TNNLS.2021.3137396

I. Introduction

THE concept of communities has been a topic of study as far back as the 1920s in sociology and social anthropology [1]. However, it is only in the 21st century that advanced scientific tools have been developed to discover communities based on real-world data [2]. Girvan and Newman [3] opened up a new direction in community detection through graph partitioning. Over the past ten years, researchers from computer science have extensively studied community detection [4] by using network topological structures [5]–[8] and semantic information [9]–[11] for both static and dynamic networks [12]–[14] and small and large networks [15]–[18]. In addition, more and more graph-based approaches have emerged to detect communities in environments with complex data structures [19], [20].

Community detection is a research area with increasing practical significance. It can be used to trace network dynamics and community impacts in detail, such as the spreading of rumors, virus outbreaks, and the evolution of tumors. As the saying goes, *birds of a feather flock together* [21]. Based on the theory of *six degrees of separation*, any person in the world can know anyone else through six acquaintances [22]. Indeed, our world is a vast network formed by a series of communities. For example, in social networks (Fig. 1), platform sponsors promote products to targeted users in detected communities [23], [24]. Community detection in citation networks [25], [26] determines the importance and interconnectedness of research topics and identifies research trends. In metabolic networks [27], [28] and protein-protein interaction (PPI) networks [29], community detection reveals the complexities of metabolisms and identifies proteins with similar biological functions. Similarly, community detection in brain networks [20], [30] reflects the functional and anatomical segregation of brain regions.

Many traditional techniques, such as spectral clustering [31], [32] and statistical inference [33]–[35], have been applied to small networks and simple cases. However, real-world networks make traditional models less applicable to practical applications. This is because real-world networks are typically rich in nonlinear information, and often include complex topologies and high-dimensional features. Hence, the computation cost of traditional techniques is high. Alternatively, the powerful techniques of **deep learning** offer flexible solutions with good community detection performance, that can: 1) learn nonlinear network properties, such as the relations extracted as the edges between nodes; 2) represent lower dimensional network embeddings that preserve complicated network structures; and 3) detect communities more accurately



(a) (b)

Fig. 1. (a) Illustration of a graph where the nodes denote users in a social network and the edges represent the following relationships between users. (b) Illustration of two communities (C_1 and C_2) based on a prediction of the users' occupations. Detection relies on the closeness of the users' online activities (topology) and account profiles (attributes).

from a range of information. Therefore, deep learning for community detection is a new trend that demands a timely and comprehensive survey.[1]

To the best of our knowledge, this article is the first comprehensive survey focusing on deep learning contributions to community detection. In discovering the patterns and functions inherent to communities [36], [37], existing surveys have mainly focused on certain aspects of community detection, including *specific techniques* [13], [14], [38]–[41], *different network types* [5], [12], [42], *community types* [6], [7], and *application scenarios* [10], [43]. The surveys on *specific techniques* are summarized but not limited to community detection based on probabilistic graphical models [38], [41], label propagation algorithms (LPAs) [39], [40], and evolutionary computation for single- and multi-objective optimizations [13], [14]. In terms of different *network types*, researchers have provided overviews on dynamic networks [12], directed networks [42], and multilayer networks [5]. Moreover, detection techniques have been reviewed over disjoint and overlapping [6], [7] *community types*. Regarding *application scenarios*, the focus has been on techniques on social networks [10], [43].

Observing the past, current, and future trends, this article aims to support researchers and practitioners to understand the community detection field with respect to the following aspects.

1) *A Systematic Taxonomy and Comprehensive Review:* We devised a new systematic taxonomy for this survey (see Fig. 2). In each category, we have reviewed, summarized, and compared the representative works. We have also briefly introduced community detection applications in the real world. These scenarios provide horizons for future community detection research and practices.
2) *Abundant Resources and High-Impact References:* Section XI outlines the available open resources, including benchmark datasets, evaluation metrics, and technique implementations.[2] Publications in the latest high-impact international conferences and high-quality peer-reviewed journals cover data mining, artificial intelligence, machine learning, and knowledge discovery.
3) *Future Directions:* As deep learning is a new research area, we have discussed its current limitations, critical challenges, and open opportunities for future directions.

The rest of the article is organized as follows: Section II defines essential notations, concepts, input, and output of deep learning approaches. Section III provides an overview of the development of community detection. Section IV introduces the deep learning taxonomy. Sections V–X summarize comprehensive reviews on each category in the taxonomy. Sections XI and XII organize the popular implementation resources and real-world applications. Finally, Section XIII discusses current challenges, and suggests future research directions before the conclusion in Section XIV. The Supplementary Material can be found in Appendix A (tables of core technique comparison of reviewed literature), Appendices B–D (resource descriptions of datasets, evaluation metrics, and implementation projects), and Appendix E (abbreviations).

[1]This article is an extended vision of our published survey [4] in IJCAI-20, which was the first published review of community detection approaches with deep learning.

[2]The repository of this article is available at https://github.com/FanzhenLiu/Awesome-Deep-Community-Detection.

TABLE I
NOTATIONS AND DESCRIPTIONS USED IN THIS ARTICLE

Notations	Descriptions
$\mathbb{R}$	A data space
$\mathcal{G}$	A graph
V, E, $\mathcal{C}$	A set of nodes, edges, communities
v_i, e_{ij}, C_k	The i-th node, edge of (v_i, v_j), k-th community
$N(v_i)$	A neighborhood of v_i
$\boldsymbol{A}$, a_{ij}	An adjacency matrix, value
$\boldsymbol{X}$, $\boldsymbol{x}_i$	A node attribute matrix, vector
y_i, c_k	The node label of v_i, community label of C_k
y_i^k	The binary community label of v_i in C_k
n, m, K, d	The number of nodes, edges, communities, attributes
$\mathcal{A}_{ij}$	The anchor links between graphs $(\mathcal{G}_i, \mathcal{G}_j)$
$\mathcal{V}$, $\mathcal{E}$, $\mathcal{X}$, Φ	A set of heterogeneous nodes, edges, attributes, metapaths
$E^{(r)}$	r-th type edges in multiplex network
$\boldsymbol{A}(+,-)$	The adjacency matrix of a signed network
$\hat{\boldsymbol{A}}$	A reconstructed adjacency matrix
$\tilde{\boldsymbol{X}}$	The corrupted node attribute matrix
$\boldsymbol{I}$	An identity matrix
$\boldsymbol{D}$	A degree matrix
$\boldsymbol{B}$, b_{ij}	A modularity matrix, value
$\boldsymbol{S}$, s_{ij}	A similarity matrix, value
$\boldsymbol{O}$, o_{ij}	Node pairwise constraint matrix, value
$\boldsymbol{P}$, p_{ij}	Community membership matrix, probability of (v_i, C_j)
$\boldsymbol{L}$, $\boldsymbol{M}$	A Laplacian, Markov matrix
$\boldsymbol{Z}$, z	A latent variable matrix, vector
Θ	Trainable parameters
$\boldsymbol{W}^{(l)}$	The weight matrix of the l-th layer in DNN
$\boldsymbol{H}^{(l)}$, $\boldsymbol{h}_i^{(l)}$	The l-th layer representation matrix, vector
$\sigma(\cdot)$	An activation function
$\mathcal{L}$	A loss function
Ω	A sparsity penalty
$\lvert\cdot\rvert$	The length of a set
$\lVert\cdot\rVert$	The norm operator
ϕ_g, ϕ_d	A generator, discriminator
ϕ_e, ϕ_r	An encoder, decoder

II. DEFINITIONS AND PRELIMINARIES

The preliminaries covered in this section include the primary definitions, the notations in Table I, and the general inputs and outputs of deep learning-based community detection models.

Definition 1 (Network): A basic network is defined as $\mathcal{G} = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the node set and $E = \{e_{ij}\}_{i,j=1}^{n}$ represents the edge set. $N(v_i) = \{u \in V | (v_i, u) \in E\}$ defines the neighborhood of a node v_i. $\boldsymbol{A} = [a_{ij}]$ denotes an $n \times n$ dimensional adjacency matrix, where $a_{ij} = 1$ if $e_{ij} \in E$; otherwise, $a_{ij} = 0$. If $a_{ij} \neq a_{ji}$, $\mathcal{G}$ is a *directed network* and an *undirected network* otherwise. If a_{ij} is weighted by $w_{ij} \in \boldsymbol{W}$, $\mathcal{G} = (V, E, \boldsymbol{W})$ is a *weighted network* and an *unweighted network* otherwise. If a_{ij}'s value differs in $+1$ (positive) and -1 (negative), $\mathcal{G}$ is a *signed network*. If node $v_i \in V$ is attributed by $\boldsymbol{x}_i \in \boldsymbol{X} \subseteq \mathbb{R}^{n \times d}$, $\mathcal{G} = (V, E, \boldsymbol{X})$ is an *attributed network*; otherwise, it is an *unattributed network*.

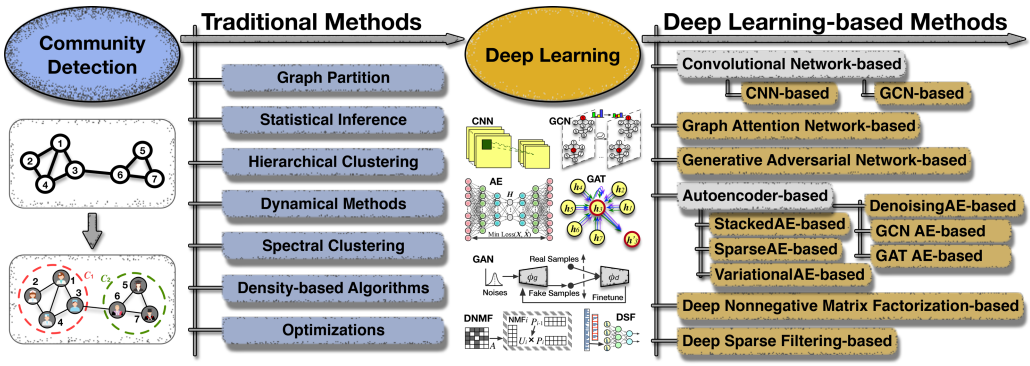


Fig. 2. Traditional community detection methods and the taxonomy of deep learning-based methods.

Definition 2 (Community): Given a set of communities $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$, each community C_k is a partition of $\mathcal{G}$ that has a regional structure and some cluster properties. A node v_i clustered into community C_k should satisfy the condition that the internal node degree inside the community exceeds its external degree. Suppose $C_k \cap C_{k'} = \emptyset, (\forall k, k',\ k \neq k')$, $\mathcal{C}$ denotes *disjoint communities*; otherwise, *overlapping communities*.

A. Community Detection Input

Deep learning models take as inputs the network topology and network attributes. The topology formed by nodes and edges can be represented in matrices, such as adjacency matrix $\boldsymbol{A}$, signed adjacency matrix $\boldsymbol{A}(+,-)$, or measurement matrices, like modularity matrix $\boldsymbol{B}$. Network attributes denote additional information about the network entities, such as node attributes $\boldsymbol{X}$.

B. Community Detection Output

Community detection methods aim to output a set of communities that can be either disjoint or overlapping. Tables I-V in Appendix A of the Supplementary Material indicate the different outputs resulting from different community detection methods. As an example, disjoint communities might represent school classes that only allow a student to belong to one class, whereas an example of overlapping communities might be a group of users who participate in several circles of a social network. The methods for detecting overlapping communities can also detect disjoint communities.

III. Development of Community Detection

Community detection has been a significant part of network analysis and data mining for some time. Fig. 3 shows its evolution from traditional computing techniques to deep learning over time. Both categories are summarized in Fig. 2. The traditional methods mainly draw on network structures to explore communities. The seven categories of methods in this section are briefly reviewed. Deep learning methods uncover deep network information and model complex relationships from high-dimensional data to lower dimensional vectors. These are reviewed in detail in Sections V–X.

A. Graph Partition

These methods, well known as graph clustering [36], are employed in deep learning models. They partition a network into communities of a given number K. Kernighan-Lin [44] is a representative heuristic algorithm. It initially divides a network into two arbitrary subgraphs and optimizes on nodes. Spectral bisection [45] is another representative method applying spectrum Laplacian matrix.

B. Statistical Inference

Stochastic block model (SBM) [33] is a widely applied generative model by assigning nodes into communities and controlling their probabilities of likelihood. The classical variants include degree-corrected SBM (DCSBM) [34] and mixed membership SBM (MMB) [35].

C. Hierarchical Clustering

This group of methods discovers hierarchical community structures (i.e., dendrogram) in three ways: divisive, agglomerative, and hybrid. The Girvan–Newman (GN) algorithm finds community structure in a divisive way by successively removing edges such that a new community occurs [3]. The fast-Newman (FN) algorithm [46], an agglomerative algorithm, gradually merges nodes, each of which is initially regarded as a community. Community detection algorithm based on structural similarity (CDASS) [47] jointly applies divisive and agglomerative strategies in a hybrid way.

D. Dynamical Methods

Random walks are used to detect communities dynamically. For example, the random walk in WalkTrap [48] calculates node distances and the probability of community membership. Information mapping (InfoMap) [49] applies the minimal-length encoding. The LPA [50] identifies diffusion communities through an information propagation mechanism.

E. Spectral Clustering

The network spectra reflect the community structures. Spectral clustering with perturbations (SCP) [31] partitions the network on the normalized Laplacian matrix and the regularized adjacency matrix and fits SBM in the pseudolikelihood algorithm. On the spectra of normalized Laplacian matrices, de Lange *et al.* [32] integrated communities in macroscopic and microscopic neural brain networks to obtain clusters.

F. Density-Based Algorithms

Significant clustering algorithms include density-based spatial clustering of applications with noise (DBSCAN) [51], structural clustering algorithm for networks (SCAN) [52], and locating structural centers for community detection (LCCD) [53]. They identify communities, hubs, and outliers by measuring the density of entities.

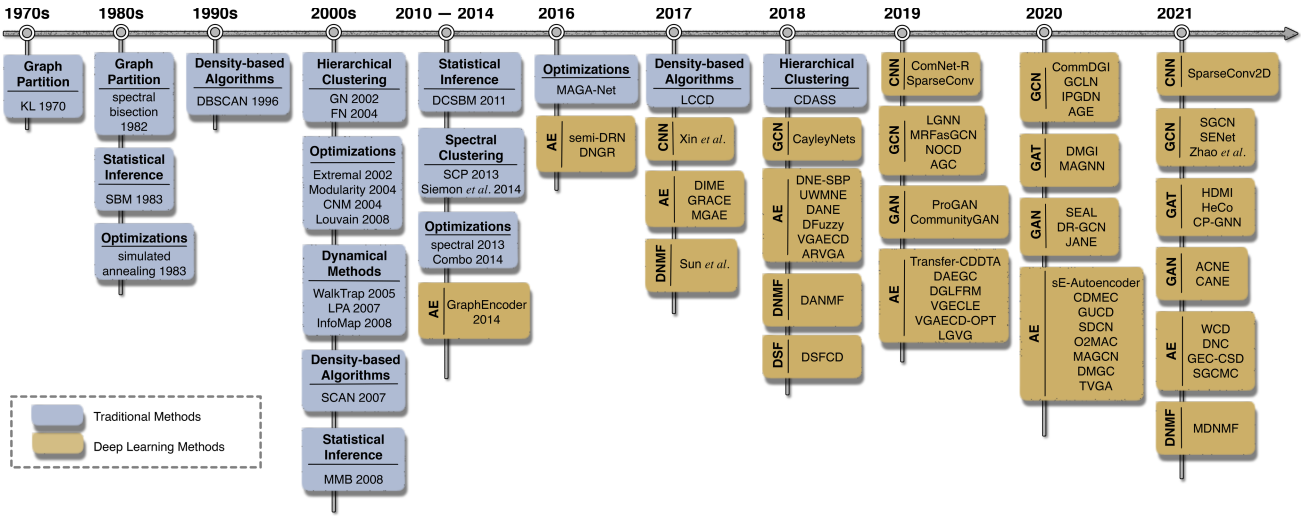


Fig. 3. Timeline of community detection development.

