Journal of Optimization of Soft Computing (JOSC)

Vol. 3, Issue 1, pp: (1-23), Spring-2025
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Variational Graph Autoencoder for Unsupervised Community Detection in Attributed Social Networks

Omid Rashnodi [1], Maryam Rastegarpour[*2], Azadeh Zamanifar[1], Parham Moradi[3]

1. Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
2. Department of Computer, College of Engineering, Saveh Branch, Islamic Azad University, Saveh, Iran.
3. School of engineering, RMIT University Melbourne, Australia

Article Info

Article History:
Received: 2025/02/21
Revised: 2025/03/08
Accepted: 2025/04/12

DOI:

Keywords:
Community Detection, Attributed Social Networks, Variational Graph Autoencoder, Graph Convolutional Networks, Deep Learning, Node Embeddings, Network Topology

Omid Rashnodi
omid.rashnodi@iau.ac.ir

* Maryam Rastegarpour
m.rastgarpour@gmail.com

Azadeh Zamanifar
azamanifar@srbiau.ac.ir

Parham Moradi
p.moradi@uok.ac.ir

Abstract

This paper introduces a novel approach named VGAEE (Variational Graph AutoEncoder Embedding), an innovative deep-learning framework for detecting communities in attributed social networks. By synergistically integrating node content with network topology, VGAEE aims to enhance the quality of community identification. Initially, we computed the modularity and Markov matrices of the input graph. These matrices were then concatenated and used as the input for the VGAEE to create a meaningful representation of the graph. In the decoder component of VGAEE, two layers of Graph Convolutional Networks (GCN) are employed. Subsequently, a K-Nearest Neighbors (KNN) algorithm was used for clustering communities based on the embeddings generated previously. We conducted experiments on three benchmark datasets—Cora, Citeseer, and PubMed—and compared the results with various baseline and state-of-the-art methods using Accuracy (ACC) and Normalized Mutual Information (NMI) as evaluation metrics. The findings demonstrate that VGAEE significantly improves community detection performance, achieving an accuracy of 84.5% on Cora , 80.5% on PubMed, and 75.6% on Citeseer. In terms of NMI, VGAEE reached 70.46% on Cora, 55.60% on PubMed, and 57.06% on Citeseer, consistently outperforming existing methods. These results confirm the superiority of VGAEE in accurately capturing community structures within large, complex networks, making it a highly effective tool for unsupervised community detection.

1.introduction

The study of community structures within networks has advanced significantly since the early days of sociological research, evolving into a critical field that employs complex mathematical tools for large-scale data analysis. Since the groundbreaking work of Girvan and Newman in 2002, identifying and understanding these structures has become essential for analyzing the composition and function of various networks, with applications spanning diverse fields such as epidemiology and marketing.

Despite advancements in topological, content-based, and graph-theoretical approaches to community detection, existing methods still face several challenges—especially in the quality of vector representations for network nodes. Many current techniques fail to fully capture both the structural and contextual information of nodes. As a result, they often struggle with tasks like clustering and classification and are unable to keep up with the increasing demands of growing and more complex networks.

This paper explores the limitations of traditional community detection methods, particularly when applied to large-scale or high-dimensional networks constrained by computational power and data volume. These challenges significantly hinder the effectiveness of conventional approaches in analyzing modern, complex relational data. To address these issues, this study leverages graph neural networks (GNNs), a specialized branch of deep learning tailored for graph data. By reducing network dimensions and enhancing node representations, this approach accelerates the community detection process. Additionally, this research integrates the modularity matrix with the Markov matrix to improve detection accuracy, making the proposed methods more efficient and suitable for complex network structures. The contributions and innovations of this study are summarized as follows:

- **Integration of Node Content and Network Topology:** The VGAEE (Variational Graph AutoEncoder Embedding) framework uniquely combines node content with network topology to enhance community detection in attributed social networks. This integration provides a more comprehensive understanding of both network structure and content.
- **Use of Modularity and Markov Matrices:** The approach introduces an innovative step by computing modularity and Markov matrices from the input graph. These matrices are then concatenated and used as inputs for VGAEE, enabling a more nuanced representation of the graph structure.
- **Graph Convolutional Networks in the Decoder:** The application of two layers of Graph Convolutional Networks (GCN) within the VGAEE decoder is a novel feature. This technique leverages GCNs' capabilities to learn and generate high-quality embeddings that accurately reflect the true community structure.
- **Community Clustering via KNN:** After generating embeddings, VGAEE utilizes the K-Nearest Neighbors (KNN) algorithm for clustering. This innovative step effectively combines a traditional machine learning algorithm with a deep learning framework to improve community identification.
- **Benchmark Dataset Experiments:** The paper conducts extensive experiments using three widely recognized benchmark datasets—Cora, Citeseer, and PubMed. These rigorous tests validate the model's effectiveness and provide a strong basis for comparison with baseline and state-of-the-art methods.
- **Superior Performance Metrics:** The VGAEE framework outperforms existing algorithms in both accuracy and Normalized Mutual Information (NMI), demonstrating its superior ability to identify and differentiate community structures in complex networks.

Community detection is widely recognized as an NP-hard problem that presents a range of computational challenges. This paper addresses these issues by focusing on both computational efficiency and detection accuracy in attributed social networks. By utilizing GNNs, the study introduces innovative embedding techniques and improved graph representation learning strategies, ultimately providing a more effective approach to community detection.

We structure the remainder of this paper as follows: **Section 2** surveys the existing literature on graph convolutional networks and dual embedding techniques, outlining fundamental advances and identifying the gaps that our study aims to address. **Section 3** introduces the necessary concepts and

notations, providing the foundation for understanding the methodologies discussed later. **Section 4** presents a detailed description of the proposed algorithm, VGAEE, along with its pseudocode. **Section 5** offers a comprehensive overview of the datasets used for testing, explains the evaluation metrics employed to assess performance, and describes the chosen parameters and experimental setup. Finally, **Section 6** presents the conclusion and discusses directions for future work.

2. Literature review

With recent advances in information technology and the digital world, complex network theory has found applications in various fields, including social networks, biological networks, and internet networks. One of the key challenges in complex network research is community detection, which aims to identify the structural properties of networks. Communities in a network are formed by groups of nodes that have stronger internal connections and fewer connections with external nodes. Early community detection methods primarily relied on the topological characteristics of networks, and numerous approaches have been proposed based on different criteria for similarity and proximity among groups. Before the development of deep learning techniques, community detection methods were broadly categorized into two main groups: Hierarchical methods and Partitioning methods. Hierarchical methods begin with either a partition where each node is considered an independent cluster or a partition where all nodes belong to a single community. Clusters are then iteratively merged or divided based on a quality measurement criterion, forming a hierarchical structure. While hierarchical methods do not require prior knowledge of the number of communities, they do depend on a specific criterion to determine meaningful partitions.

On the contrary, partitioning methods identify clusters through iterative member allocation. These methods assess the quality of partitions by optimizing one or more objective functions. Some commonly used partitioning techniques include finding the largest number of cliques in a graph [1], modularity maximization [2], matrix decomposition [3], seed expansion [4], linear sparse coding [5], sparse linear coding [5], and evolutionary algorithms [1]. Both hierarchical and partitioning methods involve high computational costs, making them inefficient for large-scale networks. In other words, these approaches struggle to find optimal solutions within a reasonable timeframe. To address this issue, more adaptive local methods have been introduced to detect separate and overlapping communities more efficiently [6]. One such example is label propagation-based methods, which use the local expansion of node labels to identify communities in linear time [7].

Deep learning (DL) techniques are widely applied in various fields, including computer and social sciences, economics, agriculture, healthcare, and medicine [8]. Network representation learning (NRL) converts complex network structure data into a low-dimensional, manageable space, making it useful across these diverse applications. This approach includes learning network representations [9], network embedding [10], and graph embedding [11], all designed to preserve the network's typological structure, vertex content, and auxiliary information.

