applied sciences

MDPI

Review

A Survey on the Recent Advances of Deep Community Detection

Stavros Souravlas [1,*,†], Sofia Anastasiadou [2] and Stefanos Katsavounis [3,†]

[1] Department of Applied Informatics, School of Information Sciences, University of Macedonia Thessaloniki, PC 54616 Thessaloniki, Greece
[2] Department of Midwifery, School of Health Sciences, University of Western Macedonia, PC 50200 Ptolemaida, Greece; sanastasiadou@uowm.gr
[3] Department of Production and Management Engineering, Democritus University of Thrace, PC 67100 Xanthi, Greece; skatsav@pme.duth.gr
* Correspondence: sourstav@uom.edu.gr or ssouravlas@uowm.gr
† These authors contributed equally to this work.

Abstract: In the first days of social networking, the typical view of a community was a set of user profiles of the same interests and likes, and this community kept enlarging by searching, proposing, and adding new members with the same characteristics that were likely to interfere with the existing members. Today, things have changed dramatically. Social networking platforms are not restricted to forming similar user profiles: The vast amounts of data produced every day have given opportunities to predict and suggest relationships, behaviors, and everyday activities like shopping, food, traveling destinations, etc. Every day, vast data amounts are generated by the famous social networks such as Facebook, Twitter, Instagram, and so on. For example, Facebook alone generates 4 petabytes of data per day. The analysis of such data is of high importance to many aspects like recommendation systems, businesses, health organizations, etc. The community detection problem received considerable attention a long time ago. Communities are represented as clusters of an entire network. Most of the community detection techniques are based on graph structures. In this paper, we present the recent advances of deep learning techniques for community detection. We describe the most recent strategies presented in this field, and we provide some general discussion and some future trends.

Keywords: deep learning; network analysis; community detection; social computing

Citation: Souravlas, S.; Anastasiadou, S.; Katsavounis, S. A Survey on the Recent Advances of Deep Community Detection. *Appl. Sci.* **2021**, *11*, 7179. https://doi.org/10.3390/app11167179

Academic Editors: Alessandro Di Nuovo and Eui-Nam Huh

Received: 16 May 2021
Accepted: 2 August 2021
Published: 4 August 2021

1. Introduction

Nowadays, more than ever before, the analysis of the vast data amounts produced in the social media is used in a variety of ways: recommendation engines, marketing, crime detection. Examples abound: Innovative companies like Netflix and Amazon have used predictive analytics for years as their basis to develop highly accurate recommendation systems (like the products recommended by Amazon or the movies and TV shows suggested specifically to each Netflix customer). The data collected by users as they browse Facebook or Instagram are used to match them with a large number of companies which offer products and services that, based on statistical models, they would probably be interested in. Facebook and Instagram maintain the biggest and most comprehensive databases of personal information. These databases are expanding rapidly every day.

The meaningfulness and usability of the data retrieved from the social networks highly depend on the existing relationships among the social media users. Apparently, people that follow the same groups are likely to be suggested similar products or services. In other words, the extraction of meaningful relationships among the billions of Facebook or Instagram users provides high value to many applications. The *community detection* paradigm mainly uses datasets that include the likes, the opinions, and the current relationships among social media users, in order to detect underlying clusters and to verify the accuracy of the already formed clusters within an entire network. In this regard,

community detection can facilitate the uncovering of hidden relationships between social network users.

Traditional methods of community detection are based on statistical inference, heuristic approaches, or conventional machine learning [1]. However, despite their past popularity, these strategies are not efficient in the modern era, where the datasets are larger, more complex, and the social relationships over the networks are becoming more complex and much more difficult to define and extract [2–7]. In conventional machine learning, community detection has been treated as a problem of forming, extracting, and verifying the accuracy of clusters on large graphs. These approaches attempt to generate a network embedding with few dimensions to reconstruct the original network. However, this type of representation to a low dimensional space is, in a broad sense, linear [8]. The fact that the real world networks include nonlinear structures makes the traditional strategies less useful. In a world where the scale of networks increases and the datasets become larger and larger, more robust and efficient techniques are required to achieve high performance [9,10]. Deep communities in a graph are connected components that can only be located after removing nodes and edges from the rest of the graph.

In this work, we are reviewing the most recent community detection approaches, which use deep learning strategies. The main contributions of our work are the following:

1. We describe and analyze the most prominent new strategies for community detection.
2. We organize the existing schemes in groups, depending on the technique being used.
3. We outline several points of interest based on the study of each family of community detection papers.
4. We present some possible future trends.

At this time, we must point out that community detection on big data networks is an ongoing field and the majority of the papers presented are quite recent. In this survey, we have selected to present quite recent papers, the majority of which have been published between 2019–2020. Still, one can expect more published papers on the topic. A quite recent review [1] in 2020 briefly discusses only a few of the papers which are presented in this survey, while only future trends are presented (there is no detailed commenting on the presented schemes). In another survey [11], the authors introduce a new taxonomy that divides the existing methods into two categories, namely probabilistic graphical models and deep learning.

The remainder of this work is organized as follows: Section 2 presents the preliminaries, a brief discussion on non-DL community detection schemes with emphasis on their categorization, and some definitions which are required for this study. Section 3 presents the three categories of DL community detection and a discussion on each category. Section 4 provides some future trends. Section 5 concludes this survey.

2. Preliminaries and Definitions Used in This Survey

It is commonly known that people's personal networks can be quite big, and their identification is very cumbersome. Generally, the idea to construct social networks is based on the assumption that a society's users will have some features in common. In this sense, researchers have developed a number of techniques that usually start from small predefined communities and implement some type of algorithm that (1) expands these communities, (2) detects newly formed communities, and (not in all cases) (3) finds overlaps between communities. For example, Figure 1 shows a network with three communities.

Let $G = (V, E)$ be a weighted, undirected graph, where V and E are the sets of nodes and edges, respectively. The nodes represent users, and edges represent the connections between two users. The *similarity* $w_{i,j}$ between users i and j is the weight of the edge that connects i and j. This value lies in the interval $[0 \dots 1]$ and indicates how similar are their interests in terms of the files they request through the network.

Regularly, the network nodes are also weighted: the weight of a node i indicates its *Network Connectivity Degree* (NCD), i.e., how well the preferences, likes, and views of a user are fitted to a community. The network connectivity degree is also between $[0 \dots 1]$.

The letters are the node names, the edge values indicate view similarities, and the node values indicate similarity degrees. The network connectivity degree NCD_i of the i user is computed as follows:

$$NCD_i = \frac{\sum w_{i,j}}{n} \quad (1)$$

where $w_{i,j}$ is the weight of any edge that connects user i with any user j that lies in the same community, and n is the number of such edges.

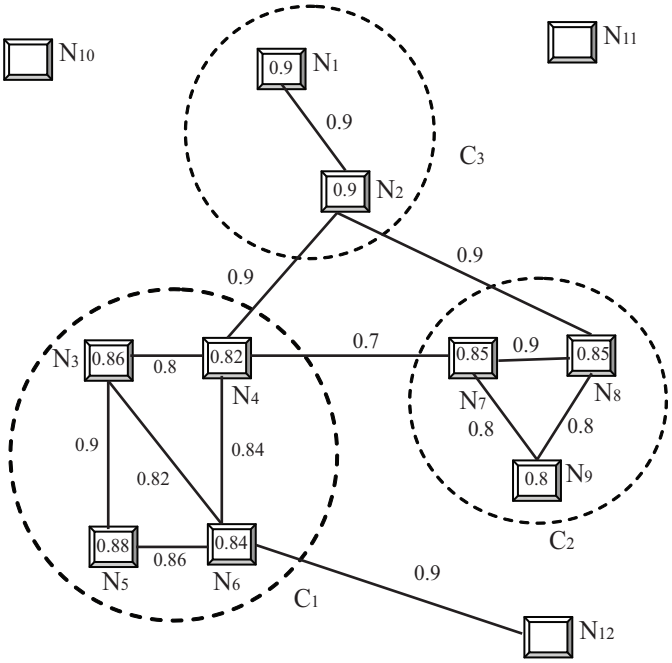

Figure 1. A network with three communities.

A community detection strategy is fed with a network structure as well as node and edge attributes. The network structure identifies the relationships between the nodes; specifically, the related nodes are linked and the weights over the links (edges) show the strength or importance of a connection. The attributes include the information for nodes, like profession, age, and groups of interest that one can find in a Facebook or Instagram profile. Sometimes, signs are also used to indicate positive or negative relationships. Datasets for a number of social networks, which can be used as inputs for a community detection problem, can be found in [12–14]. The output of a community detection strategy is a set of communities. If the strategy considers overlapping, then there are communities that have nodes in common. Otherwise, the communities are perfectly disjoint.