G. Optimizations

Community detection generally maximizes likelihood. Modularity (Q) [2], [54] is the most classic optimization function following its variant FN [46] and Clauset–Newman–Moore (CNM) [17] algorithms. Louvain [55] is another well-known optimization algorithm that employs the node-moving strategy on optimized modularity. Moreover, the extensions of greedy optimizations include simulated annealing [56], extremal optimization [57], and spectral optimization [58]. Effective in local and global searching [59], the evolutionary optimizations consist of single and multiple objectives. For example, multi-agent genetic algorithm (MAGA-Net) [60] applies a single-modularity function. Combo [61] optimizes normalized mutual information (NMI) [62], [63] and conductance (CON) [64]. Q, NMI, and CON estimate the network partition quality. Further details are given in Appendix C of the Supplementary Material.

H. Why Community Detection Needs Deep Learning?

Capturing information from connections in a straightforward way may lead to suboptimal community detection results. Deep learning models [65] bring the following additional advantages to community detection, and thus, deep learning-based community detection is a newly emerging branch of development. As a general premise, community detection through deep learning works by learning lower dimensional vectors from the high-dimensional data of complex structural relationships [11], [66]–[68]. It, therefore, enables knowledge discoveries via state-of-the-art machine learning and data mining techniques. Frameworks with representations can further embed nonstructural features, such as node attributes, to increase the knowledge of the community memberships [69]–[71]. Furthermore, various information from nodes [66], edges [72], neighborhoods [73], or multigraphs [74] can be jointly recognized with special attention in the deep learning process. In general, this has led to much more accurate community detection results. In addition, community structures in complicated real-world scenarios, such as large scale [18], high-sparse [75], complex structural [76], [77], and dynamic [78] networks, can be better explored benefiting from the capacity of deep learning to handle big data. It is worth noting that most of these advancements have happened over a relatively short period of time (see Fig. 3). And each achievement has met and overcome a series of challenges, with more still to be solved. In this article, we review these accomplishments while pointing out new opportunities for further development.

IV. Taxonomy of Community Detection With Deep Learning

To structure this survey, we devised a taxonomy for deep community detection methods according to the iconic characteristics of the employed deep learning models. The taxonomy summarizes six categories: convolutional networks, graph attention network (GAT), generative adversarial network (GAN), autoencoder (AE), deep nonnegative matrix factorization (DNMF), and deep sparse filtering (DSF). In this taxonomy, convolutional networks include convolutional neural network (CNN) and graph convolutional network (GCN). Both contribute convolutions to representing latent features for community detection. GATs are significant in that they pay special attention to community signals. The GAN model has been successfully purposed to community detection by applying an adversarial training process between the input graph and the fake samples. In the universal AE framework, the subcategories comprise stacked AEs, sparse AEs, denoising AEs, graph convolutional AEs, graph attention AEs, and variational AEs (VAEs). The taxonomy structure is shown in Fig. 2. Sections V–X review each category of methods, respectively.

V. Convolutional Network-Based Community Detection

Convolutional network models for community detection include CNNs and GCNs. CNNs [79] are a particular class of feedforward deep neural network (DNN) proposed for grid-like topological data, such as image data, where the convolution layers reduce computational costs and pooling operators ensure the CNN's robustness to feature representations. GCNs [80] were designed for graph-structured data [81], [82]. They are based on CNNs and perform first-order approximations of spectral filters. The layer-wise propagation rule for a GCN is

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right) \quad (1)$$

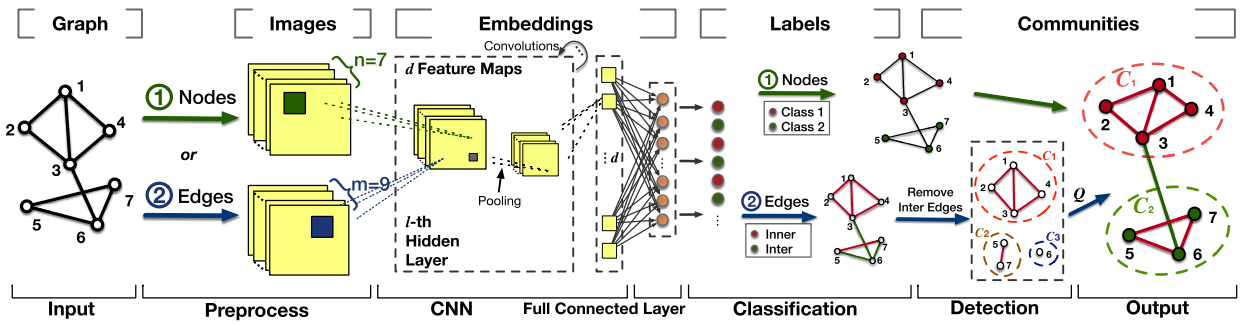


Fig. 4. General framework for CNN-based community detection (see Section V-A for more detail). Due to the strictness of CNN inputs, graph-structured data representing information of nodes and edges needs to be preprocessed into image data. The d-dimensional latent features are convolutionally mapped within multiple CNN hidden layers and a final fully connected layer outputs representations of each node or each edge for classifications. Focusing on the nodes, Flow ① predicts the community labels across k classes where nodes with the same labels are clustered into a community. Focusing on the edges, Flow ② predicts edge labels across two classes, i.e., inner and inter. Preliminary communities are formed by removing the inter-community edges and merging them into final communities under the guidance of a measurement, such as modularity Q.

where the latent representations of the lth layer are preserved in the matrix $\boldsymbol{H}^{(l)}$ ($\boldsymbol{H}^{(0)} = \boldsymbol{X}$) through an activation function $\sigma(\cdot)$ with a layer-specific trainable weight matrix $\boldsymbol{W}^{(l)}$. Here, $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$, where $\boldsymbol{I}_n$ denotes the identity matrix, and $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{a}_{ij}$, where $\tilde{a}_{ij} \in \tilde{\boldsymbol{A}}$.

A. CNN-Based Community Detection

The existing CNN-based community detection methods implement CNN models with strict data input limitations. For this reason, the graph data need to be preprocessed into labeled, image-formatted data (see Fig. 4). The techniques detailed below each solve a particular problem in community detection, as summarized in Table I in Appendix A of the Supplementary Material.

Traditional community detection is an unsupervised learning task for deep learning models, but sometimes incomplete topological structures affect the neighborhood analysis and reduce the accuracy of community detection. Networks in the real world have limited structural information. To this end, Xin *et al.* [8] proposed the first community detection model based on a supervised CNN for topologically incomplete networks (TINs). The model contains two CNN layers with max-pooling operators for network representation and a fully connected DNN layer for community detection. The CNN architecture gradually recovers intact latent features from rudimentary inputs. The convolutional layers represent each node's local features from different angles via convolutional kernels. The last full connection layer f updates the communities for each node v_i by

$$o_i^k = \sigma\left(b_k^f + \boldsymbol{W}_k^f \boldsymbol{h}_i^{(2)}\right) \tag{2}$$

where σ indicates the *sigmoid* function, $\boldsymbol{W}_k^f$ and b_k^f are the weights and bias of the kth neuron o_i^k, and $\boldsymbol{h}_i^{(2)}$ is the node representation vector output by the previous two convolutional layers. The model performs backpropagation to minimize

$$\mathcal{L} = \frac{1}{2}\sum_i \|\boldsymbol{o}_i - \boldsymbol{y}_i\|_2^2 = \frac{1}{2}\sum_i \sum_k \left(o_i^k - y_i^k\right)^2 \tag{3}$$

where $\boldsymbol{y}_i$ denotes a ground truth label vector of node v_i. Here, $y_i^k \in \{0, 1\}$ represents whether v_i belongs to the kth community or not. The experiments on this model achieve a community detection accuracy of around 80% in TINs with 10% labeled nodes and the rest unlabeled.

To deal with the high sparsity in large-scale networks, the following two methods work on particular sparse matrices (i.e., nonzero elements of adjacency matrix) for efficient community detection. Sperlí [75] designed a sparse matrix convolution (SparseConv) into CNN. De Santo *et al.* [83] further proposed a sparse CNN approach with a SparseConv2D operator so that there are significantly fewer operations in the approach.

An edge-2-image model, named community network local modularity R (ComNet-R) [84], is designed for community detection, classifying edges within and across communities through CNN. ComNet-R removes inter-community edges to prepare the disconnected preliminary communities, followed by an optimization process designed to merge these communities based on the local modularity.

B. GCN-Based Community Detection

GCNs aggregate node neighborhood information in deep graph convolutional layers to globally capture complex features for community detection (Fig. 5). There are two classes of community detection methods based on GCNs: 1) supervised/semi-supervised community classification and 2) community clustering with unsupervised network representation. Community classification methods are limited by a lack of labels in the real world. In comparison, network representations are more flexible to cluster communities through techniques, such as matrix reconstructions and objective optimizations. Table II in Appendix A of the Supplementary Material compares the techniques.

GCNs can employ traditional community detection methods as deep graph operators. For example, line graph neural network (LGNN) [85] is a supervised community detection model that improves SBMs with better community detection performance and reduced computational costs. Integrating a non-backtracking operator with belief propagation's message-passing rules, LGNN learns node represented features in directed networks. A softmax function identifies the conditional probability that a node v_i belongs to the community C_k ($o_{i,k} = p(y_i = c_k|\Theta, \mathcal{G})$) and minimizes the cross-entropy loss over all possible permutations $S_\mathcal{C}$ of the community labels, which is formulated as

$$\mathcal{L}(\Theta) = \min_{\pi \in S_\mathcal{C}} -\sum_i \log o_{i,\pi(y_i)}. \tag{4}$$

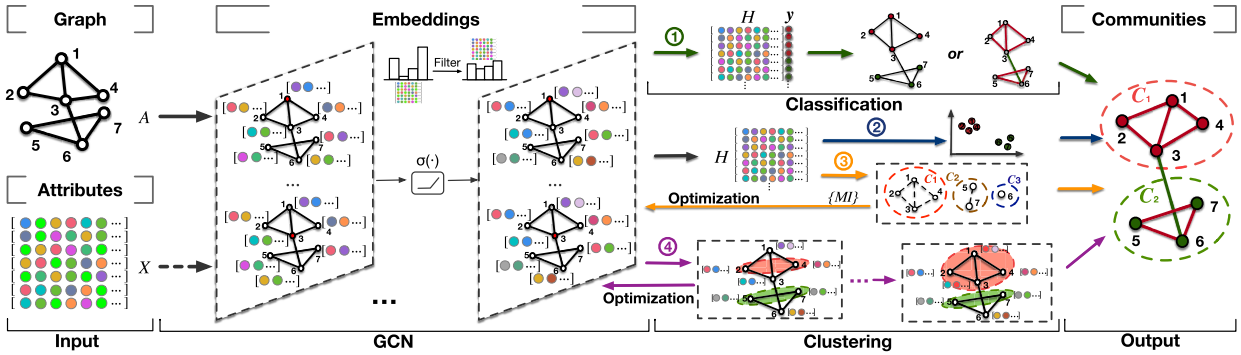


Fig. 5. General framework for GCN-based community detection (see Section V-B for more details). The inputs is a graph structure $\boldsymbol{A}$ with optional node attributes $\boldsymbol{X}$. Within multiple GCN layers, the graph's latent features are smoothed based on the requirements of the community detection scenarios. The graph representation learning is activated by $\sigma(\cdot)$. Four community detection frameworks are illustrated in ①–④ that apply either final node representations [① and ②] or temporal representations in hidden layers [③ and ④]. Given node labels, communities are detected based on the node classification in ①, while ② implements node clustering on the embeddings $\boldsymbol{H}$. This can be further optimized in ③ with measurements, e.g., Mutual Information (MI), for the best community affiliations. ④ jointly optimizes the clustering results and GCN representations, gradually detecting each node in each community with convolutionally represented node embeddings.

Since GCNs were not initially designed for community detection tasks, community structures are not the focus when learning node embeddings. To overcome this gap, a semisupervised GCN community detection model named MRFasGCN [11] characterizes hidden communities by extending a network-specific Markov random field (eMRF) as a new convolutional layer. This makes MRFasGCN community-oriented and performs a smooth refinement to the coarse results from GCN. To enable unsupervised community detection, SGCN [86] designs a local label sampling model to locate the structural centers for community detection. By integrating the label sampling model with GCN, SGCN encodes both network topology and node attributes in the training of each node's community membership without any prior label information.

In terms of a probabilistic inference framework, detecting overlapping communities can be solved by a generative model that infers the community affiliations of nodes. For example, neural overlapping community detection (NOCD) [87] combines the Bernoulli–Poisson (BP) probabilistic model and a two-layer GCN to learn community affiliation vectors by minimizing BP's negative log-likelihood. The final communities are determined by setting a threshold to keep recognizing and removing weak affiliations.

Spectral GCNs represent all latent features from the node's neighborhood. The features of neighboring nodes converge to the same values by repeatedly operating Laplacian smoothing in deep GCN layers. However, these models can lead to an over-smoothing problem in community detection. To reduce the negative impact, a new GCN architecture—graph convolutional ladder-shape networks (GCLN) [88]—was designed. It offers unsupervised community detection through a k-means clustering process, which is based on a U-Net in the CNN field. A contracting path and an expanding path are built symmetrically in the GCLN. The contextual features captured from the contracting path are fused with the localized information learned in the expanding path.

Since different types of connections are generally treated as plain edges, GCNs represent each type of connection and aggregate features along them, leading to redundant representations. Independence promoted graph disentangled network (IPGDN) [89] distinguishes the neighborhood into different parts and automatically discovers the nuances of a graph's independent latent features by neighborhood routing. In this way, the model learns a better disentangled representation for community detection. The independence among latent embeddings is enforced by Hilbert–Schmidt independence criterion (HSIC) regularization.

For attributed graphs, community detection via a GCN relies on both structural information and attributed features [90], where neighboring nodes and the nodes with similar features are likely to cluster in the same community. Therefore, graph convolutions multiply the above-mentioned two graph signals and need to filter out high-frequency noises smoothly. To this end, a low-pass filter is embedded in adaptive graph convolution (AGC) [69] with spectral clustering through

$$p(\lambda_q) = \left(1 - \frac{1}{2}\lambda_q\right)^k \tag{5}$$

where the frequency response function of $\mathcal{G}$ denoted as $p(\Lambda) = \text{diag}(p(\lambda_1), \ldots, p(\lambda_n))$ is decreasing and nonnegative on all eigenvalues. AGC convolutionally selects suitable neighborhood hop sizes in k and represents graph features by a k-order graph convolution as

$$\bar{\boldsymbol{X}} = \left(\boldsymbol{I} - \frac{1}{2}\boldsymbol{L}_s\right)^k \boldsymbol{X} \tag{6}$$

where $\boldsymbol{L}_s$ denotes the symmetrically normalized graph Laplacian on λ_q. The filter smooths the node embedding $\bar{\boldsymbol{X}}$ adjusted to k. Here, higher values lead to better filtering performance.

Adaptive graph encoder (AGE) [91] is another smoothing filter model scalable to community detection. AGE adaptively performs node similarity measurement ($\boldsymbol{S} = [s_{ij}]$) and t-stacked Laplacian smoothing filters ($\bar{\boldsymbol{X}} = (\boldsymbol{I} - \gamma\,\boldsymbol{L})^t \boldsymbol{X}$) by

$$\mathcal{L} = \sum_{(v_i, v_j) \in V'} -s'_{ij} \log(s_{ij}) - \left(1 - s'_{ij}\right) \log(1 - s_{ij}) \tag{7}$$

where V' denotes balanced training set over positive (similar) and negative (dissimilar) samples. s'_{ij} is the ranked binary similarity label on node pairs (v_i, v_j).