These advanced learning methods have transformed the way complex classification, clustering, and prediction models are constructed through effective graph data representation. They simplify the execution of analytical tasks that would traditionally require more complex models. Network Representation Learning (NRL) techniques focus on reducing the dimensionality of network vertices representations while preserving essential topological and content features of the network [9]. These representations are then utilized as vector inputs for machine learning tasks such as node classification and link prediction, fostering the creation of more refined and effective NRL strategies for complex networks [10]. Methods for graph representation learning are generally divided into three main categories: probabilistic models, deep learning-based algorithms, and matrix decomposition algorithms. Each category will be further discussed to highlight their unique approaches and applications.

Probabilistic Models: Techniques such as LINE [12] and Node2vec [13] are designed to extract varied graph patterns to enhance embedding learning. Node2vec efficiently maps nodes into a vector space, which significantly boosts the performance of link prediction and node classification tasks. LINE is notable for its large-scale application, utilizing edge sampling strategies to address the typical challenges associated with stochastic gradient descent. This adaptation improves the graph embedding process while maintaining high efficiency.

Deep Learning-Based Algorithms: DeepWalk [14] is a prime example of integrating deep learning with graph theory. It excels at encoding the complete structural information of graphs by leveraging the local structural information of vertices and incorporating the Skip-Gram model within the framework of random walks. This approach has been particularly successful in social networks for tasks like multilabel classification. Deep learning models capture the nonlinear dynamics of complex, extensive networks by analyzing various relational data, including nodes, neighbors, edges, subgraphs, and community features. These models are particularly effective in handling sparse networks and excel in unsupervised learning contexts. Algorithms like DNGR, SNDE, and ANRL [15] use deep autoencoder models for representing high-dimensional data. Conversely, end-to-end network-based methods like SNE [16] and DeepGL [17] blend structural and attribute data to enhance graph representation learning. Additionally, MGAE [18] utilizes a single-layer autoencoder, simplifying clustering tasks, while HNE [19] merges deep autoencoder neural networks with convolutional networks to process adjacent vectors and images.

Matrix Decomposition Algorithms: This category includes techniques like M-NMF [20] and TADW [21], which are focused on matrix decomposition to effectively learn node representations. These methods are crucial for untangling complex network structures, enabling deeper insights into network dynamics and interactions.

Together, these methods establish a solid framework for managing and analyzing complex networks across diverse domains, accommodating a broad spectrum of applications from theoretical research to practical, real-world problem-solving. This comprehensive approach ensures that insights derived from graph theory and network analysis are not only theoretically sound but also applicable in solving actual challenges in fields such as social networking, bioinformatics, and telecommunications.

Wang et al. [22] effectively utilized a graph autoencoder to achieve deep representations, which were then applied in a spectral clustering algorithm to enhance graph clustering. In a similar vein, He et al. [23] developed a nonlinear restructuring approach for modularity matrices using deep neural networks, which they further adapted into a semi-supervised community detection algorithm by incorporating constraints on paired graph nodes. Both approaches address significant challenges associated with high computational demands and the need for extensive parameter tuning, such as determining the number of clusters, which often remains undefined in large and heterogeneous networks globally. More recently, advancements in graph neural networks (GNNs), including graph convolutional networks (GCNs), have been introduced to address community detection issues [24, 25]. GCNs amalgamate the information from neighboring nodes through deep convolutional layers in graphs, employing convolutional operations similar to those used in convolutional neural networks to extract and represent complex community features based on network topology and node characteristics [26].

Originally, Graph Convolutional Networks (GCNs) were not designed with community detection in mind, meaning they did not specifically target community structures during node embedding learning, nor did they impose constraints on the structural relationships between communities and nodes. Addressing this limitation, Jin et al. [27] introduced a semi-supervised community detection model named MRFasGCN. This model integrates a GCN with the Markov Random Fields (MRF) statistical model to enhance community detection capabilities. The innovation lies in extending the Markov Random Field into a new convolutional layer within the GCN framework, thereby allowing MRFasGCN to effectively oversee and refine the overall outcomes of the GCN's community detection efforts.

Sun et al. [28] developed a framework to enhance network embedding for clustering nodes in attributed graphs. This innovative framework concurrently learns graph-based and cluster-oriented representations. It consists of three key components: a graph autoencoder module, a soft modularity maximization module, and a self-clustering module. The graph autoencoder module is tasked with learning node embeddings that incorporate both the topological structure and the node properties.