Before we present the categories of DL-based community detection schemes schemes, let us briefly describe how the non-DL community detection schemes can be grouped. There are three basic approaches for the non-deep community detection schemes: (a) top-down approaches (for example [15,16]), which start from the graph representing the entire network and try to divide it into communities, (b) the bottom-up approaches that use the local structures and try to expand them to form communities, and (c) the data-structure based approaches that try to convert the entire network into a data structure, which is then processed to detect communities.

The main advantage of the top-down approaches is that they can easily detect overlapping communities. However, when this overlapping is too high (a node is connected to large number of links distributed to many different communities), there are very large processing delays.

The majority of papers follow the second approach, bottom-up (for example [17,18]). This approach starts from local structures and expands to the overall network. During this

process, various communities are formed. The main advantage of the bottom-up approach is the linear time complexity in most of the cases. However, strategies that belong to these sub-categories often fail to detect very small communities, even in cases they are well-defined. This happens because the initial local structures do not capture these small communities from scratch, and the expansion method used fails to incorporate node-members of a community. Finally, the data structure-based approach is based on the idea of forming a network to some type of data structure (usually in a tree form), which is then analyzed in a way to detect communities (examples are [9,10]. Generally, the idea of converting a network to a data structure is an interesting one; however, this conversion should be implemented carefully, as it may be very expensive in terms of computation costs, especially when processing networks of millions nodes or edges. Careful parallelism can partially solve this issue.

Some definitions required are given below:

1. **Overlapping Communities:** Overlapping communities are communities that have nodes in common.
2. **Normalized-Cut:** A measure that computes the cut cost as a fraction of the total edge connections to all the nodes in the graph.
3. **Seed nodes:** Key nodes of high influence over the network.
4. **Graph embedding:** A process that learns a mapping from a network to a vector space, while preserving relevant network properties.
5. **Node embedding:** The process of mapping each node into a low-dimensional space, to get insight of its similarity and network structure.
6. **Random walks:** Given network $G(V, E)$, the random walk is a process used to obtain random sequences of vertices rooted on each vertex $v \in V$. The walks are randomly sampled starting from the neighbors of the last visited vertex.
7. **Normalized Mutual Information (NMI)**: a measure used to evaluate network partitioning performed by community detection schemes.
8. **Modularity:** A measure of the network structure for the strength of division of a network into clusters or communities.

3. Deep Community Detection Strategies

Due to the variety of information, the different network topologies, which are normally irregular, and the different node characteristics, different types of models for community detection have been proposed. Complex modern networks are represented by graphs. Graph clustering aims at partitioning graphs into several densely connected disjoint communities [19,20]. The deep community detection strategies can be divided into three big families: (i) AE-Based Community Detection (Auto-encoders are used), (ii) CNN/GNN-Based Community Detection (Convolutional Neural Networks (CNNs) and their variant Graph Neural Networks (GNNs) are used), and (iii) GAN-Based Community Detection (Generative Adversarial Networks (GANs) are used). In the remaining of this section, we describe the more recent, state-of-the-art papers that fall in these categories.

3.1. AE-Based Community Detection Strategies

Autoencoders (AEs) are very common in community detection because of their ability to to represent efficiently nonlinear real-world networks. The deep neural network based auto-encoders are used to learn data codings (representations of data sets) in an unsupervised manner. They are a tool used for dimensionality reduction. They manage to implement a proper pre-processing of the adjacency matrix, in order to improve the spatial proximity [21–25]. The embeddings provided have nonlinear properties and thus the auto-encoder is suitable for mapping data points into the low-dimensional space, as required in community detection.

In [26], a new approach to combine the models of modularity and normalized-cut via an auto-encoder was proposed. The modularity approach aims at maximizing the *modularity measure*, that is, the number of edges in the original graph divided by the

expected number of edges in a random graph that has no specific community structure. In the normalized-cut approach, the aim is to cut the minimum number of edges and have the vertices of the same partition possess a high similarity and the vertices between different partitions possess low similarity. A unified objective function is used to reconstruct the combination matrix composed of the modularity and the Markov matrix. The auto-encoder is used to produce a mapping to the low-dimensional space, which can reconstruct the joint matrix.

A multi-layer auto-encoder is introduced by cascading a number of auto-encoders. In this regard, the network is trained on a "per layer" basis. The multi-layer autoencoder provides powerful encoding with both topological and content information, which is very efficient for network community detection. Experiments were conducted using the NMI as a comparison metric. The experimental results have shown that this combination indeed improves community detection, compared to approaches that use topological or content information alone. The computational time complexity of the proposed scheme is $O(N)$, which is linear to the number of graph nodes.

In [27], DeCom, a model is proposed that leverages the idea of autoencoder pipelines used to extract overlapping communities in large-scale networks. The DeCom strategy is composed of three phases: The first phase is the seed selection phase, where the features of the network's vertices are learned in a stepwise greedy approach. The auto-encoder tries to learn the values associated with the vertices by mapping the data degree matrix to a matrix with fewer dimensions. In the second phase, the topology is learned; the goal here is to get multiple communities in a completely unsupervised manner, where each community is defined by the seed. The communities are expanded from each seed using a random walk with the seed as the starting point. Thus, each community includes the seed and nodes closed to the seed (thus, highly influential). By multiple iterations, the influence keeps spreading linearly, with label information, to the neighboring nodes. In the third phase, the formed clusters are refined using modularity optimization, that is, by finding better representations by reconstruction of the network structure. Compared to similar works [28–31], the DeCom strategy updates the values of vertices in multiple iterations to minimize the reconstruction error. This results in better graph coverage (% of vertices assigned to clusters). In addition, compared to well known LPA (label propagation algorithms) like COPRA, this reconstruction minimization error recovers the vertices which are lost in LPAs due to not obtaining a label. The cost of initializing n vertices in an l-layered auto-encodes is $O(l(n))$. The seed selection time is linear to the number of edges z, $O(z)$, and it takes $O(n\log(n))$ to sort the degrees in ascending order of their influence. The formation of k communities with n' nodes is $O(n+z)$ and the modularity optimization $O(k\log(n'))$ for modularity optimization.

Different from other AE schemes, Cavallari et al. [32] proposed ComE, a Community Embedding framework that jointly solves community embedding, community detection, and node embedding and suggests that community embedding is not only useful for community-level applications such as graph visualization, but it can support community detection and node classification. The community embedding problem is treated as a multivariate Gaussian distribution, and it is employed to reinforce the community detection. The Gaussian mixture formulation can not only detect the communities, but it can also determine their embedding distributions. For a community with its embedding expressed as a multivariate distribution in the low dimensional space, the member nodes can be forced to scatter near the embeddings mean vector. Such nodes have similar node embedding vectors without necessarily being directly linked. Thus, node embedding produces better inputs for the community detection that follows. The community detection time complexity for ComE is $O(|V|+|E|)$, where V, E are the node and edge numbers.

The NetRA model [33] is another network embedding strategy. Here, a generative adversarial training process is employed as a complementary regularizer. The advantages of this process are as follows: (1) The regularizer is helpful in detecting useful information about the data, and (2) the selected training implementation achieves a robust discrete-

space learning that overcomes the possible over-fitting on walks with sparse samples. The NetRa approach includes a random walk generator and the network embedding model with adversarially regularized auto-encoders. The random walks are sequences of nodes, and they are sampled in random by using the neighboring nodes of the last visited vertex. For training, after the random walks are executed, they pass through encoding layers to get the vector representations of vertices. These vector representations are then transformed back into n dimensions. The generator output and the encoder latent representation are used as inputs to the discriminator to obtain the adversarial loss. Moreover, the generator, through a training phase, transforms the Gaussian noise into the latent space, with a good approximation to the true data. After training, the network vertex representations are obtained by inputing the random walks to the encoder function. On a network with n nodes and m edges, the NetRa time complexity for learning is $O(n)$ and a computational cost of $O(n^2)$ is added for embedding. To the best of our knowledge, a few recent works integrate community embedding in deep learning. Node embedding methods alone may preserve the directly connected nodes or nodes with similar embedding vectors, but, in order to have knowledge of the community structure, a community detection scheme is also required. This combination produces larger costs, so more work needs to be done in this area.