As another example of how GCN filters contribute to community detection, graph convolutional neural networks with Cayley polynomials (CayleyNets) [92], in the architecture of spectral graph convolution, specializes in narrowband filtering. This is because low frequencies contain extensive community information for community detection-aimed representations.

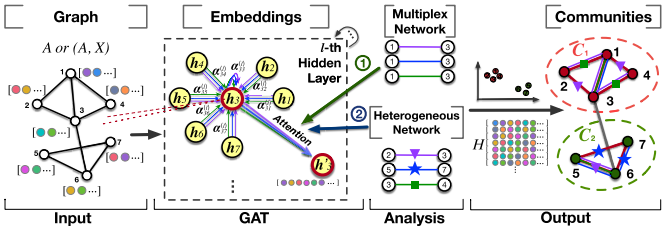


Fig. 6. General framework for GAT-based community detection (see Section VI for more details). The GAT assigns attention coefficients $\boldsymbol{\alpha}_{ij}^{(l)}$: {green, blue, purple} to each neighborhood $N(v_i)$ in the lth hidden layer. The represented vector $\boldsymbol{h}_i'$ aggregates the available information from ① differently colored edges between the same node pairs in multiplex networks, or ② meta paths or context paths in a heterogeneous network. Nodes are clustered into communities by analyzing the GAT embeddings $\boldsymbol{H}$.

Cooperating with the Cayley filter, CayleyNets involves a mean pooling in the spectral convolutional layers and a semi-supervised softmax classifier on nodes for community membership prediction.

To capture the global cluster structure for community detection, the spectral embedding network for attributed graph clustering (SENet) [93] introduces the loss of spectral clustering into a three-layer GCN's output layer by minimizing

$$\mathcal{L} = -\operatorname{tr}\left((\boldsymbol{H}^{(3)})^{\top}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{K}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(3)}\right) \tag{8}$$

where $\boldsymbol{K}$ is a kernel matrix encoding the typologies and attributes, and tr$(\cdot)$ represents the trace of a matrix.

Community deep graph infomax (CommDGI) [94] jointly optimizes graph representations and clustering through MI on nodes and communities and measures graph modularity for maximization. It applies contrastive training to obtain better representations and k-means for node clustering and targeting the cluster centers. Zhao *et al.* [95] proposed a graph debiased contrastive learning that simultaneously performs representations and clustering such that the clustering results and discriminative representations are both improved.

VI. Graph Attention Network-Based Community Detection

GATs [96] aggregate node features in neighborhoods via trainable weights with attention

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\sum_{j\in N(v_i)} \alpha_{ij}^{(l+1)}\boldsymbol{W}^{(l+1)}\boldsymbol{h}_j^{(l)}\right) \tag{9}$$

where $\boldsymbol{h}_i^{(l)}$ represents the node v_i's representation in the lth layer ($\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$) and $\alpha_{ij}^{(l+1)}$ is the attention coefficient between v_i and $v_j \in N(v_i)$. In community detection (Fig. 6), the attention mechanism contributes to adaptively learning the importance of each node in a neighborhood. Correlations between similar nodes are computed close to the ground truth of community membership, which is inherited in the representations learned by the GAT. These attentions filter spatial relations or integrate meta paths, which means the communities in attributed, multiplex, and heterogeneous networks can be easily detected. Techniques are compared in Table III in Appendix A of the Supplementary Material.

The relationships in deep community detection models need special attention. For example, citations and co-subject relationships are both pertinent when clustering papers into research topics. Hence, a multiplex network is constructed from the multiple different types of relationships reflected in the edges (i.e., $E^{(r)}$). Each type of edge is grouped into a layer r. Thus, the GAT is able to pay attention to the relationship type. As an example, deep graph infomax for attributed multiplex network embedding (DMGI) [66] independently embeds each relationship type and computes the network embeddings to maximize globally shared features when detecting communities. Contrastive learning is then conducted between the original network and a corrupted network on each layer through the discriminator. Consensus regularization is subsequently applied with the attention coefficient to integrate the final embeddings. An attention mechanism [97] can preprocess the less significant relations by weakening their coefficient, especially when various types of relationships are present.

Despite taking the mutual dependence between node embeddings and a global summary of the entire graph into account, DMGI ignores the dependence between node embeddings and node attributes. To fill this gap, the high-order deep multiplex infomax (HDMI) [70] is developed to capture both dependencies and their interactions, in order to obtain high-order mutual information. Furthermore, a fusion module based on layer-dependent semantic attention [98] combines the different node embeddings from multiplex network layers for the purpose of community detection in multiplex networks.

Heterogeneous information networks (HINs) involve diverse types of nodes $\mathcal{V}$ and edges $\mathcal{E}$. Deep community detection over HINs uses heterogeneous typologies and attributes $\mathcal{X}$. Since meta paths reflect complex semantic relations in HINs, meta path aggregated graph neural network (MAGNN) [77] offers a superior community detection solution through multi-informative meta paths Φ, where heterogeneous structures and semantics are distinguished in the graph attention layers. Specifically, an attention mechanism [99] is applied to aggregate intra- and inter-meta paths. This reduces high heterogeneity. Thus, MAGNN embeds richer topological and semantic information to enhance the detection results. In another example, HeCo [100] exploits an attention mechanism [97] to embed the network schema and meta paths for preserving local and high-order structures. A modified contrastive learning module then guides the two parts of the embeddings to learn high-level embeddings and further benefits community detection.

These approaches exploit meta paths, and yet defining meaningful meta paths requires a good deal of domain knowledge. As a possible solution, a context path-based graph neural network (CP-GNN) [101] was built to learn node embeddings with context paths. The framework leverages attention mechanisms to determine the importance of different relationships. The non-predefined context paths are very helpful for capturing high-order relationships.

VII. Generative Adversarial Network-Based Community Detection

Adversarial training is effective in generative models and improves discriminative ability. However, one problem that needs to be solved when adversarial learning is applied to community detection is overfitting (see Fig. 7 and Table IV of the Supplementary Material). GANs [102] competitively train a generator ϕ_g and a discriminator ϕ_d within an adversarial framework. Here, $\phi_d(\boldsymbol{x})$ represents the probability of input data, while $\phi_g(\boldsymbol{z})$ learns the generator's distribution p_g via input noise variables $p_z(\boldsymbol{z})$. The generator fools the discriminator by generating fake samples. Its objective function is defined as

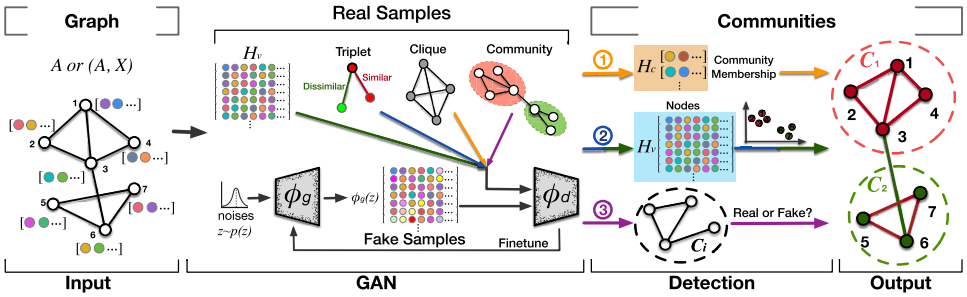


Fig. 7. General framework for GAN-based community detection with details in Section VII. GAN produces fake samples $\phi_g(z)$ by the generator ϕ_g to fool the discriminator ϕ_d. The GAN's real samples can be node embeddings $\boldsymbol{H}_v$, local topology (e.g., triplet and clique) or communities. Thus, real and fake samples are used to competitively finetune a community's features. Flow ① determines community membership via a clique-level GAN. Flow ② detects communities based on competitive node-level representations from $\boldsymbol{H}_v$ or triplets, while Flow ③ directly discriminates communities through the discriminator.

$$\min_{\phi_g}\max_{\phi_d}\ \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log\phi_d(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z(z)}[\log(1-\phi_d(\phi_g(z)))]. \quad (10)$$

Seed expansion with generative adversarial learning (SEAL) [103] generates seed-aware communities from selected seed nodes via a graph pointer network with incremental updates (iGPN). SEAL consists of four components at the community level, i.e., a generator, a discriminator, a seed selector, and a locator. The discriminator adopts graph isomorphism networks (GINs) to modify the generated communities with the ground truth of community labels. The locator is designed to provide regularization signals to the generator so that any irrelevant nodes that are detected can be eliminated.

Designed for imbalanced communities, dual-regularized graph convolutional networks [104] integrates a conditional GAN with a dual-regularized GCN model—i.e., a latent distribution alignment regularization is complemented with a class-conditioned adversarial regularization. The first regularization $\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{gcn}} + \alpha\mathcal{L}_{\text{dist}}$ balances the communities by minimizing the Kullback–Leibler (KL) divergence between the majority and minority community classes $\mathcal{L}_{\text{dist}}$ together with standard GCN training $\mathcal{L}_{\text{gcn}}$. The second regularization is designed to distinguish communities from labeled node representations via

$$\min_{\phi_g,\mathcal{L}}\max_{\phi_d}\ \mathcal{L}(\phi_d,\phi_g) = \mathbb{E}_{v_i\sim p_{\text{data}}(v_i)}\log\phi_d(v_i|y_i) + \mathbb{E}_{z\sim p_z(z)}[\log(1-\phi_d(\phi_g(z|y_i))) + \mathcal{L}_{\text{reg}}] \quad (11)$$

where $\mathcal{L}_{\text{reg}} = \sum_{u\in N(v_i)}\|\boldsymbol{h}_{v_i'} - \boldsymbol{h}_u\|_2$ forces the generated fake node v_i' to reconstruct the respective neighborhood relations (as $u\sim v_i$) and y_i is the ground truth label of node v_i.

Instead of generating only one kind of fake samples, jointly adversarial network embedding (JANE) [67] employs two types of network information—topological data and node attributes—to capture semantic variations from adversarial groups of real and fake samples. Specifically, JANE represents embeddings $\boldsymbol{H}$ through a multi-head self-attention encoder ϕ_e, where the Gaussian noise is added to fake features for competition over the generator ϕ_g and the discriminator ϕ_d. Its objective function is defined as

$$\begin{aligned}\min_{\phi_g,\phi_e}\max_{\phi_d}\ &\mathcal{L}(\phi_d,\phi_e,\phi_g)\\ &= \mathbb{E}_{(\boldsymbol{a},\boldsymbol{x})\sim p_{AX}}[\underbrace{\mathbb{E}_{\boldsymbol{h}\sim p_{\phi_e}(\cdot|\boldsymbol{a},\boldsymbol{x})}[\log\phi_d(\boldsymbol{h},\boldsymbol{a},\boldsymbol{x})]}_{\log\phi_d(\phi_e(\boldsymbol{a},\boldsymbol{x}),\boldsymbol{a},\boldsymbol{x})}]\\ &\quad + \mathbb{E}_{\boldsymbol{h}\sim p_H}[\underbrace{\mathbb{E}_{(\boldsymbol{a},\boldsymbol{x})\sim p_{\phi_g}(\cdot|\boldsymbol{h})}[\log(1-\phi_d(\boldsymbol{h},\boldsymbol{a},\boldsymbol{x}))]}_{\log(1-\phi_d(\boldsymbol{h},\phi_g(\boldsymbol{h})))}] \end{aligned} \quad (12)$$

where p_{AX} denotes the joint distribution of the topology $\boldsymbol{A}$ ($\boldsymbol{a}\subseteq\boldsymbol{A}$) and the sampled node attributes $\boldsymbol{X}$ ($\boldsymbol{x}\subseteq\boldsymbol{X}$).

The proximity can capture underlying relationships within communities. However, sparsely connected networks in the real world generally do not provide enough edges, and attributes in networks cannot be measured by proximity. To address these limitations, proximity generative adversarial network (ProGAN) [71] encodes each node's proximity assisted by a set of generated triplets adversarially, so that community relationships are discovered and preserved in a low-dimensional space.

Community detection with generative adversarial nets (CommunityGAN) [105] was designed for overlapping communities. It derives node representations by assigning each node-community pair a nonnegative factor. Its objective function optimizes through a motif-level generator $\phi_g(\cdot|v_i;\Theta_g)$ and discriminator $\phi_d(\cdot,\Theta_d)$:

$$\begin{aligned}\min_{\Theta_g}\max_{\Theta_d}\ &\mathcal{L}(\phi_g,\phi_d)\\ &= \sum_i\big(\mathbb{E}_{C'\sim p_{\text{true}}(\cdot|v_i)}\big[\log\phi_d(C';\Theta_d)\big]\\ &\quad + \mathbb{E}_{V'\sim\phi_g(V'|v_i;\Theta_g)}\big[\log(1-\phi_d(V';\Theta_d)\big]\big)\end{aligned} \quad (13)$$

where Θ_g and Θ_d unify all nonnegative representation vectors of node v_i in the generator and the discriminator. $V'\subseteq V$ denotes a node subset, C' represents motifs (i.e., cliques here), and the conditional probability $p_{\text{true}}(C'|v_i)$ describes the preference distribution of C' covering v_i over all other motifs $C'\in\mathcal{C}'$.

To capture global structural information, community-aware network embedding (CANE) [106] integrates a community detection model, i.e., latent dirichlet allocation (LDA), with an adversarial node representation process:

$$\begin{aligned}\min_{\Theta_g}\max_{\Theta_d}\ &\mathcal{L}(\phi_g,\phi_d)\\ &= \sum_i\big(\mathbb{E}_{v\sim p_{\text{true}}(v|v_i)}\big[\log\phi_d(v,v_i;\Theta_d) + P_\phi(v|v_i)\big]\\ &\quad + \mathbb{E}_{v\sim\phi_g(v|v_i;\Theta_g)}\big[\log(1-\phi_d(v,v_i;\Theta_d) - P_\phi(v|v_i)\big]\big)\end{aligned} \quad (14)$$

where $P_\phi(v|v_i)$ indicates the community similarity between a node v_i and the node sample v, and $\phi_g(v|v_i;\Theta_g)$ outputs the node connectivity distribution. In turn, the representations characterize the community property.

Network embedding and overlapping community detection with adversarial learning (ACNE) [107] further generates negative communities as fake samples to learn distinguished

community representation. ACNE uses a walking strategy with perception to assist with the sampling of overlapping communities for nodes. It also exploits the discriminator to jointly map the node and community membership embeddings so that the correlations between nodes and communities are preserved in representations.

VIII. Autoencoder-Based Community Detection

AEs are most commonly used for unsupervised community detection. They span stacked AEs, sparse AEs, denoising AEs, convolutional AEs, and VAEs. AEs can depict nonlinear, noisy real-world networks and represent communities from rich information through reconstruction. A general framework for an AE [108] comprises an encoder $\boldsymbol{H} = \phi_e(\boldsymbol{A}, \boldsymbol{X})$ and a decoder $\hat{\boldsymbol{A}} = \phi_r(\boldsymbol{H})$ or $\hat{\boldsymbol{X}} = \phi_r(\boldsymbol{H})$. The encoder ϕ_e maps a high-dimensional network structure $\boldsymbol{A}$ and possible attributes $\boldsymbol{X}$ into a low-dimensional latent feature space $\boldsymbol{H}$. The decoder ϕ_r reconstructs a decoded network from the encoder's representations $\boldsymbol{H}$ so that $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{X}}$ inherit preferred information from $\boldsymbol{A}$ and $\boldsymbol{X}$. A loss function $\mathcal{L}(\boldsymbol{x}, \phi_r(\phi_e(\boldsymbol{x})))$ is designed to maximize the likelihood between the source data $\boldsymbol{x}$ and the decoded data $\phi_r(\phi_e(\boldsymbol{x}))$. Fig. 8 shows how AEs are generally applied in community detection, and Table V of the Supplementary Material compares the techniques.