Jin et al. [29] introduced an unsupervised model for community detection using GCN embedding, employing the GCN as the primary structure of the encoder to reconcile two types of information: topology and property. This model utilizes a dual encoder setup to extract distinct embeddings from these two data sources.

Luo et al. [30] presented a deep-learning model that aims to simultaneously identify communities and structural holes using a GCN-based encoder. This

approach leverages the GCN's ability to integrate network topology and node properties for community detection. However, the model faces challenges as it (1) learns representations through encoding topological features and node properties without considering community-specific features, resulting in embeddings that are not community-centric, and (2) operates as a semi-supervised rather than a fully unsupervised model.

Wang et al. [31, 32] proposed a novel approach involving nonnegative matrix decomposition, introducing a community membership matrix and a community characteristic matrix. They also developed several efficient updating rules that ensure convergence. This method enhances community detection by incorporating node attributes, which also provide a semantic interpretation of the communities.

Efforts have also been made to develop semi-supervised methods for community detection by integrating network representations with data labels through graph-based regulation to identify unlabeled nodes. Young et al. [33] utilized node representations to predict network backgrounds and applied node labels to facilitate various transfer and inductive learning strategies. Recent advancements include the introduction of graph convolutional networks for network analysis, with GCN-based methods enhancing both network topology and attribute data analysis. Unlike most semi-supervised approaches that predominantly focus on network structure, these methods require a substantial number of node labels to classify unlabeled nodes. Sun et al. also introduced a graph convolutional autoencoder framework for clustering nodes, and several unsupervised methods have been recently proposed to advance this field.

In [34], a supervised model within the CNN framework was introduced for typological defect networks. This model incorporates two CNN layers with max-pooling operators to represent the network structure and a fully connected DNN layer dedicated to community detection. The convolutional layers are designed to capture the local attributes of each node from multiple perspectives. Testing on Topological Interference Networks (TINs), with a configuration of 10% labeled nodes and 90% unlabeled nodes, this model achieved an impressive 80% accuracy in community detection, highlighting that incorporating high-order neighbor representation can significantly enhance the accuracy of detecting communities.

In [35], a model named the Linear Graph Neural Network (LGNN) was proposed to enhance the efficiency of the Stochastic Block Model (SBM) in community detection while also reducing computational costs. The LGNN effectively learns the represented attributes of nodes in directed networks by employing a combination of non-backtracking operators and messaging rules, streamlining the process and optimizing performance.

In [36], the CommDGI model was introduced, which optimizes graph representation and clustering concurrently through mutual information on nodes and communities while aiming to maximize graph modularity. This approach utilizes k-means clustering to strategically align nodes with cluster centers, enhancing the clarity and effectiveness of community detection.

Additionally, while Spectral GCNs adeptly reveal all hidden attributes of a node's neighborhood, they can lead to over-smoothing, which may obscure distinct community structures. To counter this effect, graph convolutional ladder-shaped networks have been developed as a novel GCN architecture. Inspired by the U-Net model in the CNN domain, this unsupervised community detection approach [37] aims to mitigate the over-smoothing issue, ensuring more distinct and actionable community detection outcomes.

In scenarios where various types of links are treated as simple edges, GCNs typically represent each link separately and then aggregate them, which can lead to redundancy in representation. To address this, IPGDN [38] introduces a methodology that segments neighborhoods into different sections and autonomously identifies independent hidden attributes of a graph. This approach simplifies the process of community detection. The IPGDN model is enhanced by the use of the Hilbert–Schmidt independence criterion in neighborhood routing, facilitating more precise and effective community detection. Moreover, adaptive graph convolution has been developed to identify communities within attributed graphs. This technique relies on both structural data and representational features, categorizing neighboring nodes and nodes with similar attributes into the same community cluster. In this process, two graph signals are combined, necessitating the filtering of high-frequency noise, which is achieved through the design of a low-pass graph filter with a specific frequency response function.

In [39], a sophisticated method using Cayley polynomials was introduced to achieve high-order approximations within the spectral convolutional framework of graph neural networks. Although the exploration of GCN filters is relatively limited, CayleyNets are distinguished by their use of low-pass filters that effectively utilize extensive community data for precise community identification.