Xu et al. [34] presented the Community Detection Method via Ensemble Clustering (CDMEC). This framework combines transfer learning and a stacked auto-encoder to produce a feature representation of complex networks in low dimension. In addition, it aggregated multiple inputs via an ensemble clustering framework. This framework initially uses the results produced in the basic clustering in order to construct a consistent matrix, and then it uses the non-negative matrix factorization clustering to detect clusters resulting from the constructed consistent matrix. The time complexity is $O(k \times L)$, where k and L are the number of iterations and layers of the autoencoder, respectively. The main advantage of the CDMEC is that the combination of the deep stacked autoencoder and transfer learning can result in obtaining a low-dimensional feature representation. Compared to strategies which are based on capturing the geometric structure of data using group distances [35], the CDMEC scheme manages to get low-dimensional representations in a more efficient way. Moreover, to reveal more comprehensive similarity relationships between the nodes of a network topology, the CDMEC introduced similarity matrices based to different functions.

The Variational Graph Autoencoder (VGAE) [36–38] is an interesting method that captures the high-order features of a community structure. The VGAE has allowed models to achieve state-of-the-art performance for node clustering. In [36], a dual optimization procedure is introduced to guide the optimization process and encourage learning of the primary objective. The main objective here is to deal with the problem of extracting non-optimal communities, as a result of the loss optimization efforts through the stochastic gradient descent [37]. The proposed encoder is linearized and, in this way, the learning parameters are reduced. The time complexity of the proposed scheme is dictated by $O(N^2)$, where N is the number of network nodes.

A number of approaches focus on reducing and sharing the trainable parameters, in order to improve the efficiency of deep learning models. Several approaches can be mentioned here: Dai et al. [39] developed a method of trainable parameters reduction by sharing these trainable parameters. This method overcomes the timing and resource limitations and provides a good accuracy. In addition, Aich et al. [40] proposed a method for reducing the trainable parameters by employing weight sharing over multiple scales in convolutional networks. In a more recent but different autoencoder based approach [41], the DACDRP (Deep Autoencoder-based Community Detection using the Reduction of trainable parameters by complex network Partitioning) strategy was introduced. The method was built based on partitioning complex networks into chunks, in order to reduce the necessary training parameters of the deep learning model. These chunks shared the same training parameters. The method ensures high connectivity among these chunks and introduces a new

similarity constraint function that improves and the overall effectiveness. The DACDRP strategy was implemented on different levels of partitioning: DACDRP-L0, DACDRP-L1, and DACDRP-L2. An extra add-on of the proposed scheme is that it is fully parallelizable and thus it guarantees higher efficiency. Both CPU and GPU devices have been used for the implementation. Generally, for complex networks of size n, the model time complexity is of order of n^2, which is reduced by network splitting and parallelism.

The Adaptive Graph Encoder (AGE) is another filter-related graph embedding model for community detection. Cui et al. [42] proposed a two-part model for auto-encoder based community detection. The model consists of a Laplacian smoothing filter and an adaptive encoder. The smoothing filter is playing the role of a low-pass filter to de-noise the components of the feature matrix X, which have very high frequencies. The smoothness of a graph signal $\mathbf{x}$ is calculated by the Rayleigh quotient over the graph Laplacian L and $\mathbf{x}$. This matrix is the input of the adaptive encoder, which is trained via a supervised method.

Table 1 summarizes the auto-encoder based techniques presented in this section, in terms of the year of publication, the specific strategy being used for community, the overall time complexity whenever available, and the main experimental results found.

Table 1. Auto-encoder based community detection strategies.

Paper	Year	Strategy	Complexity	Basic Experimental Results
[26]	2018	Combines the models of modularity and normalized-cut via an auto-encoder	$O(N)$	Outperforms classic community detection schemes like the Louvian [43] or CNM [44] method or similar approaches that use only content or topological information.
[27]	2019	Two-phased approach; (1) the auto-encoder layered approach is used to initialize candidate seed nodes, (2) Refinement of the seed nodes and clusters in the last encoder layer.	$O(n\log(n))$	The selection of seed nodes reduces the number of iterations. The strategy scales up linearly produces better quality of clusters.
[32]	2017	Community Embedding framework that jointly solves community embedding, community detection and node embedding.	$O(\|V\|+\|E\|)$	Tested on real world datasets, it improves graph visualization and outperforms other schemes in application tasks, like community detection and node classification.
[33]	2018	Generative adversarial training process as a complementary regularizer	$O(n)$	Effectiveness on a variety of tasks, including network reconstruction, link prediction, and multi-label classification.
[34]	2020	Combines transfer learning and a stacked auto-encoder to produce a feature representation of complex networks in low dimension.	$O(k \times L)$	Experiments on artificial and real networks show that it is superior to the existing methods and has great potential in solving the community detection problems
[36]	2020	A dual optimization procedure, which aims at guiding the optimization process.	$O(N^2)$	The performance gain is marginal as far as community detection is concerned, and the number of learnable parameters, is reduced; thus, faster convergence and training speed are achieved.
[41]	2020	Based on partitioning complex networks and on techniques for reduction and sharing of trainable parameters.	Not inferred	Improves training speed and efficiency, which increases in deeper level of partitioning.
[42]	2020	A Laplacian smoothing filter combined to an adaptive encoder.	Not inferred	Experiments on four datasets have shown that the scheme outperforms other graph embedding methods considerably on node clustering and link prediction tasks.

Comments on the AE Based Community Detection Schemes

After the study of the most recent AE-based community detection methods, we made the following observations:

1. **The two kinds of information considered in community detection produce limited and inefficient models. Auto-encoders provide a solution:** The topology and node content information can be considered as objective functions to be combined in a linear form. However, this linearity is not certain in real world networks, and this makes the models limited. In addition, we can produce a network representation by combining the two types of information but, the optimization decision on the appropriate ration of the two types of information makes the model inefficient. Finally, the high-dimensional feature space of the input data can lead to the production of very complex neural network architectures and can increase the number of trainable parameters, which may reach millions. The auto-encoder is a good method to overcome the efficiency problems of linear optimization and to combine the two types of information.
2. **Auto-encoders are the dominant method for mapping data points into the lower dimensional spaces:** The disadvantage of the network high-dimensionality is the very large computational costs that the community detection methods suffer. Thus, a method of converting high-dimensional graphs into a space with fewer dimensions, in which the important information regarding the network structure and the node features are still maintained, is required. The graphs are generally represented by adjacency matrices, which store the node connection information but not the proximity information for non-directly connected nodes. The dominant solution to this accuracy problem is the auto-encoder (AE).
3. **Sparsity of network samples:** Network sampling techniques like random walk sampling can be used to to derive vertex sequences as training datasets. However, the sampled data represent only a small portion of all the vertex sequences. Alternatively, these structures can be encoded in a continuous code space. The idea is to adaptively find the optimal representation by adding a sparse constraint to the auto-encoder for this purpose (sparse auto-encoders). An example is the work of Fei et al. [45], which employs a similarity matrix to reveal the indirect connections between nodes and a sparse auto-encoder to lower the dimensions and extract the complex network structures.
4. **Community Overlapping:** The majority of the strategies described do not deal with overlapped communities. This can be a subject of future concern. An example of auto-encoder based community detection scheme is [27].
5. **Complexity:** The high dimensionality of the feature space of the input data normally increases the number of trainable parameters and makes the models rather complex. Most of the schemes achieve linear time complexity by employing division policies over the network and/or data space to efficiently handle the large data amounts (for examples, see [26,41]).

3.2. CNN and GNN Based Approaches

The Convolutional Neural Networks (CNNs) and their extensions, the Graph Neural Networks (GNNs), are also used in community detection, and they are a powerful tool used to extract the spatial localization. CNNs can only operate on regular Euclidean structured data like images (2D grids) while their extension, GNNs, can be used in cases where the numbers of nodes connections vary [46], and the nodes are not ordered (irregular on non-Euclidean structured data). Thus, the latter approach seems to be more favorable recently. The majority of CNN/GNN community detection methods are semi-supervised or supervised. However, there are approaches that combine them with auto-encoders, to implement unsupervised approaches ([24] is an example). In the remainder of this subsection, we present the most recent CNN and GNN approaches to community detection.