A. Stacked AE-Based Community Detection

Compared with a single AE, a set of stacked AEs developed in deep hidden layers can better embed high-dimensional community features by connecting each layer's output to the successive layer's input. In this architecture, stacked AEs represent multilevel and dynamic information to flexibly support wide community detection implementations [109].

The semi-supervised nonlinear reconstruction algorithm with DNN (semi-DRN) [68] is a design for a stacked AE, where a modularity matrix learns the nonlinear node representations in AEs and k-means is used to cluster the final community structure. Given an edge between the nodes v_i and v_j in an adjacency matrix $\boldsymbol{A} = [a_{ij}]$, a modularity $b_{ij} = a_{ij} - (k_i k_j / 2m)$ in a modularity matrix $\boldsymbol{B}$ is optimized for maximization [2], [54]. Pairwise node similarities (community membership) are encoded based on node representations and a pairwise constraint matrix $\boldsymbol{O} = [o_{i,j} \in \{0, 1\}]$ is defined to provide prior knowledge about whether or not nodes v_i and v_j belong to the same community, i.e., $(o_{i,j} = 1)$ or not $(o_{i,j} = 0)$. Hence, semi-DRN is optimized by minimizing the loss function:

$$\mathcal{L} = \mathcal{L}(\boldsymbol{B}, \hat{\boldsymbol{B}}) + \lambda \mathcal{L}(\boldsymbol{O}, \boldsymbol{H}) \tag{15}$$

where $\hat{\boldsymbol{B}}$ represents the reconstructed $\boldsymbol{B}$ over stacked representations by a series of AEs, λ denotes an adjusting weight between the AE reconstruction loss $\mathcal{L}(\boldsymbol{B}, \hat{\boldsymbol{B}})$ and the pairwise constraints $\mathcal{L}(\boldsymbol{O}, \boldsymbol{H})$, and $\mathcal{L}(\boldsymbol{O}, \boldsymbol{H})$ measures each pair of community membership o_{ij} and latent representations $(\boldsymbol{h}_i, \boldsymbol{h}_j)$ in the stacked AEs.

Similarly, deep network embedding with structural balance preservation (DNE-SBP) [72] incorporates the adjusting weight on pairwise constraints for signed networks so that the stacked AEs cluster the closest nodes distinguished by positive and negative connections. Unified weight-free multicomponent network embedding (UWMNE) and its variant with local enhancement (WMCNE-LE) [110] preserve the community properties with the network topology and semantic information, and integrate the diverse information in a deep AE from a local network structure perspective.

To discover communities in time-varying networks [111]–[113], a framework called semi-supervised evolutionary autoencoder (sE-Autoencoder) [78] was developed within an evolutionary clustering framework. The way the approach works is that community structures in previous time steps $t - 1$ successively guide detection in the current time step t. To this end, sE-Autoencoder adds a temporal smoothness regularization $\mathcal{L}(\boldsymbol{H}_{(t)}, \boldsymbol{H}_{(t-1)})$ for minimization into the following objective function:

$$\mathcal{L} = \mathcal{L}(\boldsymbol{S}_{(t)}, \hat{\boldsymbol{S}}_{(t)}) + \lambda \mathcal{L}(\boldsymbol{O}, \boldsymbol{H}_{(t)}) + (1 - \lambda)\mathcal{L}(\boldsymbol{H}_{(t)}, \boldsymbol{H}_{(t-1)}) \tag{16}$$

where the reconstruction error $\mathcal{L}(\boldsymbol{S}_{(t)}, \hat{\boldsymbol{S}}_{(t)})$ minimizes the loss between the similarity matrix $\boldsymbol{S}_{(t)}$ and the reconstructed matrix $\hat{\boldsymbol{S}}_{(t)}$ in the time step t. The parameter λ controls the node pairwise constraint $\mathcal{L}(\boldsymbol{O}, \boldsymbol{H}_{(t)})$ and the temporal smoothness regularization from time step $t - 1$ to t.

In attributed networks, deep attributed network embedding (DANE) [114] develops a two-branch AE framework: one branch maps highly nonlinear network structures to a low-dimensional feature space, and the other collaboratively learns node attributes. As similar nodes are more likely to be clustered into the same community, during representation learning, DANE measures these similarities with a series of proximities based on network topological and attribute information. Optimizations are applied to reconstruction losses at the first-order proximity $\mathcal{L}_f$, the high-order proximity $\mathcal{L}_h$, semantic proximity $\mathcal{L}_s$, and the negative log-likelihood control $\mathcal{L}_c$ for a consistent and complementary representation.

To solve the problem of shifted distributions, imbalanced features, or a lack of data, transfer learning-inspired community detection with deep transitive autoencoder (Transfer-CDDTA) [115] uses transfer learning adapted to an unsupervised community detection task. Transfer-CDDTA balances stacked AEs via KL divergence within the source and target domains while learning node embeddings. With the aim of mapping community information into one smooth feature space, Transfer-CDDTA separates the input adjacency matrix $\boldsymbol{A}$ into a source domain s and a target domain t via the similarity matrices $\boldsymbol{S}_s$ and $\boldsymbol{S}_t$. This remains the node similarity values for each stacked AE. Transfer-CDDTA then incorporates domain-independent features into the following minimization:

$$\mathcal{L} = \mathcal{L}_s(\boldsymbol{S}_s, \hat{\boldsymbol{S}}_s) + \mathcal{L}_t(\boldsymbol{S}_t, \hat{\boldsymbol{S}}_t) + \alpha \mathrm{KL}(\boldsymbol{H}_s, \boldsymbol{H}_t) + \beta \mathcal{L}(\Theta; \gamma) \tag{17}$$

where α, β, and γ are trade-off parameters, $\mathcal{L}_s$ and $\mathcal{L}_t$ denote the reconstruction losses of source and target domains, KL smooths the KL divergence on the encoded features $(\boldsymbol{H}_s, \boldsymbol{H}_t)$ across the two domains, and $\mathcal{L}(\Theta)$ is a regularization term to reduce overfitting in the optimization.

To detect communities in an emerging heterogeneous social network that lacks connections between nodes, deep aligned autoencoder-based embedding (DIME) [116] uses a corresponding mature network with extensive heterogeneous connections to assist detection. The emerging network and the mature one are aligned by anchor links $\mathcal{A}_{ij}$. DIME stacks AEs to learn diverse representations from a set of meta paths Φ. Accordingly, diverse meta proximities are calculated for

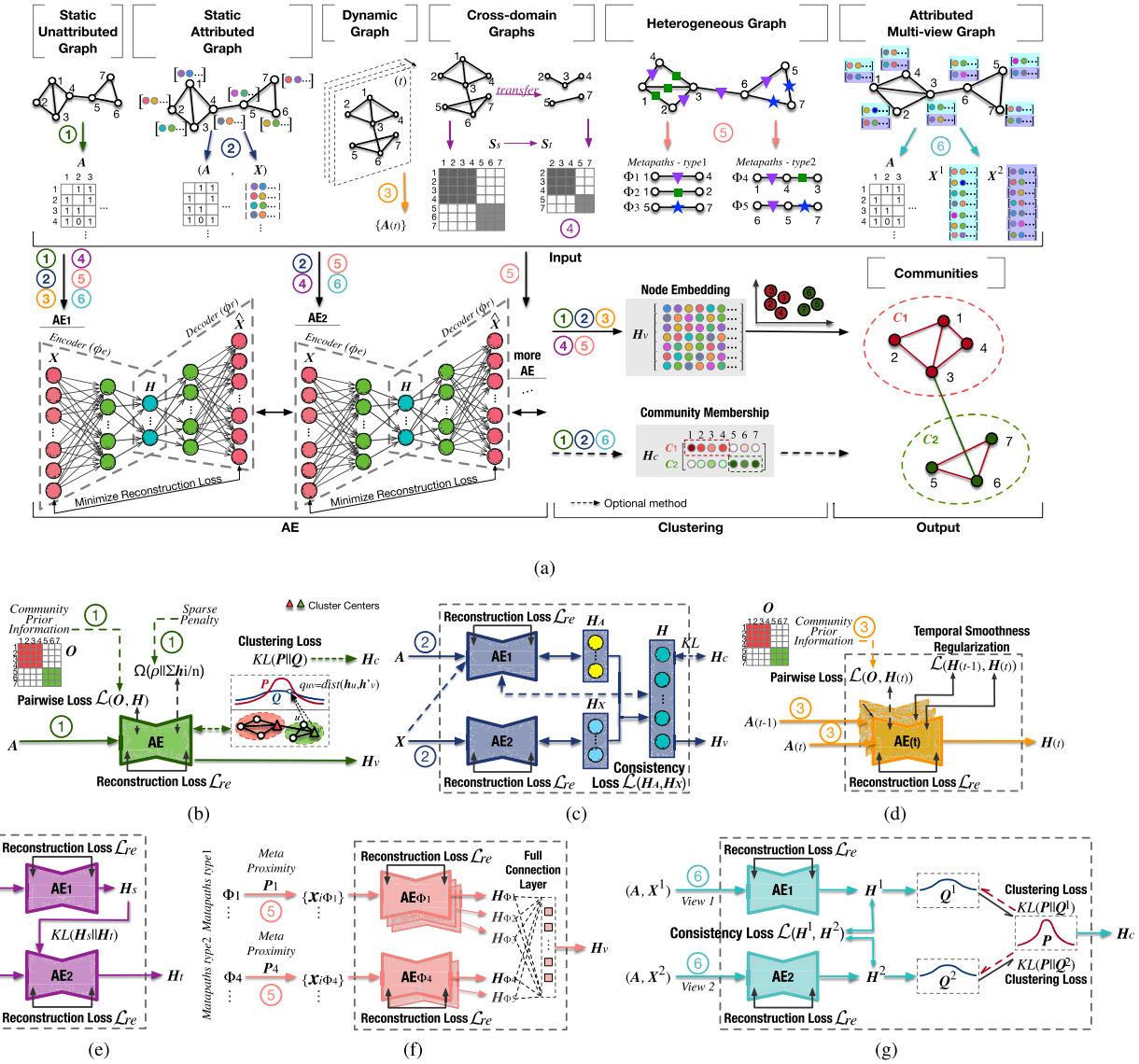


Fig. 8. General framework for AE-based community detection consists of: (a) framework of the input layer, the AE and the clustering layers, and the output layer; and (b)–(g) AE-based community detection processes for each type of graph. (a) Input is a graph with topological and attribute information. There are represented by an adjacency matrix $\boldsymbol{A}$ [①–③ and ⑥], some node attribute vectors $\boldsymbol{X}$ [② and ⑥], a node similarity matrix $\boldsymbol{S}$ [④], and meta paths Φ [⑤]. More than one AE can be included such that each AE embeds a part of the input. Two optional work flows representively output the node embeddings $\boldsymbol{H}_v$ and community membership matrix $\boldsymbol{H}_c$ which form the community detection results (i.e., the AE output layer). (b) Inputting the typologies $\boldsymbol{A}$, the AE outputs the node embeddings $\boldsymbol{H}_v$ by minimizing the reconstruction loss $\mathcal{L}_{\text{re}}$ (solid arrows), or outputs the community membership matrix $\boldsymbol{H}_c$ by further minimizing the clustering loss over $\boldsymbol{P}$ and $\boldsymbol{Q}$ distributions. This reflects a node's probability in a community (dashed arrows). Prior information about the community for $\mathcal{L}(\boldsymbol{O}, \boldsymbol{H})$ (dashed arrows) or a sparse penalty for $\Omega(\rho \| \sum \boldsymbol{h}_i/n)$ (dashed arrows) optionally guides the AE's learning over the embeddings in the hidden layer $\boldsymbol{h}_i \in \boldsymbol{H}$. (c) Typologies $\boldsymbol{A}$ and node attributes $\boldsymbol{X}$ are, respectively, represented by AE1 and AE2 (solid arrows). Their embeddings $\boldsymbol{H}_A$ and $\boldsymbol{H}_X$ are aggregated by minimizing the consistency loss. Otherwise, an AE embeds all inputs over $(\boldsymbol{A}, \boldsymbol{X})$ (dashed arrows). $\boldsymbol{H}_v$ or $\boldsymbol{H}_c$ is outputted in the same way as (b). (d) tth snapshot $\boldsymbol{A}_{(t)}$ in the dynamic graph for the node embeddings $\boldsymbol{H}_{(t)}$. A temporal smoothness regularization optimizes $\boldsymbol{H}_{(t)}$ over the previous snapshot's embeddings. $\boldsymbol{O}$ is optionally employed. (e) Two AEs representing similarity matrices from the source and target domains into $\boldsymbol{H}_s$ and $\boldsymbol{H}_t$. Here, $\boldsymbol{H}_s$ guides AE2's training and the output is the node embeddings of the target graph $\boldsymbol{H}_t$. (f) Each meta path in an AE with the input feature vectors $\{\boldsymbol{x}_{i\Phi}\}$ denoting the meta proximity $\boldsymbol{p}_i \in \boldsymbol{P}$ between node v_i in the meta paths Φ. All meta path embeddings are aggregated for $\boldsymbol{H}_v$ in the fully connected layer in these stacked AEs. (g) Attributes in each view are input with $\boldsymbol{A}$ into an AE where the node embeddings in the different views are optimized by the clustering loss.

each meta path and similar nodes gather in tight regions in a low-dimensional latent feature space. Community information implied by such tight regions is transferred from the mature aligned network to help detection in the emerging network.

B. Sparse AE-Based Community Detection

Sparsity is common in real-world networks and often creates real computational difficulties for community detection algorithms. To solve these issues, sparse AEs [117] incorporate a sparsity penalty $\Omega(\boldsymbol{h})$ into the hidden layers $\boldsymbol{h}$ of vanilla AEs. The reconstruction loss function is generally formulated as

$$\mathcal{L}(\boldsymbol{x}, \phi_r(\phi_e(\boldsymbol{x}))) + \Omega(\boldsymbol{h}). \tag{18}$$

The autoencoder-based graph clustering model (GraphEncoder) [118] was the first approach to use an AE for graph clustering. It processes sparsity with a sparsity term as a part of its loss function:

$$\mathcal{L}(\Theta) = \sum_i \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2 + \beta\Omega\left(\rho \left\| \frac{1}{n}\sum_i \boldsymbol{h}_i \right\|\right). \tag{19}$$

Here, the weight parameter β controls the sparsity penalty $\Omega(\cdot\|\cdot)$ over a configuration value ρ (a small constraint, such as 0.01) and the average of hidden layers' activation values. GraphEncoder improves clustering efficiency with large-scale networks and proves that sparse networks can provide enough structural information for representations. A weighted community detection model (WCD) [119] was further designed for weighted networks with a spare AE which considers second-order neighbors for more accurate community structures.

Also in this category is the deep learning-based fuzzy clustering model (DFuzzy) [18]. DFuzzy was designed to detect overlapping communities in sparse large-scale networks. In a parallel processing framework, DFuzzy identifies community centers and forms overlapping or disjoint communities using a stacked sparse AE by providing greedy layer-wise training toward modularity maximization. The experiments show that the detection performance of DFuzzy is obviously superior to non-deep learning baselines.

Community detection method via ensemble clustering (CDMEC) [120] combines sparse AEs with a transfer learning model to discover more valuable information from local network structures. To this end, CDMEC constructs four similarity matrices and employs transfer learning to share local information via the AEs' parameters. A consensus matrix aggregates the community detection results individually produced by the four similarity matrices and supported by k-means. Factorizing the consensus matrix helps to determine the final communities.