In [40], challenges associated with graph convolutional neural networks in processing complex relational graphs, such as excessive smoothing during node classification, are addressed. The newly developed SM-GCN model strives to enhance node categorization accuracy by reducing dependency on individual node features and incorporating scattering embeddings. This innovation is specifically designed to mitigate the over-smoothing effect, ensuring more distinct and accurate node classifications in complex network structures.

In [41], a new model known as the Graph Convolutional Fusion Model (GCFM) was introduced for enhancing community detection in multiplex networks, which are composed of multiple layers, each representing a different type of relationship among the same set of nodes. The GCFM utilizes a graph convolutional autoencoder for each layer to capture and encode the structural features specific to each layer while considering the connections between neighboring nodes. This approach allows for a more nuanced and accurate detection of communities across the complex interlayer dynamics of multiplex networks.

In [42], the Temporal Attributed Network Matrix Factorization (TANMF) algorithm was developed to detect dynamic modules within cancer temporal-attributed networks, incorporating both genomic data and temporal network changes. The experimental results showed that TANMF not only surpasses existing methods in accuracy but also enriches identified modules with known biological pathways and demonstrates correlations with patient survival outcomes, providing valuable insights into cancer progression.

In [43], the Joint Learning Dynamic Edge Community (jLDEC) algorithm was proposed for identifying dynamic communities within temporal networks. This algorithm integrates graph representation learning with community detection and the dynamics of network edges into a unified framework, significantly enhancing the precision of community detection. The jLDEC algorithm has been shown to perform better than traditional methods, particularly in accurately capturing the changing dynamics of community structures within temporal networks.

In [44], the Network Embedding to Nonnegative Matrix Factorization (NE2NMF) algorithm addresses the challenge of detecting dynamic communities by combining network embedding with nonnegative matrix factorization. It incorporates a third-order smoothness strategy that accounts for previous, current, and subsequent network snapshots, thereby providing a more comprehensive characterization of community dynamics. Experimental validations confirm that NE2NMF not only improves accuracy but also enhances the robustness of community detection compared to conventional approaches, making it particularly effective in dynamic network environments.

In [45], the Joint Learning of Multidimensional Clustering (jLMDC) algorithm was presented for dynamic community detection in temporal networks. This approach integrates feature extraction and clustering into a single framework, significantly enhancing both the accuracy and efficiency of detecting dynamic communities. Compared to traditional methods, jLMDC shows marked improvements in computational speed and accuracy, making it highly effective for managing large-scale networks and their complex community dynamics.

In [46], the Deep Autoencoder-like Nonnegative Matrix Factorization for Multi-View Learning (DANMF-MRL) was introduced, employing a deep encoding process to create a representation matrix. This matrix is subsequently decoded to reconstruct the original data. Utilizing the DANMF framework, the method addresses the challenges of maintaining consistency and complementarity in multi-view data, greatly enriching the depth and comprehensiveness of data representations.

In [47], a Nonnegative Matrix Factorization-based Multi-View Learning (MRL) framework was proposed, which considers two critical components: an exclusivity term to leverage diverse intra-view information and a consistency term to ensure unified representations across multiple views. Additionally, a local manifold component is included to preserve the local geometric structure of the data. An alternating optimization algorithm based on multiplicative updates was introduced to solve this problem, with proven convergence.

Review studies have shown that graph embedding methods can substantially improve efficiency and

reduce the time needed for community detection in social networks. Variational Graph AutoEncoder (VGE), a deep learning-based embedding technique, is utilized for network representation learning. However, a significant challenge with GCNs is their lack of inherent community orientation, which can result in node representations that may not be sufficiently precise for effective community detection. To address this, the k-core algorithm is used first to filter the graph and eliminate less significant nodes, thereby reducing the graph's size and enhancing the distinctiveness of its communities. Subsequently, the modularity matrix and the Markov matrix, which represent the graph's structure and content respectively, are concatenated and used as input for the VGE. The VGE encoder processes this input through two layers of the graph convolution network, producing a reduced-dimensional representation for each node. This representation is then normalized and utilized as the input for the k-nearest neighbors clustering algorithm to identify communities.