3.2.1. CNN-Based Community Detection

The CNNs are an attractive solution for community detection because they combine the convolution application to pooling. A pooling layer is a layer added after the convolutional layer and its objective is to reduce the data dimensionality. A quite recent idea for

CNN based models is the introduction of Markov Random Fields (MRF). MRF have been used in other applications like image processing. A first attempt was made in [47], where a network MRF (NetMRF) model was presented, to encode the modular properties of an irregular network in the energy function to describe communities. The MRF was converted to a convolutional layer, which was incorporated into CNN. The drawback of NetMRF is that it does not consider information on nodes and requires a substantial amount of computation for learning the model. On the contrary, the MRFasGCN [48] exploits both network topology and node semantic information in an end-to-end deep network architecture. The approach is based on adding an MRF model as a new convolutional layer: the MRF model and its inference is first transformed to a convolutional layer and then added as the third (and last) layer of the basic two-layer convolutional network architecture. The third convolutional layer refines the result by performing the MRF. The final output is obtained by subtracting the refined result from first two layers. The total computational time complexity for all the layers is $O(emhk^2)$, where e, m, and k are the numbers of edges, attributes, and communities, respectively, while h is the number of hidden units of the first layer. The above methods are semi-supervised. He et al. [49] extended the idea of combining Markov Random Fields (MRF) with CCN [48] and developed an approach for unsupervised community detection, by adding an auto-encoder. The MRFasGCN is cast as an encoder and the node community membership is found in the hidden layer of the encoder. A dual decoder is used to separately reconstruct the network structures and node attributes in an unsupervised fashion to obtain reliable community detection.

Two new approaches can be found in [50,51]. Sperli's approach [50] is based on deep learning approaches and on the topological properties of social networks. Specifically, it leverages a particular CNN that has been modified so that it can deal with large dimensions and high sparsity of adjacency matrices. Two convolutional steps are used, and each one is composed of a convolution and a max-pooling layer for topology learning of the input network, plus a full connection layer. Because the pooling layer organizes the input in some defined areas and stores an aggregate value for each area, the computational time complexity is reduced. In [51], a scheme based on learning ground-truth communities for community detection over large networks is introduced. The proposed scheme uses an edge-to-image (E2I) model that can map the edge network structure to an image structure. Two type of edges are defined and classified: edges of the same community and edges that lie in different communities. This classification facilitates breadth search over the network to get localized community images.

Generally, the CNN and its variants have been widely adopted in network embedding [52–55]. For example, Xu et al. [52] re-formatted the complex network topology adjacent matrix into an image and designed a CNN model to extract and classify relevant features. Hanocka et al. used MeshCNN to analyze 3D shapes directly, in order to leverage their intrinsic connections. Inspired by these works, Wu et al. [24] introduced a CNN + AE (Convolutional Neuran Network + Autoencoder) approach to implement unsupervised community detection. The introduction of an autoencoder guarantees of a more efficient extraction of the spatial features of social networks and improves the community detection. In addition, the combined model can become the basis for dynamic community detection.

3.2.2. GNN-Based Community Detection

GNNs technically combine graph mining and deep learning and thus they are capable of modeling and detecting the underlying relationships in graph-based data. An interesting approach of GNN based schemes employ attention networks to capture the importance of the neighboring nodes to a target node. A representative work is the DAEGC algorithm (Deep Attentional Embedded Graph Clustering) [56], which develops a graph attentional autoencoder to learn latent representation and rank the importance of attributed nodes within a neighbor. More specifically, DAEGC uses attributed graphs to explore the information found in graphs. The framework consists of two parts: (1) an attention network auto-encoder that captures the affect of the neighboring nodes to a specific (target) node

and, in this way, it compactly encodes the topology structure of a network, and (2) a self training clustering module that performs clustering based on the learned representation, and, in return, manipulates the latent representation according to the current clustering result. A deep architecture is built by stacked layers of encoders.

Shchur and Günnemann [57] addressed the issue of overlapping communities, that is, communities with members in common. They introduce a GNN based model for overlapping community detection and four datasets for overlapping community detection, which are used as benchmarks. They combine the GNNs with the Bernoulli–Poisson model that is able to produce a variety of community topologies (e.g., nested, hierarchical), and lead to dense overlaps between communities. The computational time complexity is $O(N + M)$, where N is the number of nodes and M is the number of edges of the network. In addition, the authors pre-processed four real-world datasets, to satisfy the criterion of possessing reliable information on node attributes and ground-truth overlapping communities: Chemistry, Computer Science, Medicine, and Engineering (networks, constructed from the Microsoft Academic Graph). A two level GNN model was used, with batch normalization after the first graph convolution layer. In addition, in this model, the weight matrices undergo a regularization process.

Levie et al. [58] introduced a spectral domain convolutional architecture for deep learning on graphs. The basis of the model is a class of parametric rational complex functions (Cayley polynomials), which efficiently compute spectral filters on graphs. The main advantage of these filters compared to other schemes that use such filters [59] is their ability to detect narrow frequency bands of importance during training. The model produces localized in space filters, and its time complexity is linear to the input data for sparsely connected graphs. The Cayley polynomials are real-valued functions with one real, some complex coefficients, and a spectral zoom parameter which can be tuned to zoom in to different parts of the spectrum. In this way, different frequency band filters can be obtained. The Cayley filters use basic matrix operations such as powers, additions, multiplications by scalars, and inversions, so the application of the filter does not require expensive eigendecomposition of the Laplacian operator. For a sparse graph with n edges, the application of a Cayley filter on a signal requires $O((K + 1)rn)$ operations, where K is the number of times that vertices exchange information with their neighbors, and r is the number of filter coefficients.

In [60], a strategy that incorporates an AE to a GNN is proposed. A delivery operator is used to transfer the representations learned by auto-encoder to the corresponding layer. In addition, a dual self-supervised mechanism is used as guidance for the whole model updates. The proposed model is composed of a K-Nearest Neighbor (KNN) graph, a Deep Neural Network and a GNN network layer, and a dual self-supervised model. For a KNN network with N samples and a dimension of d, the proposed model finds, per sample, the top K similar neighbors and set edges to connect it these neighbors. The KNN graph is the input to the auto-encoder and the GNN. Each layer of the autoencoder is connected to the corresponding layer of GNN, so that the auto-encoder based representation can be integrated into the structure-aware representation. The overall time complexity of the proposed scheme is $O(Nd^2d_1^2 \ldots ...d_L^2 + |E||\mathcal{E}|dd_1 \ldots d_L + NK + N \log N)$, which is linear to the number of nodes. In this quantity, $\mathcal{E}$ is the number of edges, and d is the number of samples.

The Multi-View Attribute Graph Convolution Networks (MAGCN) [61] is based on two-pathway encoders that map graph embedding features and learn view-consistency information. The first pathway (Multi-view attribute graph convolution encoders) implements graph attention networks for multiple features, in order to reduce redundancies and learn the embedding features of the multiple view data. The second pathway (Consistent embedding encoders) further implements embedding encoders used to capture the geometry relationships and consistency of probability distribution among the information views. In this manner, the multi-view attributes are located in a clustered embedded space. Generally, the multi-view GNNs have the following limitations: (1) They cannot allocate

learnable different weights to different nodes in neighborhood, and (2) The similarity is not explicitly considered for the consistency relationship among different views [62,63]. The multi-view attribute graph convolution encoder equipped with attention mechanism for learning graph embedding from multi-view graph data provides the solution for learnable weight allocation to different nodes. In addition, the embedding encoders incorporate the geometric relationship and the probability distribution consistency among multi-view graph data, thus facilitating the clustering task.

The Dual-Regularized Graph Convolutional Networks (DRGCN) [64] is the first work that studies the node-level class-imbalanced graph embedding problem using GNNs. The DRGCN strategy has been developed to handle multi-class imbalanced graphs. It imposes two regulation types to handle the representation learning (Class-Imbalanced Convolution Learning). Two more ideas are implemented: (a) a class-conditioned training process, which facilitates separating the labeled nodes, and (b) a process that makes the unlabeled nodes follow the same latent distribution to the labeled nodes. More specifically, the Class-Imbalanced Convolution Learning is a two-layer graph convolutional network, which is used for learning the node-level representation on the input graph. The two-order relationships between neighboring nodes are modeled in a sequential manner. The Class-Conditioned Adversarial Regularization is used to enhance the separation of different classes by imposing conditional training on all labeled nodes. Finally, the Latent Distribution Alignment Regularization is a a distribution alignment training of both labeled and unlabeled node representations under the assumption that the imbalanced classes in the unlabeled space will match the imbalanced classes in the labeled space.