C. Denoising AE-Based Community Detection

The denoising process, i.e., removing noise within DNN layers, can increase a model's robustness. In this vein, denoising AEs [121] minimize the reconstruction loss between the input $\boldsymbol{x}$ and the decoded features from corrupted input $\widetilde{\boldsymbol{x}}$ by minimizing

$$\mathcal{L}(\boldsymbol{x}, \phi_r(\phi_e(\widetilde{\boldsymbol{x}}))). \tag{20}$$

Deep neural networks for graph representation (DNGR) [122] applies a stacked denoising encoder to increase robustness when capturing local structural information. Specifically, it generates a probabilistic co-occurrence matrix and a shifted positive pointwise MI matrix by randomly walking over communities. As an extension to DNGR, deep neural network-based clustering-oriented network embedding (DNC) [123] jointly learns node embeddings and cluster assignments.

To handle sparse and noisy node attributes, graph clustering with dynamic embedding (GRACE) [73] exploits denoising AEs to learn the robust node embeddings which are then propagated within neighborhoods to capture the dynamically changing inter-community activities in a self-training clustering process.

Marginalized graph autoencoder (MGAE) [124] denoises both graph attributes and structures to improve community detection through a marginalization process. It generates the corrupted attributes $\widetilde{\boldsymbol{X}}$ η times by randomly removing some attributes and trains the following objective function:

$$\mathcal{L} = \frac{1}{\eta}\sum_{i=1}^{\eta}\left\|\boldsymbol{X} - \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{X}}\boldsymbol{W}\right\|_2 + \lambda\mathcal{L}(\boldsymbol{W}) \tag{21}$$

where $\mathcal{L}(\boldsymbol{W})$ denotes a regularization term on AE's parameters $\boldsymbol{W}$ with a coeffcient λ.

D. Graph Convolutional AE-Based Community Detection

The marriage of GCNs and AEs has resulted in great success in the field of community detection. GCNs provide high-order structural information via neighborhood aggregation for community representations, while AEs alleviate the over smoothing issue in GCNs to clarify community boundaries. For example, structural deep clustering network (SDCN) [125] includes a delivery operator that connects an AE and a GCN with the layers of a DNN in order to produce structure-aware representations. When SDCN integrates the structural information into deep clustering, it updates communities by applying a dual self-supervised optimization during backpropagation.

The GCN-based approach for unsupervised community detection (GUCD) [126] employs a semi-supervised MRF [11] as an encoder and a community-centric dual decoder to identify communities in attributed networks. The dual decoder reconstructs the network topology and node attributes.

One2Multi graph autoencoder for multi-view graph clustering (O2MAC) [74] deals with the multi-view graph by dividing it into multiple one-view graphs and assigning each an AE, named One2Multi. In the encoder, a GCN embeds a set of view-separated graphs. The decoders select the most informative graph with which to represent the multi-view graph. O2MAC is highly effective at capturing the shared features in multi-view graphs and improves clustering results through self-training optimization.

E. Graph Attention AE-Based Community Detection

Instead of integrating GCNs, this community detection category integrates GATs with AEs, where GAT serves as the encoder and ranks the importance of nodes within a neighborhood. For example, deep attentional embedded graph clustering (DAEGC) [127] exploits high-order neighbors in an AE equipped with graph attention to cluster communities during a self-training process. Similarly, graph embedding clustering with cluster-specificity distribution (GEC-CSD) [128] uses an AE also equipped with graph attention as a generator to learn community representations within an adversarial learning framework that also integrates self-training. Here, the discriminator ensures there is diversity in the cluster distributions.

In terms of multi-view attributes in networks, multi-view attribute graph convolution networks (MAGCN) [129] involves a two-pathway encoder. The first pathway encodes multi-view attributes with GATs to reduce noise. The second one learns consistent embeddings over multi-view attributes. Thus, noise and distribution variances are removed. Also in this vein is self-supervised graph convolutional network for multi-view clustering [130] which was developed from MAGCN but shares a coefficient matrix among different views.

Deep multi-graph clustering (DMGC) [131] introduces AEs to represent each graph with an attention coefficient in which node embeddings of multiple graphs cluster cross-graph centroids to obtain communities on a Cauthy distribution.

F. VAE-Based Community Detection

VAEs are extensions of AEs based on variational inference (e.g., mean and covariance of features) [132]. They were first introduced into graph learning as variational graph autoencoder (VGAE) [133] that assuming a prior distribution and applied GCN as the encoder through which the learned representation $\boldsymbol{Z}$ is a latent distribution. VAEs easily incorporate deeper nonlinear relationship information for community detection. For example, a triad (variational) graph autoencoder

(TGA/TVGA) [134] replaces the vanilla decoder with a new triad (variational) decoder that preserves the triadic closure property in communities.

Variational graph embedding and clustering with Laplacian eigenmaps (VGECLE) [135] divides the graph representation into mean and covariance while generatively detecting communities, to indicate each node's uncertainty of implicit relationships to its actual geographic position. With a mixture-of-Gaussian prior and a teacher–student (T-S) like regularization, VGECLE aims to let the node v_i (student) learn a distribution close to its neighbors' (teacher).

The deep generative latent feature relational model (DGLFRM) [136] and ladder gamma variational autoencoder for graphs (LGVG) [137] further capture the community membership strength of each node. DGLFRM models the sparse node embeddings by a beta-Bernoulli process that can detect overlapping communities and infer the number of communities. LGVG is devised to learn the multilayered and gamma-distributed embeddings so that it detects communities at fine-grained and coarse-grained granularities.

To capture the high-order structural features of communities, variational graph autoencoder for community detection (VGAECD) [138] employs a Gaussian mixture model (GMM) that generalizes the network generation process by introducing a community assignment parameter. VGAECD defines the joint probability of generative process, which can be detailed as

$$p(\boldsymbol{a}, z, c) = p(\boldsymbol{a}|z)p(z|c)p(c) \tag{22}$$

where $p(c) = \pi_c$ denotes the prior probability of a community C_c, and $p(z|c)$ is calculated from the Gaussian distribution corresponding to community C_c with a mean μ_c and a standard deviation σ_c from a GCN layer, i.e., $z \sim \mathcal{N}(\mu_c, \sigma_c^2 \boldsymbol{I})$. $\boldsymbol{a}$ with a posterior (reconstruction) probability $p(\boldsymbol{a}|z)$ is sampled from a multivariate Bernoulli distribution parametrized by μ_x learned from the decoder on z. The variational distribution is obtained from an overall loss function as

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{x}) = \mathbb{E}_{q(z|\boldsymbol{x},\boldsymbol{a})}[\log p(\boldsymbol{x}, \boldsymbol{a}|z)] - \text{KL}[q(c|\boldsymbol{x}) \| p(c|z)] \tag{23}$$

where the first term calculates the reconstruction loss, while the KL divergence minimizes the clustering loss. The evidence lower bound (ELBO) is maximized to optimize the function.

The training progress of VGAECD reveals a favored treatment for minimizing the reconstruction loss over the community loss, which leads to a suboptimal community detection result. To resolve this, an approach called optimizing variational graph autoencoder for community detection (VGAECD-OPT) [139] was proposed, which involves a dual optimization to learn the community-aware latent representation. The ELBO maximization then becomes

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{a}) = \mathbb{E}_{q(z,c|\boldsymbol{a})}[\log p(\boldsymbol{a}|z)] - \text{KL}[q(z, c|\boldsymbol{a}) \| p(z, c)]. \tag{24}$$

To overcome real-world sparsity and noise, a method called adversarially regularized (variational) graph autoencoder (ARGA/ARVGA) [140] embeds the latent representations $\boldsymbol{Z}$ based on the prior distribution p_z under the adversarial AE framework. $\boldsymbol{Z}$ is represented by a (variational) graph encoder aimed at a robust community detection. Meanwhile, a special discriminator ϕ_d is designed to adversarially distinguish between p_z guided embeddings $\boldsymbol{Z}$ and encodings on real samples $\phi_g(\boldsymbol{X}, \boldsymbol{A})$ via

$$\min_{\Theta_g} \max_{\Theta_d} \; \mathbb{E}_{z\sim p_z}[\log \phi_d(\boldsymbol{Z})] + \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\log(1 - \phi_d(\phi_g(\boldsymbol{X}, \boldsymbol{A})))]. \tag{25}$$

IX. Deep Nonnegative Matrix Factorization-Based Community Detection

In network embedding, nonnegative matrix factorization (NMF) [141] is a particular technique that factorizes an adjacency matrix $\boldsymbol{A}$ into two nonnegative matrices ($\boldsymbol{U} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{P} \in \mathbb{R}^{k\times n}$) with the nonnegative constraints that $\boldsymbol{U} \geq 0$ and $\boldsymbol{P} \geq 0$. The matrix $\boldsymbol{U}$ corresponds to the mapping between the given network and the community membership space. Each column in the matrix $\boldsymbol{P} = [p_{ij}]$ indicates the community membership for each node (v_i, C_j) in the probability of p_{ij}. NMF is applicable for disjoint and overlapping community detection. While real-world networks contain complicated topology information, traditional NMFs cannot fully uncover it to detect communities. Deep NMFs (DNMFs) [142] stack multiple layers of an NMF $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_p\}$ to capture pairwise node similarities of various aspects at various levels, like the deep layers in GCNs and AEs.

In the field of community detection, one model—deep autoencoder-like nonnegative matrix factorization (DANMF) [143]—is influential in unsupervised settings. In contrast to conventional NMF-based detection mappings, i.e., simple community membership, DANMF employs an AE framework to reconstruct networks on hierarchical mappings. The learning objective of community membership $\boldsymbol{P}_p$ and the hierarchical mappings $\{\boldsymbol{U}_i\}_1^p$ are trained by combining the reconstruction losses and a λ weighted graph regularization as follows:

$$\begin{aligned} \min_{\boldsymbol{P}_p, \boldsymbol{U}_i} \; & \mathcal{L}(\boldsymbol{P}_p, \boldsymbol{U}_i) = \|\boldsymbol{A} - \boldsymbol{U}_1 \ldots \boldsymbol{U}_p \boldsymbol{P}_p\|_F^2 \\ & + \left\|\boldsymbol{P}_p - \boldsymbol{U}_p^T \ldots \boldsymbol{U}_1^T \boldsymbol{A}\right\|_F^2 \\ & + \lambda \text{tr}\left(\boldsymbol{P}_p \boldsymbol{L} \boldsymbol{P}_p^T\right) \\ \text{s.t. } & \boldsymbol{P}_p \geq 0, \quad \boldsymbol{U}_i \geq 0, \quad \forall i = 1, \ldots, p \end{aligned} \tag{26}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\boldsymbol{L}$ represents the graph Laplacian matrix, and graph regularization focuses on the network topological similarity to cluster neighboring nodes. A further study [144] adds a sparsity constraint into the above-mentioned DNMF-based community detection.

Although DNMF provides a solution to map multiple factors in forming communities, the computational cost is relatively high on matrix factorizations. To address this issue, modularized deep nonnegative matrix factorization (MDNMF) [145] applies modularity directly into a basic multilayer deep learning structure targeting community detection. The node membership in communities $\boldsymbol{P}_p$ can be derived by minimizing the objective function in the following:

$$\begin{aligned} \mathcal{L} = & \|\boldsymbol{A} - \boldsymbol{U}_1 \ldots \boldsymbol{U}_p \boldsymbol{P}_p\|_F^2 + \alpha \left\|\boldsymbol{M}' - \boldsymbol{P}_p^T \boldsymbol{K}^T\right\|_F^2 \\ & - \beta \text{tr}(\boldsymbol{M}'^T \boldsymbol{B} \boldsymbol{M}') + \lambda \text{tr}\left(\boldsymbol{P}_p \boldsymbol{L} \boldsymbol{P}_p^T\right) \\ & \text{s.t. } \boldsymbol{P}_p \geq 0, \quad \boldsymbol{U}_i \geq 0, \quad \forall i = 1, \ldots, p \end{aligned} \tag{27}$$

where $\boldsymbol{B}$ is the modularity matrix and $\boldsymbol{M}'$ is its corresponding community membership matrix, and $\boldsymbol{K}$ is an extra nonnegative matrix combining modularity information so that DNMF can explore the hidden features of network topology.

TABLE II
SUMMARY OF COMMONLY USED EVALUATION METRICS IN COMMUNITY DETECTION

Metrics	Overlap	Ground Truth	Publications
NMI	×	Yes	[8], [11], [66]–[69], [71], [77], [78], [84], [86]–[89], [91], [93]–[95], [101], [104], [105], [110], [115], [118], [120], [122]–[124], [128], [130]
Overlapping-NMI	✓		[70], [74], [100], [125]–[127], [129], [131], [134], [138]–[140], [143], [145]
ACC	✓	Yes	[11], [67], [69], [71], [74], [75], [86], [88], [89], [91]–[95], [104], [110], [114], [123]–[131], [134], [135], [138]–[140], [143], [145]
Precision	✓	Yes	[8], [67], [88], [89], [104], [124], [134], [140]
Recall	✓	Yes	[124]
F1-Score	✓	Yes	[66], [67], [69], [73], [74], [83], [84], [88], [89], [94], [101], [103]–[105], [124], [125], [127], [134], [140]
ARI	✓	Yes	[67], [74], [77], [88], [89], [91], [93]–[95], [100], [101], [123]–[125], [127]–[130], [134], [140], [143]
Q	×	No	[18], [78], [115], [119], [120], [138], [139]
Extended-Q	✓		[107]
Jaccard	✓	Yes	[8], [73], [103]
CON	✓	No	[106], [138], [139]
TPR	✓	No	[138], [139]

"Overlap" indicates whether the metric can evaluate the overlapping community that is defined in Section II.
"Ground Truth" indicates whether the actual community structure of the network is required.

X. DEEP SPARSE FILTERING-BASED COMMUNITY DETECTION

Sparse filtering (SF) [146] is a two-layer learning model capable of handling high-dimensional graph data. The highly sparse input $\boldsymbol{A}$ with many zero elements is represented into lower dimensional feature vectors $\boldsymbol{h}_i$ with nonzero values. To explore deeper information, such as community membership, deep SF (DSF) stacks multiple hidden layers to finetune the hyperparameters Θ and extensively smooth the data distributions $Pr(\boldsymbol{h}_i)$.

As a representative method, community discovery based on deep sparse filtering (DSFCD) [147] operates over three phases: network representation, community feature mapping, and community discovery. The network representation phase operates on an adjacency matrix $\boldsymbol{A}$, a modularity matrix $\boldsymbol{B}$, and two similarity matrices $\boldsymbol{S}$ and $\boldsymbol{S}'$, respectively. The best representation is selected as the input into the DSF for community feature mapping represented on each node $\boldsymbol{h}_i$. Meanwhile, $\boldsymbol{h}_i$ preserves the node similarity in the original network $\boldsymbol{A}$ and latent community membership features. The node similarity is modeled in the loss function

$$\mathcal{L} = \sum_i \|\boldsymbol{h}_i\|_1 + \lambda \sum_i \mathrm{KL}\left(\boldsymbol{h}_i, \boldsymbol{h}_j^*\right) \tag{28}$$

where $\|\cdot\|_1$ is the L_1 norm penalty to optimize sparseness, and $\boldsymbol{h}_j^*$ denotes the representation of the most similar node to v_i from the perspective of distances measured by KL divergence. When the learning process is optimized on the minimized loss, similar nodes are clustered into communities. In experiments, DSFCD discovers communities in higher accuracy than SF.

XI. PUBLISHED RESOURCES

In this section, we summarize the popular and available benchmark datasets, evaluation metrics, and open-source implementations from the surveyed literature in Sections V–X. Further details are provided in Appendices B–D of the Supplementary Material.