3. Preliminaries and Notation

This section provides a concise introduction to the foundational concepts, including essential notations and the formal problem statement. These preliminaries establish the groundwork necessary for understanding the proposed approach.

3.1. Attributed graph

Suppose that $G = (V, E, A, X)$ is an attributed network where V is a set of vertices $\{v_1, v_2, \dots, v_n\}, E$ is a set of edges between nodes, A is the adjacency matrix, and X is the attribute matrix where an element X_{ip} represents the value of the *p-th* attribute for the vertex v_i. In adjacency matrix *A*, if there is an edge between the two vertices of v_i and v_j then $a_{ij} > 0$. For weightless networks, if there is an edge, $a_{ij} = 1$; otherwise, $a_{ij} = 0$. if the network is not direct, $a_{ij} = a_{ji}$ also holds [50].

3.2. Community and community detection

Consider that we have the community set $C = \{C_1, C_2, \dots, C_r\}$. Each community is a network partition with regional structures and shared cluster attributes. The node v_i that is clustered in the community C_i It should meet the condition that the internal degree of every node is greater than its external degree. In this paper, community detection is considered in the attributed graph. The graph has G attributes and the number of *r* communities. This paper aims to find the function $f: v \to \{1,2,3, \dots, r\}$ such that r is true for all $f(v_i) = r$ nodes of the *r* community. Function partitions should follow the following principles: (1) Nodes of a group are connected, while the nodes are not connected in different groups. (2) Nodes in the same community tend to have similar attribute values, while those from different communities may vary relatively, even if they are neighbors at the graph level. (3) The function can adequately maintain the attributed graph's node attributes and structural information. Finally, we can find the groups separate from the nodes and their inductive subnodes, i.e., communities.

3.3. Decomposition k-core:

Assume a graph G = *(V, E)* of $|V| = n$ vertices and $|E| = e$ edges; a *k*-core is defined as follows: A subgraph $H = (C, E|C)$ induced by the set $C \subseteq V$ is a *k*-core or a core of order *k* iff $\forall v \in C$: *degree H (v)* $\geq k$, and *H* is the maximum subgraph with this property. Therefore, a *k*-core of G can be obtained by recursively removing all the vertices of degrees less than *k* until all vertices in the remaining graph have at least degree *k*.

3.4. Modularity and normalization cut:

Assume that network G = *(A, S)* is undirected and attributed to *n* nodes, where $A = [a_{ij}] \in R^{n*n}$ is the adjacency matrix. In this matrix $a_{ij} = 1$ if there is an edge between nodes *i* and *j*; otherwise, $a_{ij} = 0$. Here, $\beta_i = \sum_j a_{ij}$ is the degree of node *i*, and $m = \frac{1}{2}\sum_i \beta_i$ is the total number of network edges. $S = [s_{ij}] \in R^{n*n}$ is a similarity matrix in which s_{ij} is the cosine similarity value between the corresponding content vectors of nodes *i* and *j*. According to these explanations, the normalized cut and modularity models are defined as follows:

3.4.1. Modularity Model:

The modularity function Q was first introduced by Newman and Girvan in [51] and is widely recognized as one of the most prominent quality functions for community detection. Due to its effectiveness, optimizing Q-modularity has become a fundamental approach in community detection. Equation (1) formally defines this function for two communities:

$$\emptyset = \frac{1}{4m}\sum_{ij}\left(a_{ij} - \frac{\beta_i\beta_j}{2m}\right)(\psi_i\psi_j) \qquad (1)$$

Where ψ_i is equal to 1 (or -1) if node v_i Belongs to community 1 (or 2). Modularity can be easily optimized using specific vectors and values by defining a modularity matrix, as shown in equation (2):

$B = [b_{ij}] \in R^{n*n}$, with entries $b_{ij} = a_{ij} - \frac{\beta_i \beta_j}{2m}$ (2)