3.2.3. Comments on the GNN/CNN Based Community Detection Schemes

After the study of the most recent CNN/GNN-based community detection methods, we made the following observations:

1. **The CNN/GNN approaches consider both data structure and node features:** The recent deep clustering strategies combine feature extraction and dimensionality reduction in a model which enables deep neural networks to learn suitable representations based on the criteria of the clustering module being used. These strategies usually focus their attention on the data features and not on the structure of the data taken into account during representation learning. This structure reveals the similarity among samples, and therefore is rather useful on representation learning. The emerging CNN/GNN approaches encode the graph structure and node features for the representation. The structural information is very important in data representation learning.
2. **The CNN/GNN based community detection strategies are basically semi-supervised or supervised learning strategies:** Typically, the real networks are unique in the sense that the training data of one network cannot be properly used in a network with a different structure. When communities are located, the only data that can be used for analysis is the same network's data. Schemes that apply semi-supervised learning strategies can be employed so that a network's node label information can be used to predict community features for the remaining non-labeled nodes of the same network. These strategies are sometimes very expensive. In this regard, there are CNN/GNN strategies that exploit the strength of auto-encoders to develop automatic feature learning [24,49].
3. **Many real-world networks are characterized, at a large scale, by skewed class distributions:** This is because different network parts may evolve without restrictions. For example, different nodes may be topologically connected with multiple other nodes, so the node class assignment largely depends on its connected nodes. Representation learning through accurately identified boundaries for each class cannot be easily achieved because each class is highly affected by the neighboring classes in the graph. The problem of imbalanced class distributions is hardly addressed in most of the papers, which assume or purposefully provide balanced label distributions for each class. A solution is given in [22], where a two-layered graph network is used to extract node representations trained on imbalanced class labels.

4. **Performance degrades as graph convolutional layers are added:** As the depth of the convolutional network is increased, performance declines, so does the quality of the neighbor propagation. This is due to the fact that the addition of many convolutional layers is similar to multiple Laplacian smoothing procedures. This in fact smooths the node features and the nodes from different clusters become indistinguishable (in the sense that their attribute values converge and they become equal). The proposed strategies try to keep the number of layers to relatively small numbers (up to three, some examples are [48,49,57,60]). The ladder-shaped architecture proposed by [65] et al. is another idea to overcome this issue.
5. **Real-world graphs are formed by a combination of too many latent factors (for example, the same likes, hobbies, professions, etc.)**. Removing these factors from a graph is a difficult procedure: to learn the node representations free from such factors may need extraction of subsets of a neighborhood, in a quite complex network. Moreover, there are factors that may connect two people which are somehow overlapping (a football fan and a fan of a specific football club). Without completely distinguishing them (or making them independent), redundant representations will appear. Therefore, special attention should be given to implement techniques that support representation learning that discovers independent latent factors in a graph network. Ref. [66] is such an example.
6. **Convolutional networks are integrated with undirected graph models:** Such graphical models are the Markov Random Fields [48,49]. The convolutional networks generally do not consider community properties and thus the embeddings they construct are not community oriented. On the other hand, the undirected graph models, do not take into account the node information and the model learning requires complex computations. In this regard, the convolutional networks and graphical models like MRF are complementary and form a good combination for CNN/GNN based community detection.
7. **Overlapping and complexity analysis:** Most of the strategies do not consider the fact that communities may overlap; an exception is [57]. Moreover, many strategies do not provide comprehensive complexity analysis, as can be seen in Table 2. For those who do, we can observe linear complexity in terms of the number of nodes, edges, and samples being used.

3.3. GAN Based Approaches

Typically, the generative adversarial networks (GANs) are employing two competing deep neural networks: a generator and a discriminator. The latter discriminates if an input sample comes from the prior data distribution or from the generator and the former is trained to generate the samples in such a way that the discriminator is convinced that the samples come from a prior data distribution [67]. The result of this scheme is high training precision. Recently, the GANs have been used to community detection problems and have presented high efficiency.

In [67], the authors presented an adversarially regularized framework for graph embedding. The proposed strategy used the graph convolutional network as an encoder, allowing the embedding of the topology and node information into a vector representation. This representation is used to reconstruct the input graph. Two variants of the adversarial models were derived, the adversarially regularized graph auto-encoder (ARGA), as well as the adversarially regularized variational graph auto-encoder (ARVGA), to efficiently learn the graph embedding. The architecture of the ARGA consists of (i) a graph convolutional auto-encoder, which reconstructs a given graph from a latent representation. The latent representation has been produced by the encoder that is input with the graph structure (adjacency matrix) and the node content matrix, and (ii) an adversarial network which is trained to discriminate if a sample is generated from the embedding or from a prior distribution (regularization). The ARVGA architecture is the same as ARGA, but it employs a variational graph autoencoder instead of a graph convolutional auto-encoder (in part i).

The convolution time complexity is $O(|E|md)$, where E represents an encoding of edges between the nodes, d is the dimension of the latent variable, and m is the sample entities from the latent matrix or prior distribution.

Table 2. CNN/GNN based community detection strategies.

Paper	Year	Strategy	Complexity	Basic Experimental Results
[48]	2019	**CNN-Based:** The MRF model and its inference are transformed to a convolutional layer, and it is added as the third (and last) layer of the basic two-layer convolutional network architecture.	$O(emhk^2)$	Superior performance over state-of-the-art methods in terms of accuracy, normalized mutual information (NMI) and runtime—in addition, high scalability on several large benchmark problems.
[49]	2021	**CNN-Based with AE:** MRFasGCN [48] is cast as an encoder and then the node community membership is determined in the hidden layer of the encoder.	Not inferred	Improved performance in terms of accuracy, normalized mutual information (NMI) and runtime—in addition, high scalability on several large benchmark problems.
[50]	2019	**CNN-Based:** Uses two convolutional steps, each composed by a convolution and a max-pooling layer for topology learning of the input network, plus a full connection layer to refine the results.	Not inferred	Improved efficiency of execution time (in terms of the density variation of sparse matrix), especially for larger node numbers and improved accuracy of predictions.
[51]	2020	**CNN-Based:** Uses an edge-to-image model that can map the edge network structure to an image structure.	Not inferred	Increases the accuracy of community structure evaluation for both computer-generated networks and large-scale real-world networks.
[24]	2020	**CNN-Based with AE:** Based on three procedures, namely a matrix reconstruction method, a spatial feature extraction method (via an auto-encoder) and a community detection strategy.	Not inferred	Offers higher modularity thus it can efficiently detect high quality communities in social networks.
[56]	2019	**GNN-Based:** Uses an attention network to capture the importance of the neighboring nodes to a target node, to achieve a compact representation, on which a decoder is trained to reconstruct the graph structure.	Not inferred	Better performance compared to other strategies in terms of Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI).
[57]	2019	**GNN-Based:** Overlapping community detection, combines GNN with Bernoulli–Poisson model.	$O(M+N)$	It better recovers communities in graphs with known ground-truth communities, compared to other state-of-the-art schemes.
[58]	2019	**GNN-Based:** A spectral graph GNN architecture, based on complex rational Cayley filters, which can detect narrow frequency bands of importance during training.	$O[(K+1)rn]$	Experiments have shown improved performance compared to spectral domain convolutional approaches in terms, of image classification, accuracy, vertex classification, and matrix completion tasks.
[60]	2020	**GNN-Based:** a delivery operator to transfer the representations learned by auto-encoder to the corresponding layer, combined with a dual self-supervised mechanism to use as guidance for the whole model updates.	Linear to the number of nodes N and the number of samples, d	Better clustering accuracy and better clustering quality compared to other state-of-the-art schemes
[61]	2020	**GNN-Based:** Based on two-pathway encoders that map graph embedding features and learn view-consistency information.	Not inferred	Better performance compared to other strategies in terms of Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI).
[64]	2020	Method that imposes two regulation types to handle the representation learning, a class-conditioned training process, and a process that makes the unlabeled nodes follow the same latent distribution to the labeled nodes.	Not inferred	Effectiveness in handling graph data with not balanced distributions of classes.

CommunityGAN [68] is a community detection scheme that jointly solves overlapping community detection and graph representation learning. The CommunityGAN strategy learns network embeddings like AGM (Affiliation Graph Model) through a specifically designed GAN. The affiliation graph model is a framework that can model dense overlapped communities. As in [67], two models are trained: a generator that tries to produce a vertex subset to compose a motif (clique) and a discriminator which tries to deduce if the vertex subset is a real or unreal motif. The two models are combined by participating in a game where the generator tries to fool the discriminator by approximating a conditional probability value (the preference distribution of motifs covering an edge over all the remaining motifs), in order to produce the vertex subsets which are closer to the real motifs covering a specific edge. Accordingly, the discriminator distinguishes the ground-truth motifs from the ones produced by the generator. Given the distribution samples and the samples produced by the generator, the discriminator tries to maximize the probability of correct classification of these samples. The AGM is a non-negative affiliation weight matrix: when a vertex is assigned to a community, there is a non-negative strength value for this assignment. The big strength of CommunityGan is that, by combining the AGM with GAN, it can detect overlapped communities *and* learn graph representations. Because random walk is involved in the design of the AGM, the overall time complexity is not easy to infer. Although the model is not the fastest, the authors claim that its training is considered acceptable. Another approach for overlapped communities is the Seed Expansion with generative Adversarial Learning (SEAL) [69]. It contains an adversarial network, where the discriminator predicts if a community is real or unreal while the the generator tries to fool the discriminator by generating communities that fit the characteristics of real ones. The generator models seed expansion as a sequential decision process and learns heuristics from data. Another novelty is the use of a locator that forms a closed loop with the generator to get iterative improvements. If a generated community includes irrelevant nodes (free-rider effect), the locator would have difficulties in locating the seed node and returns a low reward, while, for a good community, the generator would receive a high reward as the seed node is easily located.