A. Datasets

The field uses both real-world datasets and synthetic datasets in its development. Real-world datasets for community detection experiments have been collected from real-world applications. These test the performances of proposed methods for practical purposes. Synthetic datasets have also been generated by specific models based on manually designed rules. Particular functions can be tested by these datasets. The state-of-the-art popular real-world datasets can be divided into citation/co-authorship networks, social networks (online and offline), webpage networks, and product co-purchasing networks. The typical datasets covering various network shapes (i.e., unattributed, attributed, multi-view, signed, heterogeneous, and dynamic) are summarized in Table VI of the Supplementary Material. Related descriptions are detailed in Appendix B-A (real-world datasets) of the Supplementary Material. GN networks [3] and Lancichinetti–Fortunato–Radicchi (LFR) networks [148], [149] are two widely applied synthetic benchmarks, described in Appendix B-B (synthetic benchmark datasets) of the Supplementary Material.

B. Evaluation Metrics

This section summarizes 12 of the most popular and commonly applied evaluation metrics from the reviewed deep community detection literature (details in Appendix C of the Supplementary Material). Which metric is used varies depending on the types of the community (i.e., disjoint or overlapping) and whether ground truth information is available [150]. Table II summarizes publications by metric according to these criteria.

C. Open-Source Implementations

The open-source implementations are summarized by tool, category, repository link in Table III. The majority of implementations are under Python 3.x, including PyTorch, TensorFlow, and Keras. A few implementations use MATLAB. Each implementation project is briefly described and organized by tools and deep learning models in Appendix D of the Supplementary Material.

XII. PRACTICAL APPLICATIONS

Community detection has many applications across different tasks and domains, as summarized in Fig. 9. We now detail some typical applications in the following areas.

A. Recommendation Systems

Community structure plays a vital role in graph-based recommendation systems [151]–[153], as the community members may have similar interests and preferences. By detecting relations between nodes (i.e., users–users, items–items, and users–items), models, such as CayleyNets [92] and UWMNE/WMCNE-LE [110], produce high-quality recommendations.

TABLE III
SUMMARY OF OPEN-SOURCE IMPLEMENTATIONS

Category	Tool	Method	URL
GCN	PyTorch	AGE [91]	https://github.com/thunlp/AGE
		LGNN [85]	https://github.com/zhengdao-chen/GNN4CD
		NOCD [87]	https://github.com/shchur/overlapping-community-detection
	TensorFlow	AGC [69]	https://github.com/karenlatong/AGC-master
		CayleyNet [92]	https://github.com/amoliu/CayleyNet
GAT	PyTorch	CP-GNN [101]	https://github.com/RManLuo/CP-GNN
		DMGI [66]	https://github.com/pcy1302/DMGI
		MAGNN [77]	https://github.com/cynricfu/MAGNN
		HDMI [70]	https://github.com/baoyujing/HDMI
		HeCo [100]	https://github.com/liun-online/HeCo
GAN	PyTorch	SEAL [103]	https://github.com/yzhang1918/kdd2020seal
	TensorFlow	CommunityGAN [105]	https://github.com/SamJia/CommunityGAN
Stacked AE	TensorFlow	DANE [114]	https://github.com/gaoghc/DANE
		DIME [116]	http://www.ifmlab.org/files/code/Aligned-Autoencoder.zip
	MATLAB	DNE-SBP [72]	https://github.com/shenxiaocam/Deep-network-embedding-for-graph-representation-learning-in-signed-networks
		semi-DRN [68]	http://yangliang.github.io/code/DC.zip
Sparse AE	PyTorch	GraphEncoder [118]	https://github.com/zepx/graphencoder
Denoising AE	Keras	DNGR [122]	https://github.com/MdAsifKhan/DNGR-Keras
	MATLAB	MGAE [124]	https://github.com/FakeTibbers/MGAE
Graph Convolutional AE	PyTorch	SDCN [125]	https://github.com/bdy9527/SDCN
	TensorFlow	O2MAC [74]	https://github.com/songzuolong/WWW2020-O2MAC
Graph Attention AE	TensorFlow	DMGC [131]	https://github.com/flyingdoog/DMGC
		SGCMC [130]	https://github.com/xdweixia/SGCMC
VAE	TensorFlow	ARGA/ARVGA [140]	https://github.com/Ruiqi-Hu/ARGA
		DGLFRM [136]	https://github.com/nikhil-dce/SBM-meet-GNN

B. Biochemistry

In this field, nodes represent proteins/atoms in compound or molecule graphs, while edges denote their interactions. Community detection can identify complexes of new proteins [29] and chemical compounds [28] which are functional in regional organs (e.g., in brain [154]) or pathogenic factor of a disease (e.g., community-based lung cancer detection [155]). Given various tumor types on genomic datasets, Haq and Wang [156] show relevance between community survival rates and the distributions of different types of tumor over different communities.

C. Online Social Networks

Analyzing online social activities can identify online communities and correlate them in the real world [157], [158]. In practice, online social networks [3], such as Twitter and Facebook, reveal similar interests among online users when individual preferences are available. Meanwhile, community detection can be used for online privacy [159] and to identify **criminals** based on their online behaviors [160]. Waskiewicz [161] demonstrates that community detection can be used to reveal those who support and diffuse criminal ideas and even those who may be terrorists.

D. Community Deception

Community deception [162] refers to groups of users in social networks, such as Facebook, who hide from community detection. Deception can either be harmful to virtual communities or harmless. Some community deception even provides justifiable benefits. Residual entropy minimization (REM) nullifies community detection algorithms [163] via an approach based on community structural entropy, while Fionda and Pirro [164] proposed a community deception model via defining safeness in large networks.

E. Community Search

The aim of community search is to find the communities containing query nodes from large-scale networks in real time [165]. For example, one might search for the interest groups (communities) in a social network to which different users (query nodes) belong. To this end, communities are formed in a temporal manner based on user interests. Several techniques have been applied to such scenarios. Local community search [166] assumes one query node at a time and expands the search space around it. The strategy is attempted repeatedly until the community finds all its members. Alternatively, attributed truss communities (ATC) [167] interconnects communities on query nodes with similar node attributes.

XIII. FUTURE DIRECTIONS

Although deep learning has brought community detection into a prosperous era, several open issues need further investigation. In this section, we recommend 12 future directions of promising research.

A. Unknown Number of Communities

The cost of acquiring labeled data is still very high, so in most real-world community detection scenarios, the majority of data are unlabeled. This means the number of communities K is usually unknown. Unsupervised detection methods in deep learning provide an effective way to handle

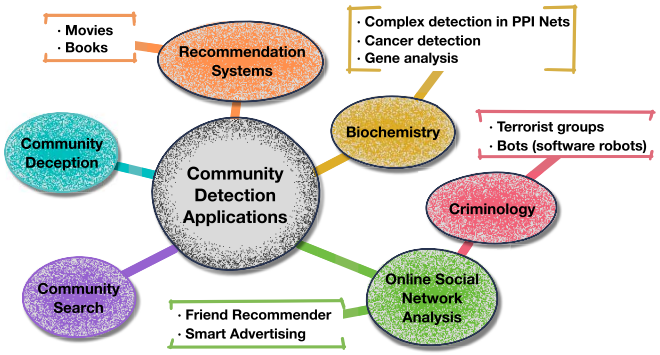


Fig. 9. Practical applications of community detection.

these scenarios, but they generally need to specify K as the prior knowledge. This phenomenon brings us a catch-22: approaches require prior knowledge which does not exist. Therefore, there is an urgent demand to find ways of handling the issue of an unknown number of communities.

Opportunities: Analyzing network topological structures offer a potential solution to tackle this challenge and some research efforts have been made in this direction [18]. Typically, these methods perform random walks to determine some preliminary communities and then refine the detection results. Nevertheless, when it comes to disconnected networks in practice, random walks cannot involve every node, and thus, detection performance degrades. This is, therefore, an open issue that calls for a complete solution and further research.

B. Community Embedding

Node embedding methods traditionally preserve the neighborhood information of nodes but ignore the structural information of communities [168]. To this end, designing community-aware learning processes to characterize community information can improve the accuracy of community detection [169].

Opportunities: To date, a few works integrate community embedding into a deep learning model; therefore, more effort is needed to further research in this promising area. Because community embedding that learns representations for each community increases computational costs, one thing that is needed is to develop computationally fast algorithms. Furthermore, the optimization mechanism for the hyperparameters in the community embeddings needs a new design so that deep community detection can meet future expectations.

C. Hierarchical Networks

Networks, such as the Web, often show a treelike hierarchical organization of communities at different scales, in which small communities at lower levels group together to form larger ones in higher levels [170]. Hence, community detection is required to detect communities from low to high levels.

Opportunities: Traditional methods generally follow one of three work lines: 1) estimating the hierarchy directly; 2) recursively merging communities in a bottom-up fashion; and 3) recursively splitting communities in a top-down fashion. Their performance is limited by either a large number of parameters or the requirements for network density [171]. Network embedding shows the efficiency of handling these issues [76], [172]. However, preserving the community hierarchy into the embeddings remains unsolved [172]. With the advancement of high-order relationship representations [173], we believe deep learning models can facilitate the development of hierarchical community detection.

D. Multilayer Networks

Realistic systems are always multilayered, and extracting this property means extracting multiple interdependent graphs—one for each layer. Here, the layers represent different types of interactions among entities, and the cross-layer links reflect dependencies among entities [174]. Taking a multilayer social network where the nodes are individuals as an example, the intra-layer connections would represent friendship, kinship, schoolmates, and so on, while the inter-layer connections align the same individuals [175]. Thus, community detection in such networks can leverage more information to benefit the results.

Opportunities: In contrast to the leaps and bounds in community detection for single-layer networks, research on multilayer networks is still in its infancy [5]. A rudimentary community detection process with multilayered networks is to fuse the information in multiple layers into one layer and follow through with a single-layer detection method. In deep learning architecture, a similar solution is to flatten the multilayer information into low-dimensional representations with the following general open issues: 1) differences among the interaction types; 2) varying levels of sparsity in layers; 3) possible connections across layers; and 4) the scalability scheme of layers.

E. Heterogeneous Networks

Another type of more complex network is one that involves heterogeneous information [176], [177]. Such networks characterize the relationships between different types of entities, such as the relationships between actors and movies. Since community detection methods designed for homogeneous networks cannot be employed directly due to a lack of capacity to model the complex structural and semantic information, community detection research on heterogeneous networks is challenging.

Opportunities: Meta paths are a promising avenue for dealing with diverse semantic information. They describe a composite relationship between different node types. Meta paths also allow deep models to represent nodes by aggregating information from first-order structures via different meta paths and can be used to measure a node's similarity to its community cluster [77], [116], [178]. However, the basis on which to select the most meaningful meta paths remains an open problem. Future efforts should focus on a flexible schema for meta path selection and novel models that can exploit various types of relationships.

F. Network Heterophily

Network heterophily [179] can be interpreted as a phenomenon, where connected nodes belong to different communities or are characterized by dissimilar features. For example, fraudsters often intentionally make connections with normal users to avoid being discovered. For community detection, the boundary nodes connected across communities comply with this property. It is significant to capture network heterophily providing valuable information on community division.

Opportunities: Since most methods heavily rely on homophily, they assume connected nodes share more similarities and are more likely from the same community. Deep learning methods that exploit network heterophily are expected for better community detection performance.

G. Topologically Incomplete Networks

Relationships in real-world scenarios are not always available, which leads to incomplete network topology and isolated subgraphs [8]. For example, PPI networks are usually incomplete since monitoring all protein-protein interactions is expensive [180]. Deriving meaningful knowledge of communities from limited topological information is crucial in this case.

Opportunities: The requirement of having a complete network topology reduces the applicability of community detection methods, especially those based on neighborhood aggregations. To this end, deep learning methods should be further developed with an information recovery mechanism to achieve accurate community detection on TINs.

H. Cross-Domain Networks

That many networks are similar, e.g., Facebook and Twitter, is an underexploited benefit. In cases of similar networks, each network has its independent domain, but borrowing knowledge from an information-rich domain could benefit network learning in the target domain [181]. Therefore, community detection should be encouraged to develop its own deep learning models to improve detection results [115].

Opportunities: In community adaptation, transferring latent representations from a source to a target domain faces the following challenges: 1) lack of explicit community structures; 2) node label shortages; 3) missing a community's ground truth; 4) poor representation performance caused by the inferior network structures; and 5) deep learning with unsuitable small-scale networks. Future methods might aim to measure cross-domain co-efficiency and distribution shifts while controlling computational costs.

I. Attributed Multi-Views Networks

Real-world networks are complex and contain distinctively featured data [182]. Attributed multi-views networks provide a new way to describe relational information in multiple informative views, each containing a type of attributes [183]. Communities are initially detected on each single-view network and deeply encoded over multiple views in an expensive way. Hence, cheap community detection methods aim to exploit the multi-views complementarity [184].

Opportunities: A straightforward workflow is to combine representations learned separately from each view. However, this may create noise and redundancy to the detriment of the detection results. To develop an integrated framework, deep learning approaches try to extract consistent information among multiple views by learning a common clustering embedding for community detection [129]. Since multi-view node attributes still require a better integration scheme in the learning process, future works on networks with multi-view attributes need to develop a suitable integration scheme within the learning process. Meanwhile, more study is needed on the underexplored issue of global representation for community detection over multi-views to avoid suboptimization.

J. Signed Networks

Researchers are increasingly noting that not all occurring connections reflect positive relationships [185]–[187]. In general, friendship indicates a positive sentiment, such as like and support, while enemies are associated with negative attitudes, such as dislike. These distinctions are reflected on signed edges in signed networks [188]. As the impacts of positive and negative ties are different, existing community detection methods working on unsigned networks are not applicable to signed networks.

Opportunities: To detect communities in signed networks, the main challenges lie in adapting negative ties. Deep learning techniques should be exploited to represent both ties. Negative ties offer distinguishing community knowledge in learning signed network embeddings [72]. What is needed are methods of coping with signed edges and also methods that can automatically identify positive/negative information on edges.

K. Dynamic Networks

Dynamic structures and temporal semantic features mean nodes and edges may be added or removed as time passes, which changes community structures [189], [190]. Accordingly, deep learning models need to be able to rapidly capture the network evolutions to explore community changes. In the course of our literature review, we only came across one study that touches on this topic. It concerns the design of an evolutionary AE to discover smoothly changing community structures [78].

Opportunities: Both deep learning and community detection need to handle shifting distributions and evolving data scales. Yet repeated training over snapshots is expensive and slow. The technical challenges of detecting communities in a dynamic network mainly come from the dynamic control of both spatial and temporal parts. Future directions include: 1) the analysis of spatial changes on communities; 2) the recognition of dynamic network patterns; and 3) tracks of network dynamics and community detection robustness.

L. Large-Scale Networks

Large-scale networks may contain a massive number of nodes, edges, and communities. Their inherent characteristics can influence the performance of deep community detection. For instance, being scale-free or having a power-law degree distribution [22], [191], such as with many social networks, can be highly problematic. Here, scalability is a crucial issue [18]. A suggested direction for overcoming this problem is to develop a flexible deep learning approach aimed at collaborative computing with high robustness.

Opportunities: Common dimension reduction strategies in deep learning, such as matrix low-rank approximation, cannot cope with the high-dimensional relationships in large-scale networks, and the current solutions based on distributed computing are too expensive. Consequently, there is a crying need for novel deep learning frameworks, models, and algorithms that exceed the current benchmarks in terms of precision and speed.