Therefore, the modularity $\emptyset$ can be rewritten as equation (3):

$$\emptyset = \frac{1}{4m}\psi^T B \psi \quad (3)$$

Where $\psi = [\psi_i] \in \{-1,1\}^n$represents membership in a community node. However, maximizing modularity is an NP-hard problem. By simplifying the problem and allowing variables ψ_i to take any integer value, the problem can be easily solved as equation (4):

$$max\, \emptyset = max\, Tr(\Psi^T B \Psi) \quad (4)$$

Where $\Psi = [\psi_{ij}] \in R^{n*p}$ is the matrix that hints at membership in the community, and *Tr (0)* is the trace function. The solution is to obtain *p* of the most significant specific vector of modularity matrix *B*. In addition, the solution space allows Ψ reconstruction of network topology from a community structure viewpoint. Therefore, any row of the Ψ matrix can be assumed to be a good representation of the corresponding node in the hidden space to detect the community.

3.4.2-Normalize cut model:

This model calculates the ratio of external edges to internal edges, providing a measure of community separation. To compute a normalized cut, the cut between clusters A and B, denoted as Cut (A, B), represents the total number of edges that connect nodes in different clusters. The volume of cluster AA, represented as Vol (A), is the sum of the degrees of all nodes within cluster A [52]. These values are determined using equations (5) and (6):

$$cut(A,B) = \sum_{i\in A, j\in B} w_{ij} \quad (5)$$

$$Vol(A) = \sum_{i\in A} k_i \quad (6)$$

Given equations (5) and (6), the objective function of the normalized cut for two clusters, *A* and *B*, will be equation (7) or equation (8) when there are *k* clusters $C_1, C_2 \ldots C_k$.

$$Ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)} \quad (7)$$

$$Ncut(C_1, C_2, \ldots, C_k) = \sum_{t=1}^{k} \frac{link(C_t, \bar{C_t})}{vol(C_t)} \quad (8)$$

Where $link(C_t, \bar{C_t}) = \frac{1}{2}\sum_{i\in C_t, j\in \overline{C_t}} S_{ij}$ is the total connection from nodes in C_t to all nodes in $\bar{C_t}$ (not inC_t) and $vol(C_t) = \sum_{i\in C_t} d_i$ is the total internal connection in C_t.

To achieve the minimum objective function, the normalized cut is wrapped in an optimization problem as per Equation (9), where L is the Laplacian graph matrix of similarity and its normalized form $D^{-1}L = D^{-1}(D-S) = I - D^{-1}S$ is the identity matrix (I). Equation (10) is known as the Markov matrix:

$$min\ \ Tr(\emptyset^T L \emptyset)$$

$$\emptyset \in R^{n*k}$$

S.t $L=D-S$

$$D= diag(d_1, d_2, \ldots, d_n) \quad (9)$$

$$\emptyset_{ij} = \begin{cases} \frac{1}{\sqrt{vol(C_j)}} & \text{if } v_i \in C_j \\ 0 & \text{otherwise} \end{cases}$$

$$M=D^{-1}S \quad (10)$$

In the case of this problem, the solution matrix $\emptyset$ of the specific vectors of *k* is the minimum nonzero particular value of the normalized Laplacian matrix $D^{-1}L$. In other words, *k* is the most significant specific value *M* covers, representing the solution in the hidden space. More importantly, the solution matrix Φ provides a perfect representation for obtaining the clustering.

Given the above, a higher modularity leads to a better partition structure; conversely, a lower normalized cut value enhances the two critical principles of graph classification, namely maximum integrity and minimum connection.

3.5. Graph embedding:

Let G= (*V, E, X*), where $V = \{v_i\}\ i = 1,2,..,n$ is formed of a set of graph nodes and $e_{ij} =< v_i, v_j > \in E$ represents a connection between the nodes. The topological structure of graph G is illustrated by adjacency matrix *A*, where $A_{ij} = 1$ if $e_{ij} \in E$ and otherwise $A_{ij} = 0$. $X \in R^{n*d}$ is the node attribute matrix, and *d* is the number of attributes. In addition, $x_i \in X$ shows the attributes of the content of each node v_i. The objective of the embedding problem is to map nodes $v_i \in V$ to low-dimensional vectors $\overrightarrow{z_i} \in R^d$, with a formal format $f: (A, X) \rightarrow Z$, where ${z_i}^T$ is the-*i* row of the $Z \in R^{n*d}$ matrix (*n* is the number of nodes, and *d* is the packing dimension). We assume that *Z* is the packing matrix, so the packings should preserve *A*'s topology and content information, *X*.