Wang et al. [70] provided GraphGAN, which is also a graph representation learning framework where the generator and the discriminator play a minimax game, and this competition leads to an iterative performance improvement. Moreover, the limitations of the softmax function (does not consider graph structure and proximity information and its computation involves all the graph vertices, which is inefficient and time consuming) are overcome by a new implementation of the generator, the Graph softmax, which defines a new method of computing connectivity distribution. This method satisfies the normalization, graph structure awareness, and computational efficiency. The proposed online generating strategy, which is more computationally efficient and consistent to the definition of graph softmax, employs a random walk from a root node, during which, if the generator decides to visit the currently visited node's v parent (reverse on the path) for the first time, v becomes the generated vertex. The complexity of the GraphGan is $O(V(V+E))$, where V is the set of nodes, and E is the set of edges.

JANE [71] is a framework that jointly distinguishes the real and fake combinations of the embeddings, topology information, and node features. JANE is composed of pluggable components, Embedding module, Generator module, and Discriminator module. Compared to typical GANs, the JANE framework includes an embedding module to produce network embeddings instead of generating new data (attribute network). Furthermore, the fake attribute network is not based only on the Gaussian distribution, as in GANs, but its generation is based on combining the embedding results of real attribute networks with the samples generated from Gaussian distribution. The Discriminator module is input with the real and fake topology and node features and embeddings and jointly distinguishes the real and fake combinations. The fake inputs are produced by the generator, based on the samples taken from a Gaussian distribution. Table 3 summarizes the most important features of the GAN-based strategies.

Table 3. GAN-based community detection strategies.

Paper	Year	Strategy	Complexity	Basic Experimental Results
[67]	2020	Two combined modules: a graph convolutional auto-encoder, which reconstructs a given graph from a latent representation and an adversarial network which is trained to discriminate if a sample is generated from the embedding.	$O(\|E\|md)$	Good average node clustering performance Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) and good link prediction with different prior distributions compared to a large number of competitive schemes.
[68]	2019	Combines AGM and GAN in a unified framework, which achieves good GAN performance and direct vertex-community membership representation in AGM. The Graph AGM generates the most likely motifs with graph structure awareness.	Not easy to infer due to random walk involved in the design of the AGM.	Experiments on synthetic data and real-world tasks show its improved performance over other state-of-the-art scheme for different clique sizes and good quality of overlapped communities in terms of NMI and F1-score metrics.
[69]	2020	Strategy that learns heuristics for community detection from training data with generative adversarial networks. The discriminator predicts if a community is real or fake and the generator tries to fool the discriminator by generating communities that fit the characteristics of real ones.	Not inferred	Good performance in terms of evaluation metrics for community detection like the bi-matching F1 and Jaccard scores.
[70]	2019	GraphGAN, a unified framework that designs models via adversarial training in a minimax game. The graph softmax implements the the generator, to resolve the limitations of traditional softmax function and satisfy the properties of normalization, graph structure awareness and computational efficiency.	$O(V(V+E))$	Achieves important gains in graph reconstruction, link prediction, node classification, and visualization over state-of-the-art schemes.
[71]	2021	A Jointly Adversarial Network Embedding (JANE) framework with pluggable components to benefit the embedding methods from the adversarial mechanism. JANE distinguishes the real and fake combinations of the embeddings, topology information and node features.	Not inferred	Superiority on link prediction and node clustering in terms of F1 score.

Comments on the GAN-Based Community Detection Schemes

After the study of the most recent GAN-based community detection schemes, the following observations can be made:

1. **Regularizations:** Deep learning strategies (AE- or GNN- based) focus on preserving the structure relationship and generally ignore the latent data distribution of the representation. Using GANs, the new data generated have the same distribution as real data, and this is a powerful tool for analyzing network data. For example, in [67], some form of regularization is introduced to the latent codes, thereby forcing them to follow some prior data distribution.
2. **Dense Overlapping Issues:** In real-world datasets, vertices can simultaneously belong to numerous communities and clustering algorithms cannot handle such dense overlapping. A weakness in performing overlapping community detection and network embedding simultaneously has been reported (see also our comments in the presentation of AE and CNN/GNN-based schemes). An idea is to combine the effective GANS with affiliation graph models (AGM), which can model densely overlapping community structures. Thus, the performance of GANs and the direct vertex-community membership representation of AGMs join forces to solve the dense overlapping issues [68]. In addition, the scheme presented in [69] is working on overlaps.
3. **Undistinguished real and not real embedding combinations:** Most of the GANs compare the generated fake data, which are produced based on Gaussian distribution sampling to the real data. In adversarial learning approaches, the Gaussian distribu-

tions cannot be rectified with real data so the results are not really beneficial for the network embeddings. Approaches like JANE [71] can be used to separate real and fake combinations of embeddings, topology information, and node features. Because of the combined topology information and node features, the Gaussian distribution is capable of capturing the semantic variations in latent space. Thus, the overall embedding results become more useful and the performance of network analysis is highly improved.

4. Future Trends

The strategies we have presented are the most recent developments in community detection strategies over the last two years. From our study, we have spotted several issues that need to be further elaborated, and improved solutions must be provided. In this section, we provide some of these issues that may lead the way for future trends.

1. **Large scale of modern networks:** The analysis of modern social networks becomes rather cumbersome, as their size and number keeps growing larger and larger. Older methods tried to employ some level of parallelism, but these suffered from low speedups. Generally, there is still a growing need for algorithms that do not fail to scale with the increasing number of users and the growing complexity of their relationships. Moreover, the existing schemes that were presented in this work generally have not been tested on graphs with billions of vertices or nodes using more powerful machines.
2. **Dynamic Community Detection:** Dynamic community detection requires that the changes in user relationships and in the overall network topology should also be considered. This means that models that extract spatio-temporal features of the social networks have to be developed. Generally, the irregular topologies, the frequent changes of relationships among users, and the network updates pose certain burdens on the processing of synchronous social networks, and more computational efforts are now required to accommodate these changes.
3. **Better use of computational resources:** Certain computations performed necessarily require the consumption of large quantities of computational resources. For example, in [45], the authors admit that the similarity matrix is obtained through processing of the adjacency matrix, and the calculations involved require large memory consumption and powerful experimental equipment. Thus, strategies that consume large amounts of resources or even exhaust the existing ones should be equipped with some type of decomposition strategy, which also may be accompanied with a well-designed parallelism scheme.
4. **Community overlapping:** As mentioned during the description of the most recent schemes in Section 2, most of the strategies presented do consider the situation of community overlaps (some exceptions have been mentioned in the text). In the future, most of these works should be expanded so that this very important issue is taken into account.
5. **Meaningful semantic representations of communities:** The community detection problem is a well-studied problem, and the first papers were published back in the beginning of this century. However, the datasets being used as inputs to construct communities contain rather discrete information like well-used words, symbols, forms of categorization for professions or other attributes or short phrases. Better datasets should be developed and used in deep learning strategies to extract and even predict communities. Generally, the vast amounts of information produced by the social networks should be used in a better and more informative way, in order to produce a more meaningful semantic representation of communities. In addition, the better interpretation of specific communities (medical, sport, politic, etc.) is closely related to this direction.
6. **Natural Language Processing (NPL embeddings)**: A recent trend is to compute node embeddings by using the second order random walks [72]. These embeddings aim at

keeping similar nodes close in their representations. Simplicial complexes represent a promising way for representing embeddings, but still the community extraction remains a challenge to be dealt with [73]. Other future trends include the ego networks, the streaming community detection (graph streams), and the temporal graphs.

7. **More straightforward comparisons are needed:** During our study, we have noticed that there is a lack of straightforward comparisons among the presented strategies. Perhaps, tools like the NetworKit [74] can be helpful in this regard. NetworKit is a growing open-source toolkit for large-scale network analysis and already contains novel algorithms from recently published research.