XIV. Conclusion

This survey provides a comprehensive overview of the state-of-the-art community detection methods. Over the past decade, many notable methods based on deep learning models have been developed—a trend that has completely changed the

field of community detection and its framework. According to our survey of high-impact international conferences and peer-reviewed journals, these methods have significantly increased the effectiveness, efficiency, robustness, and applicability of community detection. New techniques are much more flexible to use than traditional methods, and a larger volume of data can be leveraged during preprocessing. Such overwhelming change warrants a new taxonomy, and, for this reason, we selectively summarized the reviewed methods into six novel categories. The developments in each category to date along with the contributions of each direction are discussed in turn. We also compiled a handy guide to the commonly used resources based on the reviewed literature, including datasets, evaluation metrics, and open-source implementations. The survey concludes with insights into a range of community detection applications and, to stimulate further research into this area, we identified 12 open research directions.

REFERENCES

[1] S. A. Rice, "The identification of blocs in small political bodies," *Amer. Political Sci. Rev.*, vol. 21, no. 3, pp. 619–627, 1927.

[2] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, 2004, Art. no. 026113.

[3] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 99, no. 12, pp. 7821–7826, 2001.

[4] F. Liu *et al.*, "Deep learning for community detection: Progress, challenges and opportunities," in *Proc. IJCAI*, 2020, pp. 4981–4987.

[5] X. Huang, D. Chen, T. Ren, and D. Wang, "A survey of community detection methods in multilayer networks," *Data Min. Knowl. Disc.*, vol. 35, no. 1, pp. 1–45, 2021.

[6] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 1–35, 2013.

[7] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig, "Community detection in networks: A multidisciplinary review," *J. Netw. Comput. Appl.*, vol. 108, pp. 87–111, 2018.

[8] X. Xin, C. Wang, X. Ying, and B. Wang, "Deep community detection in topologically incomplete networks," *Physica A*, vol. 469, pp. 342–352, 2017.

[9] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. IJCAI*, vol. 2015, pp. 2111–2117.

[10] P. Chunaev, "Community detection in node-attributed social networks: A survey," *Comput. Sci. Rev.*, vol. 37, 2020, Art. no. 100286.

[11] D. Jin, Z. Liu, W. Li, D. He, and W. Zhang, "Graph convolutional networks meet Markov random fields: Semi-supervised community detection in attribute networks," in *Proc. AAAI*, 2019, pp. 152–159.

[12] G. Rossetti and R. Cazabet, "Community discovery in dynamic networks: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–37, 2018.

[13] C. Pizzuti, "Evolutionary computation for community detection in networks: A review," *IEEE Trans. Evol. Comput.*, vol. 22, no. 3, pp. 464–483, 2018.

[14] Q. Cai, L. Ma, M. Gong, and D. Tian, "A survey on network community detection based on evolutionary computation," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 2, pp. 84–98, 2016.

[15] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.

[16] D. Lusseau, "The emergent properties of a dolphin social network," *Proc. R. Soc. Lond. B*, vol. 270, no. 2, pp. S186–S188, 2003.

[17] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, 2004, Art. no. 066111.

[18] V. Bhatia and R. Rani, "DFuzzy: A deep learning-based fuzzy clustering model for large graphs," *Knowl. Inf. Syst.*, vol. 57, no. 1, pp. 159–181, 2018.

[19] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Proc. NIPS*, 2012.

[20] C. Nicolini, C. Bordier, and A. Bifone, "Community detection in weighted brain connectivity networks beyond the resolution limit," *NeuroImage*, vol. 146, pp. 28–39, 2017.

[21] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annu. Rev. Sociol.*, vol. 27, no. 1, pp. 415–444, 2001.

[22] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[23] J. Xie and B. K. Szymanski, "Towards linear time overlapping community detection in social networks," in *Proc. PAKDD*, 2012, pp. 25–36.

[24] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proc. ICDM*, 2013, pp. 1151–1156.

[25] P. Chen and S. Redner, "Community structure of the physical review citation network," *J. Informetr.*, vol. 4, no. 3, pp. 278–290, 2010.

[26] G. Namata, B. London, L. Getoor, B. Huang, and U. EDU, "Query-driven active surveying for collective classification," in *Proc. KDD Workshop on Mining and Learning with Graphs*, vol. 8, 2012.

[27] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[28] R. Guimera and L. A. N. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, pp. 895–900, 2005.

[29] J. Chen and B. Yuan, "Detecting functional modules in the yeast protein–protein interaction network," *Bioinformatics*, vol. 22, pp. 2283–2290, 2006.

[30] O. Sporns and R. F. Betzel, "Modular brain networks," *Annu. Rev. Psychol.*, vol. 67, pp. 613–640, 2016.

[31] A. A. Amini, A. Chen, P. J. Bickel, and E. Levina, "Pseudo-likelihood methods for community detection in large sparse networks," *Ann. Statist.*, vol. 41, no. 4, pp. 2097–2122, 2013.

[32] S. C. de Lange, M. A. de Reus, and M. P. van den Heuvel, "The Laplacian spectrum of neural networks," *Front. Comput. Neurosci.*, vol. 7, 2014.

[33] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Soc. Networks*, vol. 5, pp. 109–137, 1983.

[34] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, 2011, Art. no. 016107.

[35] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *J. Mach. Learn. Res.*, vol. 9, no. 65, pp. 1981–2014, 2008.

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

[37] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Phys. Rep.*, vol. 659, pp. 1–44, 2016.

[38] E. Abbe, "Community detection and stochastic block models: Recent developments," *J. Mach. Learn. Res.*, vol. 18, no. 177, pp. 1–86, 2018.

[39] S. E. Garza and S. E. Schaeffer, "Community detection with the label propagation algorithm: A survey," *Physica A*, vol. 534, 2019, Art. no. 122058.

[40] J. H. Chin and K. Ratnavelu, "A semi-synchronous label propagation algorithm with constraints for community detection in complex networks," *Sci. Rep.*, vol. 7, no. 1, pp. 1–12, 2017.

[41] D. Jin *et al.*, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Trans. Knowl. Data Eng.*, early access, 2021.

[42] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Phys. Rep.*, vol. 533, no. 4, pp. 95–142, 2013.

[43] P. Bedi and C. Sharma, "Community detection in social networks," *WIREs Data Mining Knowl. Discov.*, vol. 6, no. 3, pp. 115–135, 2016.

[44] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, 1970.

[45] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM J. Alg. Disc. Meth.*, vol. 3, no. 4, pp. 541–550, 1982.

[46] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, 2004, Art. no. 066133.

[47] F. D. Zarandi and M. K. Rafsanjani, "Community detection in complex networks using structural similarity," *Physica A*, vol. 503, pp. 882–891, 2018.

[48] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proc. ISCIS*, 2005, pp. 284–293.

[49] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 105, no. 2, pp. 1118–1123, 2008.

[50] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, 2007, Art. no. 036106.

[51] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.

[52] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proc. KDD*, 2007, pp. 824–833.

[53] X. Wang, G. Liu, J. Li, and J. P. Nees, "Locating structural centers: A density-based clustering method for community detection," *PLoS ONE*, vol. 12, no. 1, 2017, Art. no. e0169355.

[54] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 103, no. 23, pp. 8577–8582, 2006.

[55] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 2008, no. 10, 2008, Art. no. P10008.

[56] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[57] S. Boettcher and A. G. Percus, "Optimization with extremal dynamics," *Complexity*, vol. 8, no. 2, pp. 57–62, 2002.

[58] M. E. J. Newman, "Spectral methods for community detection and graph partitioning," *Phys. Rev. E*, vol. 88, no. 4, 2013, Art. no. 042822.

[59] F. Liu, J. Wu, C. Zhou, and J. Yang, "Evolutionary community detection in dynamic social networks," in *Proc. IJCNN*, 2019, pp. 1–7.

[60] Z. Li and J. Liu, "A multi-agent genetic algorithm for community detection in complex networks," *Physica A*, vol. 449, pp. 336–347, 2016.

[61] S. Sobolevsky, R. Campari, A. Belyi, and C. Ratti, "General optimization technique for high-quality community detection in complex networks," *Phys. Rev. E*, vol. 90, no. 1, 2014, Art. no. 012811.

[62] L. N. F. Ana and A. K. Jain, "Robust data clustering," in *Proc. CVPR*, 2003, pp. 1–6.

[63] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," 2011, *arXiv:1110.2515*.

[64] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, no. 3, pp. 497–515, 2004.

[65] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[66] C. Park, D. Kim, J. Han, and H. Yu, "Unsupervised attributed multiplex network embedding," in *Proc. AAAI*, 2020, pp. 5371–5378.

[67] L. Yang, Y. Wang, J. Gu, C. Wang, X. Cao, and Y. Guo, "JANE: Jointly adversarial network embedding," in *Proc. IJCAI*, 2020, pp. 1381–1387.

[68] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *Proc. IJCAI*, 2016, pp. 2252–2258.

[69] X. Zhang, H. Liu, Q. Li, and X.-M. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proc. IJCAI*, 2019, pp. 4327–4333.

[70] B. Jing, C. Park, and H. Tong, "HDMI: Higher-order deep multiplex infomax," in *Proc. WWW*, 2021, pp. 2414–2424.

[71] H. Gao, J. Pei, and H. Huang, "ProGAN: Network embedding via proximity generative adversarial network," in *Proc. KDD*, 2019, pp. 1308–1316.

[72] X. Shen and F.-L. Chung, "Deep network embedding for graph representation learning in signed networks," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1556–1568, 2020.

[73] C. Yang, M. Liu, Z. Wang, L. Liu, and J. Han, "Graph clustering with dynamic embedding," 2017, *arXiv:1712.08249*.

[74] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2Multi graph autoencoder for multi-view graph clustering," in *Proc. WWW*, 2020, pp. 3070–3076.

[75] G. Sperlí, "A deep learning based community detection approach," in *Proc. SAC*, 2019, pp. 1107–1110.

[76] L. Du, Z. Lu, Y. Wang, G. Song, Y. Wang, and W. Chen, "Galaxy network embedding: A hierarchical community structure preserving approach," in *Proc. IJCAI*, 2018, pp. 2079–2085.

[77] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proc. WWW*, 2020, pp. 2331–2341.

[78] Z. Wang, C. Wang, C. Gao, X. Li, and X. Li, "An evolutionary autoencoder for dynamic community detection," *Sci. China Inf. Sci.*, vol. 63, no. 11, pp. 1–16, 2020.

[79] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1995, p. 3361.

[80] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017.

[81] L. Getoor, "Link-based classification," in *Advanced Methods for Knowledge Discovery From Complex Data*. London, U.K.: Springer, 2005, pp. 189–207.

[82] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

[83] A. De Santo, A. Galli, V. Moscato, and G. Sperlì, "A deep learning approach for semi-supervised community detection in online social networks," *Knowl.-Based Syst.*, vol. 229, 2021, Art. no. 107345.

[84] B. Cai, Y. Wang, L. Zeng, Y. Hu, and H. Li, "Edge classification based on convolutional neural networks for community detection in complex network," *Physica A*, vol. 556, 2020, Art. no. 124826.

[85] Z. Chen, L. Li, and J. Bruna, "Supervised community detection with line graph neural networks," in *Proc. ICLR*, 2019.

[86] X. Wang, J. Li, L. Yang, and H. Mi, "Unsupervised learning for community detection in attributed networks based on graph convolutional network," *Neurocomputing*, vol. 456, pp. 147–155, 2021.

[87] O. Shchur and S. Günnemann, "Overlapping community detection with graph neural networks," in *Proc. DLG Workshop, KDD*, 2019.

[88] R. Hu, S. Pan, G. Long, Q. Lu, L. Zhu, and J. Jiang, "Going deep: Graph convolutional ladder-shape networks," in *Proc. AAAI*, pp. 2838–2845, 2020.

[89] Y. Liu, X. Wang, S. Wu, and Z. Xiao, "Independence promoted graph disentangled networks," in *Proc. AAAI*, 2020, pp. 4916–4923.

[90] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proc. SDM*, 2017, pp. 633–641.

[91] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proc. KDD*, 2020, pp. 976–985.

[92] R. Levie, M. Federico, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, 2018.

[93] X. Zhang, H. Liu, X.-M. Wu, X. Zhang, and X. Liu, "Spectral embedding network for attributed graph clustering," *Neural Netw.*, vol. 142, pp. 388–396, 2021.

[94] T. Zhang, Y. Xiong, J. Zhang, Y. Zhang, Y. Jiao, and Y. Zhu, "CommDGI: Community detection oriented deep graph infomax," in *Proc. CIKM*, 2020, pp. 1843–1852.

[95] H. Zhao, X. Yang, Z. Wang, E. Yang, and C. Deng, "Graph debiased contrastive learning with joint representation clustering," in *Proc. IJCAI*, 2021, pp. 3434–3440.

[96] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018.

[97] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015.

[98] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proc. CVPR*, 2016, pp. 4651–4659.

[99] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NIPS*, 2017.

[100] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *Proc. KDD*, 2021, pp. 1726–1736.

[101] L. Luo, Y. Fang, X. Cao, X. Zhang, and W. Zhang, "Detecting communities from heterogeneous graphs: A context path-based graph neural network model," in *Proc. CIKM*, 2021, pp. 1170–1180.

[102] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014.

[103] Y. Zhang *et al.*, "SEAL: Learning heuristics for community detection with generative adversarial networks," in *Proc. KDD*, 2020, pp. 1103–1113.

[104] M. Shi, Y. Tang, X. Zhu, D. Wilson, and J. Liu, "Multi-class imbalanced graph convolutional network learning," in *Proc. IJCAI*, 2020, pp. 2879–2885.

[105] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "CommunityGAN: Community detection with generative adversarial nets," in *Proc. WWW*, 2019, pp. 784–794.

[106] J. Wang, J. Cao, W. Li, and S. Wang, "CANE: Community-aware network embedding via adversarial training," *Knowl. Inf. Syst.*, vol. 63, no. 2, pp. 411–438, 2021.

[107] J. Chen *et al.*, "Self-training enhanced: Network embedding and overlapping community detection with adversarial learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2021.

[108] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[109] F. Liu, J. Wu, S. Xue, C. Zhou, J. Yang, and Q. Sheng, "Detecting the evolving community structure in dynamic social networks," *World Wide Web*, vol. 23, no. 2, pp. 715–733, 2020.

[110] D. Jin, M. Ge, L. Yang, D. He, L. Wang, and W. Zhang, "Integrative network embedding via deep joint reconstruction," in *Proc. IJCAI*, 2018, pp. 3407–3413.

[111] P. Vanhems *et al.*, "Estimating potential infection transmission routes in hospital wards using wearable proximity sensors," *PLoS ONE*, vol. 8, no. 9, 2013, Art. no. e73970.

[112] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, "What's in a crowd? Analysis of face-to-face behavioral networks," *J. Theor. Biol.*, vol. 271, no. 1, pp. 166–180, 2011.

[113] B. Klimt and Y. Yang, "Introducing the enron corpus," in *Proc. CEAS*, 2004.

[114] H. Gao and H. Huang, "Deep attributed network embedding," in *Proc. IJCAI*, 2018, pp. 3364–3370.

[115] Y. Xie, X. Wang, D. Jiang, and R. Xu, "High-performance community detection in social networks using a deep transitive autoencoder," *Inf. Sci.*, vol. 493, pp. 75–90, 2019.

[116] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and S. Y. Philip, "BL-MNE: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder," in *Proc. ICDM*, 2017, pp. 605–614.

[117] A. Ng, "Sparse autoencoder," *CS294A Lect. Notes*, vol. 72, pp. 1–19, 2011.

[118] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. AAAI*, 2014, pp. 1293–1299.

[119] S. Li, L. Jiang, X. Wu, W. Han, D. Zhao, and Z. Wang, "A weighted network community detection algorithm based on deep learning," *Appl. Math. Comput.*, vol. 401, 2021, Art. no. 126012.

[120] R. Xu, Y. Che, X. Wang, J. Hu, and Y. Xie, "Stacked autoencoder-based community detection method via an ensemble clustering framework," *Inf. Sci.*, vol. 526, pp. 151–165, 2020.

[121] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. ICML*, 2008, pp. 1096–1103.