5. Conclusions

This survey provides a comprehensive literature review on the most recent advances in the field of community detection. The papers presented have been published within the last two years and the deep learning strategies employed. We aimed at organizing these works into categories, based on the tool being used. Thus, we derived the following taxonomy: AE-based, CNN/GNN-based, and GAN-based community detection. For each category, we discussed each work in detail, focusing on the architecture of each model and on the specific method being employed. We also presented some tables, where a synopsis of each work is given along with the computational analysis (whenever it was provided by the authors) and the main experimental results. At the end of each category of papers, we provided some general observations derived from our study. Finally, we provided some general future trends in the field of community detection. Many researchers are still active on the topic, and we expect that numerous papers will be published within the next few years, so we believe that these future trends may provide some ideas or opportunities for contribution in the field of community detection.

Author Contributions: Conceptualization, S.S.; methodology, S.S., S.A., S.K.; formal analysis, S.S., S.A., S.K.; investigation, S.S., S.A., S.K.; resources, S.A., S.K.; writing—original draft preparation, S.S.; writing—review and editing, S.S., S.A., S.K.; visualization, S.S., S.K.; supervision, S.A.; project administration, S.S.; funding acquisition, S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the University of Western Macedonia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, F.; Xue, S.; Wu, J.; Zhou, C.; Hu, W.; Paris, C.; Nepal, S.; Yang, J.; Yu, P.S. Deep Learning for Community Detection: Progress, Challenges and Opportunities. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), Yokohama, Japan, 11–17 January 2021; pp. 4981–4987.
2. Jin, D.; Wang, K.; Zhang, G.; Jiao, P.; He, D.; Fogelman-Soulie, F.; Huang, X. Detecting communities with multiplex semantics by distinguishing background, general and specialized topics. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 2144–2158. [CrossRef]
3. Satuluri, V.; Wu, Y.; Zheng, X.; Qian, Y.; Wichers, B.; Dai, Q.; Tang, G.M.; Jiang, J.; Lin, J. Simclusters: Community-based representations for heterogeneous recommendations at twitter. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, Online, 23–27 August 2020; pp. 3183–3193.
4. Keyvanpour, M.R.; Shirzad, M.B.; Ghaderi, M. AD-C: A new node anomaly detection based on community detection in social networks. *Int. J. Electron. Bus.* **2020**, *15*, 199–222. [CrossRef]
5. Chen, Y.; Li, X.; Xu, J. Convexified modularity maximization for degree-corrected stochastic block models. *Ann. Stat.* **2018**, *46*, 1573–1602. [CrossRef]
6. Pizzuti, C.; Socievole, A. Multiobjective optimization and local merge for clustering attributed graphs. *IEEE Trans. Cybern.* **2019**, *50*, 4997–5009. [CrossRef]
7. Gulikers, L.; Lelarge, M.; Massoulié, L. A spectral method for community detection in moderately sparse degree-corrected stochastic block models. *Adv. Appl. Probab.* **2017**, *49*, 686–721. [CrossRef]

8. Dhilber, M.; Bhavani, S.D. Community Detection in Social Networks Using Deep Learning. In Proceedings of the 16th International Conference, ICDCIT 2020, Bhubaneswar, India, 9–12 January 2020.
9. Souravlas, S.; Sifaleras, A.; Katsavounis, S. A Parallel Algorithm for Community Detection in Social Networks, Based on Path Analysis and Threaded Binary Trees. *IEEE Access* **2019**, *7*, 20499–20519. [CrossRef]
10. Souravlas, S.; Sifaleras, A.; Katsavounis, S. Hybrid CPU-GPU Community Detection in Weighted Networks. *IEEE Access* **2020**, *8*, 57527–57551. [CrossRef]
11. Jin, D.; Yu, Z.; Jiao, P.; Pan, S.; Yu, P.S.; Zhang, W. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Representation. *arXiv* **2021**, arXiv:2101.01669.
12. Leskovec, J.; Krevl, A. *SNAP Datasets: Stanford Large Network Dataset Collection*; Tech. Rep.; University Stanford: Stanford, CA, USA, 2014. Available online: http://snap.stanford.edu/data (accessed on 26 July 2021).
13. Citation Network Dataset: DBLP+Citation, ACM Citation Network. Available online: https://www.aminer.org/citation (accessed on 26 July 2021).
14. Email-EU. Available online: https://paperswithcode.com/dataset/email-eu (accessed on 8 May 2012).
15. Lu, Z.; Sun, X.; Wen, Y.; Cao, G.; Porta, T.L. Algorithms and applications for community detection in weighted networks. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 2916–2926. [CrossRef]
16. Qiao, S.; Han, N.; Gao, Y.; Li, R.-H.; Huang, J.; Guo, J.; Gutierrez, L.A.; Wu, X. A fast parallel community discovery model on complex networks through approximate optimization. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1638–1651. [CrossRef]
17. Matin, P.; Zhan, J. Optimized Label Propagation Community Detectionon Big Data Networks. In Proceedings of the 2018 International Conference on Big Data and Education(ICBDE'18), Honolulu, HI, USA, 9–11 March 2018; pp. 57–62.
18. Dusan, D.; Aloise, D.; Mladenovic, N. Ascent–descent variable neigh-borhood decomposition search for community detection by modularity maximization. *Ann. Oper. Res.* **2018**, *272*, 273–287.
19. Fan, S.; Wang, X.; Shi, C.; Lu, E.; Lin, K.; Wang, B. One2Multi Graph Autoencoder for Multi-view Graph Clustering. In Proceedings of the Web Conference 2020 (WWW '20), Taipei, Taiwan, 20–24 April 2020; pp. 3070–3076. [CrossRef]
20. Luo, D.; Ni, J.; Wang, S.; Bian, Y.; Yu, X.; Zhang, X. Deep Multi-Graph Clustering via Attentive Cross-Graph Association. In Proceedings of the 13th International Conference on Web Search and Data Mining, (WSDM 2020), Houston, TX, USA, 3–7 February 2020; pp. 393–401.
21. Sarkar, A.; Mehta, N.; Rai, P. Graph Representation Learning via Ladder Gamma Variational Autoencoders. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), New York, NY, USA, 7–12 February 2020; pp. 5604–5611.
22. Shi, H.; Fan, H.; Kwok, J.T. Effective Decoding in Graph Auto-Encoder Using Triadic Closure. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), New York, NY, USA, 7–12 February 2020; pp. 906–913.
23. Shang, J.; Wang, C.; Xin, X.; Ying, X. Community detection algorithm based on deep sparse autoencoder. *J. Softw.* **2017**, *28*, 648–662.
24. Wu, L.; Zhang, Q.; Chen, C.; Guo, K.; Wang, D. Deep Learning Techniques for Community Detection in Social Networks. *IEEE Access* **2020**, *8*, 96016–96026. [CrossRef]
25. Cao, J.; Jin, D.; Dang, J. Auto-encoder based community detection with adaptive integration of network topology and node contents. In Proceedings of the International Conference on Knowledge Science, Engineering and Management, Changchun, China, 17–19 August 2018; pp. 184–196.
26. Cao, J.; Jin, D.; Yang, L.; Dang, J. Incorporating network structure with node contents for community detection on large networks using deep learning. *Neurocomputing* **2018**, *297*, 71–81. [CrossRef]
27. Bhatia, V.; Rani, R. A distributed overlapping community detection model for large graphs using autoencoder. *Future Gener. Comput. Syst.* **2019**, *94*, 16–26. [CrossRef]
28. Shao, M.; Li, S.; Ding, Z.; Fu, Y. Deep linear coding for fast graph clustering. In Proceedings of the 24th International Conference on Artificial Intelligence, Virtually, Online, 13–15 April 2021; pp. 3798–3804.
29. Song, C.; Liu, F.; Huang, Y.; Wang, L.; Tan, T. Auto-encoder based data clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 117–124.
30. Tian, F.; Gao, B.; Cui, Q.; Chen, E.; Liu, T.-Y. Learning deep representations for graph clustering. In Proceedings of the 28th AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 1293–1299.
31. Yann, L.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444.
32. Cavallari, S.; Zheng, V.W.; Cai, H.; Chang, K.C.-C.; Cambria, E. Learning community embedding with community detection and node embedding on graphs. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 377–386.
33. Yu, W.; Zheng, C.; Cheng, W.; Aggarwal, C.C.; Song, D.; Zong, B.; Chen, H.; Wang, W. Learning deep network representations with adversarially regularized autoencoders. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2663–2671.
34. Xu, R.; Che, Y.; Wang, X.; Hu, J.; Xie, Y. Stacked auto-encoder based community detection method via an ensemble clustering framework. *Inf. Sci.* **2020**, *526*, 151–165. [CrossRef]
35. Ivannikova, E.; Park, H.; Hamalainen, T.; Lee, K. Revealing community structures by ensemble clustering using group diffusion. *Inform. Fusion* **2018**, *42*, 24–36. [CrossRef]
36. Choong, J.J.; Liu, X.; Murata, T. Optimizing Variational Graph Autoencoder for Community Detection with Dual Optimization. *Entropy* **2020**, *22*, 197. [CrossRef] [PubMed]

37. Choong, J.J.; Liu, X.; Murata, T. Learning community structure with variational autoencoder. In Proceedings of the IEEE International Conference on Data Mining, Singapore, 17–20 November 2018; pp. 69–78.
38. Chen, Z.; Chen, C.; Zhang, Z.; Zheng, Z.; Zou, Q. Variational graph embedding and clustering with Laplacian eigenmaps. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; pp. 2144–2150.
39. Dai, D.; Yu, L.; Wei, H. Parameters sharing in residual neural networks. *Neural Process. Lett.* **2020**, *51*, 1393–1410. [CrossRef]
40. Aich, S.; Yamazaki, M.; Taniguchi, Y.; Stavness, I. Multi-scale weight sharing network for image recognition. *Pattern Recognit. Lett.* **2020**, *131*, 348–354. [CrossRef]
41. Al-Andoli, M.; Cheah, W.P.; Tan, S.C. Deep learning-based community detection in complex networks with network partitioning and reduction of trainable parameters. *J. Ambient. Intell. Human Comput.* **2021**, *12*, 2527–2545. [CrossRef]
42. Cui, G.; Zhou, J.; Yang, C.; Liu, Z. Adaptive Graph Encoder for Attributed Graph Embedding. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, Online, 23–27 August 2020; pp. 976–985. [CrossRef]
43. Clauset, A.; Newman, M.E.; Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **2004**, *70*, 066111. [CrossRef]
44. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *10*, P10008. [CrossRef]
45. Fei, R.; Sha, J.; Xu, Q.; Hu, B.; Wang, K.; Li, S. A new deep sparse autoencoder for community detection in complex networks. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 91. [CrossRef]
46. Chen, Z.; Li, X.; Bruna, J. Supervised Community Detection with Line Graph Neural Networks. In Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
47. He, D.; You, X.; Feng, Z.; Jin, D.; Yang, X.; Zhang, W. A Network-Specific Markov Random Field Approach to Community Detection. In Proceedings of the 32th AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
48. Jin, D.; Liu, Z.; Li, W.; He, D.; Zhang, W. Graph Convolutional Networks Meet Markov Random Fields: Semi-Supervised Community Detection in Attribute Networks. In Proceedings of the Thirty-Third Conference on Artificial Intelligence (AAAI-19), Honolulu, HI, USA, 27 January–1 February 2019; pp. 152–159.
49. He, D.; Song, Y.; Jin, D.; Feng, Z.; Zhang, B.; Yu, Z.; Zhang, W. Community-Centric Graph Convolutional Network for Unsupervised Community Detection. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), Yokohama, Japan, 11–17 July 2020; pp. 3515–3521.
50. Sperlí, G. A deep learning based community detection approach. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; pp. 1107–1110.
51. Cai, B.; Wanga, Y.; Zeng, L.; Hua, Y.; Li, H. Edge classification based on Convolutional Neural Networks for community detection in complex network. *Phys. A Stat. Mech. Appl.* **2020**, *556*, 124826. [CrossRef]
52. Xu, Y.; Chi, Y.; Tian, Y. Deep convolutional neural networks for feature extraction of images generated from complex networks topologies. *Wireless Pers. Commun.* **2018**, *103*, 327–338. [CrossRef]
53. Xu, J.; Li, M.; Jiang, J.; Ge, B.; Cai, M. Early prediction of scientific impact based on multi-bibliographic features and convolutional neural network. *IEEE Access* **2019**, *7*, 92248–92258. [CrossRef]
54. Hanocka, R.; Hertz, A.; Fish, N.; Giryes, R.; Fleishman, S.; Cohen-Or, D. MeshCNN: A network with an edge. *ACM Trans. Graph.* **2019**, *38*, 1–12. [CrossRef]
55. Gao, W.; Zhou, P. Customized high performance and energy efficient communication networks for AI chips. *IEEE Access* **2019**, *7*, 69434–69446. [CrossRef]
56. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; pp. 3670–3676.
57. Shchur, O.; Günnemann, S. Overlapping Community Detection with Graph Neural Networks. In Proceedings of the First International Workshop on Deep Learning on Graphs (In Conjunction with the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining), Anchorage, AK, USA, 4–8 August 2019; pp. 1–7.
58. Levie, R.; Monti, F.; Bresson, X.; Bronstein, M.M. CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters. *IEEE Trans. Signal Process.* **2019**, *67*, 97–109. [CrossRef]
59. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3844–3852.
60. Bo, D.; Wang, X.; Shi, C.; Zhu, M.; Lu, E.; Cui, P. Structural Deep Clustering Network. In Proceedings of the Web Conference, Taipei, Taiwan, 20–24 April 2020; pp. 1400–1410.
61. Cheng, J.; Wang, Q.; Tao, Z.; Xie, D.; Gao, Q. Multi-View Attribute Graph Convolution Networks for Clustering. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), Yokohama, Japan, 11–17 July 2020; pp. 2973–2979.
62. Zhang, X.; He, L.; Chen, K.; Luo, Y.; Zhou, J.; Wang, F. Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson's disease. In Proceedings of the AMIA Annual Symposium Proceedings, San Francisco, CA, USA, 3–7 November 2018; p. 1147.

63. Ma, T.; Xiao, C.; Zhou, J.; Wang, F. Drug similarity integration through attentive multiview graph auto-encoders. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; pp. 3477–3483.
64. Shi, M.; Tang, Y.; Zhu, X.; Wilson, D.; Liu, J. Multi-Class Imbalanced Graph Convolutional Network Learning. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), Yokohama, Japan, 11–17 July 2020; pp. 2879–2985.
65. Hu, R.; Pan, S.; Long, G.; Lu, Q.; Zhu, L.; Jiang, J. Going Deep: Graph Convolutional Ladder-Shape Networks. In Proceedings of the Thirty-Fourth Conference on Artificial Intelligence (AAAI-20), New York, NY, USA, 7–12 February 2020; pp. 2838–2845.
66. Liu, Y.; Wang, X.; Wu, S.; Xiao, Z. Independence Promoted Graph Disentangled Networks. In Proceedings of the Thirty-Fourth Conference on Artificial Intelligence (AAAI-20), New York, NY, USA, 7–12 February 2020; pp. 4916–4923.
67. Pan, S.; Hu, R.; Fung, S.-F.; Long, G.; Jiang, J.; Zhang, C. Learning Graph Embedding With Adversarial Training Methods. *IEEE Trans. Cybern.* **2020**, *50*, 2475–2487. [CrossRef] [PubMed]
68. Jia, Y.; Zhang, Q.; Zhang, W.; Wang, X. CommunityGAN: Community Detection with Generative Adversarial Nets. In Proceedings of the WWW '19: The World Wide Web Conference, San Francisco, CA, USA, 13 May 2019; pp. 784–794.
69. Zhang, Y.; Xiong, Y.; Ye, Y.; Liu, T.; Wang, W.; Zhu, Y.; Yu, P.S. SEAL: Learning Heuristics for Community Detection with Generative Adversarial Networks. In Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Online, 6–10 July 2020; pp. 1103–1113.
70. Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Li, W.; Xie, X.; Guo, M. Learning Graph Representation with Generative Adversarial Nets. *IEEE Trans. Knowl. Data Eng.* **2019**, *33*, 1–13. [CrossRef]
71. Yang, L.; Wang, Y.; Gu, J.; Wang, C.; Cao, X.; Guo, Y. JANE: Jointly Adversarial Network Embedding. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), Yokohama, Japan, 11–17 July 2020; pp. 1381–1387.
72. Kulkarni, S.; Katariya, J.K.; Potika, K. GloVeNoR: GloVe for Node Representations with Second Order Random Walks. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), The Hague, The Netherlands, 7–10 December 2020; pp. 536–543. [CrossRef]
73. Billings, J.C.W.; Hu, M.; Lerda, G.; Medvedev, A.N.; Mottes, F.; Onicas, A.; Santoro, A.; Petri, G. Simplex2Vec embeddings for community detection in simplicial complexes. *arXiv* **2019**, arXiv:1906.09068.
74. Staudt, C.L.; Meyerhenke, H. Engineering parallel algorithms for community detection in massive networks. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 171–184. [CrossRef]