[122] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. AAAI*, 2016, pp. 1145–1152.

[123] B. Li, D. Pi, Y. Lin, and L. Cui, "DNC: A deep neural network-based clustering-oriented network embedding algorithm," *J. Netw. Comput. Appl.*, vol. 173, 2021, Art. no. 102854.

[124] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. CIKM*, 2017, pp. 889–898.

[125] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proc. WWW*, 2020, pp. 1400–1410.

[126] D. He *et al.*, "Community-centric graph convolutional network for unsupervised community detection," in *Proc. IJCAI*, 2020, pp. 3515–3521.

[127] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. IJCAI*, 2019, pp. 3670–3676.

[128] H. Xu, W. Xia, Q. Gao, J. Han, and X. Gao, "Graph embedding clustering: Graph attention auto-encoder with cluster-specificity distribution," *Neural Netw.*, vol. 142, pp. 221–230, 2021.

[129] J. Cheng, Q. Wang, Z. Tao, D. Xie, and Q. Gao, "Multi-view attribute graph convolution networks for clustering," in *Proc. IJCAI*, 2020, pp. 2973–2979.

[130] W. Xia, Q. Wang, Q. Gao, X. Zhang, and X. Gao, "Self-supervised graph convolutional network for multi-view clustering," *IEEE Trans. Multimedia*, early access, 2021.

[131] D. Luo, J. Ni, S. Wang, Y. Bian, X. Yu, and X. Zhang, "Deep multi-graph clustering via attentive cross-graph association," in *Proc. WSDM*, 2020, pp. 393–401.

[132] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014.

[133] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS*, 2016.

[134] H. Shi, H. Fan, and J. T. Kwok, "Effective decoding in graph auto-encoder using triadic closure," in *Proc. AAAI*, 2020, pp. 906–913.

[135] Z. Chen, C. Chen, Z. Zhang, Z. Zheng, and Q. Zou, "Variational graph embedding and clustering with Laplacian eigenmaps," in *Proc. IJCAI*, 2019, pp. 2144–2150.

[136] N. Mehta, L. C. Duke, and P. Rai, "Stochastic blockmodels meet graph neural networks," in *Proc. ICML*, 2019, pp. 4466–4474.

[137] A. Sarkar, N. Mehta, and P. Rai, "Graph representation learning via ladder gamma variational autoencoders," in *Proc. AAAI*, 2020, pp. 5604–5611.

[138] J. J. Choong, X. Liu, and T. Murata, "Learning community structure with variational autoencoder," in *Proc. ICDM*, 2018, pp. 69–78.

[139] J. J. Choong, X. Liu, and T. Murata, "Optimizing variational graph autoencoder for community detection," in *Proc. IEEE BigData*, 2019, pp. 5353–5358.

[140] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. IJCAI*, 2018, pp. 2609–2615.

[141] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[142] H. A. Song, B.-K. Kim, T. L. Xuan, and S.-Y. Lee, "Hierarchical feature extraction by multi-layer non-negative matrix factorization network for classification task," *Neurocomputing*, vol. 165, pp. 63–74, 2015.

[143] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. CIKM*, 2018, pp. 1393–1402.

[144] B.-J. Sun, H. Shen, J. Gao, W. Ouyang, and X. Cheng, "A non-negative symmetric encoder-decoder approach for community detection," in *Proc. CIKM*, 2017, pp. 597–606.

[145] J. Huang, T. Zhang, W. Yu, J. Zhu, and E. Cai, "Community detection based on modularized deep nonnegative matrix factorization," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 35, no. 2, 2021, Art. no. 2159006.

[146] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng, "Sparse filtering," in *Proc. NIPS*, 2011.

[147] Y. Xie, M. Gong, S. Wang, and B. Yu, "Community discovery in networks with deep sparse filtering," *Pattern Recognit.*, vol. 81, pp. 50–59, 2018.

[148] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, 2008, Art. no. 046110.

[149] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, no. 1, 2009, Art. no. 016118.

[150] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.

[151] C.-Y. Liu, C. Zhou, J. Wu, Y. Hu, and L. Guo, "Social recommendation with an essential preference space," in *Proc. AAAI*, 2018, pp. 346–353.

[152] L. Gao, J. Wu, Z. Qiao, C. Zhou, H. Yang, and Y. Hu, "Collaborative social group influence for event recommendation," in *Proc. CIKM*, 2016, pp. 1941–1944.

[153] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)," in *Proc. RecSys*, 2011, pp. 387–388.

[154] J. O. Garcia, A. Ashourvan, S. Muldoon, J. M. Vettel, and D. S. Bassett, "Applications of community detection techniques to brain graphs: Algorithmic considerations and implications for neural function," *Proc. IEEE*, vol. 106, no. 5, pp. 846–867, 2018.

[155] J. J. Bechtel, W. A. Kelley, T. A. Coons, M. G. Klein, D. D. Slagel, and T. L. Petty, "Lung cancer detection in patients with airflow obstruction identified in a primary care outpatient practice," *Chest*, vol. 127, no. 4, pp. 1140–1145, 2005.

[156] N. Haq and Z. J. Wang, "Community detection from genomic datasets across human cancers," in *Proc. GlobalSIP*, 2016, pp. 1147–1150.

[157] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. election: Divided they blog," in *Proc. KDD Workshop Link Discovery*, 2005, pp. 36–43.

[158] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. KDD*, 2008, pp. 990–998.

[159] C. Remy, B. Rym, and L. Matthieu, "Tracking bitcoin users activity using community detection on a network of weak signals," in *Proc. CNA*, 2017, pp. 166–177.

[160] Z. Chen, W. Hendrix, and N. F. Samatova, "Community-based anomaly detection in evolutionary networks," *J. Intell. Inf. Syst.*, vol. 39, no. 1, pp. 59–85, 2012.

[161] T. Waskiewicz, "Friend of a friend influence in terrorist social networks," in *Proc. ICAI*, 2012.

[162] V. Fionda and G. Pirrò, "From community detection to community deception," 2016, *arXiv:1609.00149*.

[163] Y. Liu, J. Liu, Z. Zhang, L. Zhu, and A. Li, "REM: From structural entropy to community structure deception," in *Proc. NIPS*, 2019.

[164] V. Fionda and G. Pirrò, "Community deception or: How to stop fearing community detection algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 4, pp. 660–673, 2018.

[165] J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu, "Most influential community search over large social networks," in *Proc. ICDE*, 2017, pp. 871–882.

[166] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proc. SIGMOD*, 2014, pp. 991–1002.
[167] X. Huang and L. V. S. Lakshmanan, "Attribute-driven community search," *Proc. VLDB Endow.*, vol. 10, no. 9, pp. 949–960, 2017.
[168] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. KDD*, vol. 2014, pp. 701–710.
[169] 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. CIKM*, 2017, pp. 377–386.
[170] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Physica A*, vol. 388, no. 8, pp. 1706–1712, 2009.
[171] T. Li *et al.*, "Hierarchical community detection by recursive partitioning," *J. Am. Stat. Assoc.*, pp. 1–18, 2020.
[172] Q. Long, Y. Wang, L. Du, G. Song, Y. Jin, and W. Lin, "Hierarchical community structure preserving network embedding: A subspace approach," in *Proc. CIKM*, 2019, pp. 409–418.
[173] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proc. KDD*, 2017, pp. 555–564.
[174] J. Wu, S. Pan, X. Zhu, C. Zhang, and S. Y. Philip, "Multiple structure-view learning for graph classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3236–3251, 2018.
[175] J. Kim and J.-G. Lee, 'Community detection in multi-layer graphs: A survey," *SIGMOD Rec.*, vol. 44, no. 3, pp. 37–48, 2015.
[176] Y. Cao, H. Peng, J. Wu, Y. Dou, J. Li, and P. S. Yu, "Knowledge-preserving incremental social event detection via heterogeneous GNNs," in *Proc. WWW*, 2021, pp. 3383–3395.
[177] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," in *Proc. NIPS*, 2009.
[178] X. Wang *et al.*, "Heterogeneous graph attention network," in *Proc. WWW*, 2019, pp. 2022–2032.
[179] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *Proc. NIPS*, 2020.
[180] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.
[181] S. Xue, J. Lu, and G. Zhang, "Cross-domain network representations," *Pattern Recognit.*, vol. 94, pp. 135–148, 2019.
[182] J. Wu, X. Zhu, C. Zhang, and Z. Cai, "Multi-instance multi-graph dual embedding learning," in *Proc. ICDM*, 2013, pp. 827–836.
[183] J. Wu, Z. Hong, S. Pan, X. Zhu, Z. Cai, and C. Zhang, "Multi-graph-view learning for graph classification," in *Proc. ICDM*, 2014, pp. 590–599.
[184] R. Li, C. Zhang, Q. Hu, P. Zhu, and Z. Wang, "Flexible multi-view representation learning for subspace clustering," in *Proc. IJCAI*, 2019, pp. 2916–2922.
[185] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on Epinions. com community," in *Proc. AAAI*, vol. 5, 2005, pp. 121–126.
[186] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: Mining a social network with negative edges," in *Proc. WWW*, 2009, pp. 741–750.
[187] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Governance in social media: A case study of the Wikipedia promotion process," in *Proc. ICWSM*, no. 1, 2010.
[188] P. Xu, W. Hu, J. Wu, and B. Du, "Link prediction with signed latent factors in signed social networks," in *Proc. KDD*, 2019, pp. 1046–1054.
[189] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI*, 2015, pp. 4292–4293.
[190] R. Mastrandrea, J. Fournet, and A. Barrat, "Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys," *PLoS ONE*, vol. 10, no. 9, 2015, Art. no. e0136497.
[191] A.-L. Barabási and E. Bonabeau, "Scale-free networks," *Sci. Amer.*, vol. 288, no. 5, pp. 60–69, 2003.

Xing Su received the M.Eng. degree in computer technology from Lanzhou University, Lanzhou, China, in 2020. She is currently pursuing the Ph.D. degree with the School of Computing, Macquarie University, Sydney, NSW, Australia.

Her current research interests include community detection, deep learning, and social network analysis.

Shan Xue received the Ph.D. degree in information management from the School of Management, Shanghai University, Shanghai, China, in 2018, and the Ph.D. degree in computer science from the Center for Artificial Intelligence, University of Technology Sydney, Ultimo, NSW, Australia, in 2019.

She is currently a Post-Doctoral Research Fellow with the Faculty of Science and Engineering, School of Computing, Macquarie University, Sydney, NSW, Australia, and also a Researcher with the CSIRO Data61, Sydney. Her current research interests include artificial intelligence, machine learning, and knowledge mining.

Fanzhen Liu received the M.Res. degree from Macquarie University, Sydney, NSW, Australia, where he is currently pursuing the Ph.D. degree in computer science.

His current research interests include graph mining, machine learning, and social network analysis.

Jia Wu (Senior Member, IEEE) received the Ph.D. degree from the University of Technology Sydney, Ultimo, NSW, Australia.

He is currently an ARC DECRA Fellow with the School of Computing, Macquarie University, Sydney, NSW, Australia. His current research interests include data mining and machine learning. He has authored or coauthored more than 100 refereed research papers in these areas, such as TPAMI, TKDE, TNNLS, TMM, NIPS, KDD, ICDM, IJCAI, AAAI, and WWW.

Dr. Wu was a recipient of the SDM'18 Best Paper Award in Data Science Track and the IJCNN'17 Best Student Paper Award. He currently serves as an Associate Editor for the *ACM Transactions on Knowledge Discovery from Data*.

Jian Yang (Member, IEEE) received the Ph.D. degree in data integration from The Australian National University, Canberra, ACT, Australia, in 1995.

She is currently a full Professor with the School of Computing, Macquarie University, Sydney, NSW, Australia. She has authored or coauthored more than 200 journal and conference articles in international journals and conferences, such as the IEEE TRANSACTIONS, INFORMATION SYSTEMS, DATA, AND KNOWLEDGE ENGINEERING, VLDB, ICDE, ICDM, and CIKM. Her research interests include business process management, data science, and social networks.

Dr. Yang is currently serving as an Executive Committee for the Computing Research and Education Association of Australasia.

Chuan Zhou received the Ph.D. degree from the Chinese Academy of Sciences, Beijing, China, in 2013.

He is currently an Associate Professor with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. He has authored or coauthored more than 80 articles, including TKDE, ICDM, AAAI, IJCAI, and WWW. His research interests include social network analysis and graph mining.

Dr. Zhou received the Outstanding Doctoral Dissertation Award from the Chinese Academy of Sciences in 2014, the Best Paper Award from ICCS in 2014, and the Best Student Paper Award from IJCNN'17.

Wenbin Hu is currently a Professor with the School of Computer Science, Wuhan University, Wuhan, China. He has more than 80 publications appeared in several top conferences, such as SIGKDD, IJCAI, AAAI, and ICDM, and journals, such as the IEEE TKDE, IEEE TMC, IEEE TITS, and the *ACM Transactions on Knowledge Discovery from Data*. His current research interests include intelligent and complex networks, data mining, and social networks.

Cecile Paris received the bachelor's degree from the University of California at Berkeley, Berkeley, CA, USA, and the Ph.D. degree from Columbia University, New York, NY, USA.

She is currently a Chief Research Scientist with the CSIRO Data61, Sydney, NSW, Australia. She is also an Honorary Professor with Macquarie University, Sydney. She is a Pioneer in natural language processing, user modeling, artificial intelligence, and human–machine communication. She has over 25 years of experience in research and research management and over 300 refereed publications.

Dr. Paris is a fellow of the Academy for Technology, Science and Engineering and the Royal Society of New South Wales.

Surya Nepal is with the CSIRO Data61, Sydney, NSW, Australia, since 2000, where he is currently a Senior Principal Research Scientist, the Deputy Research Director of Cybersecurity Cooperative Research Centre, and also leads the distributed systems security group comprising 30 staffs and 50 Ph.D. students at CSIRO. He also holds an honorary professor position at Macquarie University, Sydney, and a conjoint faculty position at the University of New South Wales, Sydney. He has more than 250 peer-reviewed publications to his credit. His research interests include distributed systems, with emphasis on security, privacy, and trust.

Dr. Nepal is a member of the Editorial Boards of the IEEE TSC, ACM TOIT, and IEEE DSC.

Di Jin received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 2012.

He was a Post-Doctoral Research Fellow with the School of Design, Engineering, and Computing, Bournemouth University, Poole, U.K., from 2013 to 2014. He is currently an Associate Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. He has authored or coauthored more than 50 articles in international journals and conferences in the areas of community detection, social network analysis, and machine learning.

Quan Z. Sheng (Member, IEEE) received the Ph.D. degree in computer science from the University of New South Wales, Sydney, NSW, Australia.

He is currently a full Professor and Head with the School of Computing, Macquarie University, Sydney. He has more than 400 publications. His research interests include big data analytics, service-oriented computing, and the Internet of Things.

Prof. Sheng was a recipient of the Microsoft Fellowship in 2003, the Chris Wallace Award for Outstanding Research Contribution in 2012, the ARC Future Fellowship in 2014, and the AMiner Most Influential Scholars Award in IoT in 2019. He is ranked by Microsoft Academic as one of the Most Impactful Authors in Services Computing (ranked top 5 all time).

Philip S. Yu (Fellow, IEEE) is currently a Distinguished Professor of computer science with the University of Illinois at Chicago, Chicago, IL, USA, where he also holds the Wexler Chair in information technology. He holds or has applied for more than 300 U.S. patents. He has authored or coauthored more than 990 articles in refereed journals and conferences. His research interest includes big data, including data mining, data stream, database, and privacy.

Prof. Yu is a fellow of the ACM. He has received several IBM honors, including two IBM outstanding innovation awards. He was the Editor-in-Chief of the IEEE TKDE from 2001 to 2004.