3.6. Notations:

Table 1 consolidates the essential symbols used throughout this paper, encompassing various matrices, graph properties, and representation details relevant to the discussed methods. This table serves as a reference for understanding the notations and mathematical formulations employed in our approach.

4. The proposed method: VGAEE

Our proposed model is designed to detect communities within attributed social networks by utilizing a parallel dual graph convolutional neural network (GCN) for an efficient and interpretable embedding process. The model is structured into four distinct phases:

1. **Graph Filtering**: This initial phase filters the graph to prepare it for further processing, enhancing the clarity of the underlying structures within the network.
2. **Modularity and Markov Matrices Calculation**: The second phase calculates modularity and Markov matrices, which are crucial for understanding the community structure and the transition probabilities between nodes.
3. **Network Embedding**: During the third phase, a Variational Graph AutoEncoder is employed to generate a new and meaningful representation of the network. This step is pivotal for capturing the essence of community structures in a lower-dimensional space.
4. **Clustering**: The final phase involves clustering the embedded representations to identify distinct communities within the network. This step categorizes nodes into groups based on the learned embeddings.

The output from each phase is meticulously designed to feed into the subsequent phase as input, ensuring a smooth transition and integration of data throughout the model. Fig. 1 provides a detailed schematic of the proposed method, visually outlining each phase and their interconnections. The upcoming sections will explore the intricacies and functionalities of each phase in greater detail, offering a comprehensive understanding of our approach.

4.1. Graph Filtering

By implementing the k-core algorithm, we strategically streamline the graph by removing nodes of lesser significance, typically those with low degrees. This method significantly reduces the graph's size and complexity, enhancing the efficiency of community detection algorithms applied thereafter. The k-core algorithm highlights the graph's most prominent regions, facilitating more focused and faster computations. Essentially, a k-core represents a maximal subset of a graph's nodes where each node maintains at least k connections within that subset. For inclusion in the k-core, a node's degree within the subset must be no less than k. The process involves calculating the k-core by first removing nodes with degrees less than k, then recalculating the degrees, and iteratively repeating this removal process until all nodes satisfy the k-core condition. Each iteration carries a computational complexity of O(E), where E denotes the total number of edges.

Through successive iterations, the graph is methodically reduced by excluding nodes lacking sufficient connectivity, ultimately yielding a simplified core that depicts the most interconnected nodes. As delineated in this section, the k-core algorithm inherently defines a community based on its density, thereby reducing the overall graph size—this accelerates the community detection process in subsequent phases and bolsters the community-centric focus of graph neural networks. The choice of k in this algorithm is contingent upon the specific dataset being analyzed; in this study, a k-value of 3 was selected based on a trial-and-error method to optimize the balance between simplification and structural integrity.

4.2. Calculation of the modularity matrix and normalized cut matrix

This section details the calculation of the modularity matrix (Matrix B) and the Markov matrix (Matrix M) for the filtered graph, a product of applying the 3-core algorithm. These calculations are fundamental for understanding the structural and transitional properties of the graph and are crucial for subsequent analyses, such as community detection or dynamic behavior studies.

Table 1: List of notations used in this paper

Symbols	Descriptions	Symbols	Descriptions
A	Graph adjacency matrix	S	A similarity matrix
X	Graph attribute matrix	B_{ij}	The modularity value of (v_i; v_j)
N	Number of nodes in the graph	Q	The modularity evaluation metric
Z	Representations of nodes	S_{ij}	The pairwise node similarity value of (v_i; v_j)
H	Hidden dimensions	D	A degree matrix
$\bar{A}$	Reconstructed graph adjacency matrix	L	A Laplacian matrix
K	Number of communities in the graph	B	A modularity matrix
$H^{[i+1]}$	Feature representation at layer i+1	M	A Markov matrix
$\sigma(0)$	The Activation function	H^i	Feature representation at layer i
W^i	Weight at layer i	b^i	Based on layer i

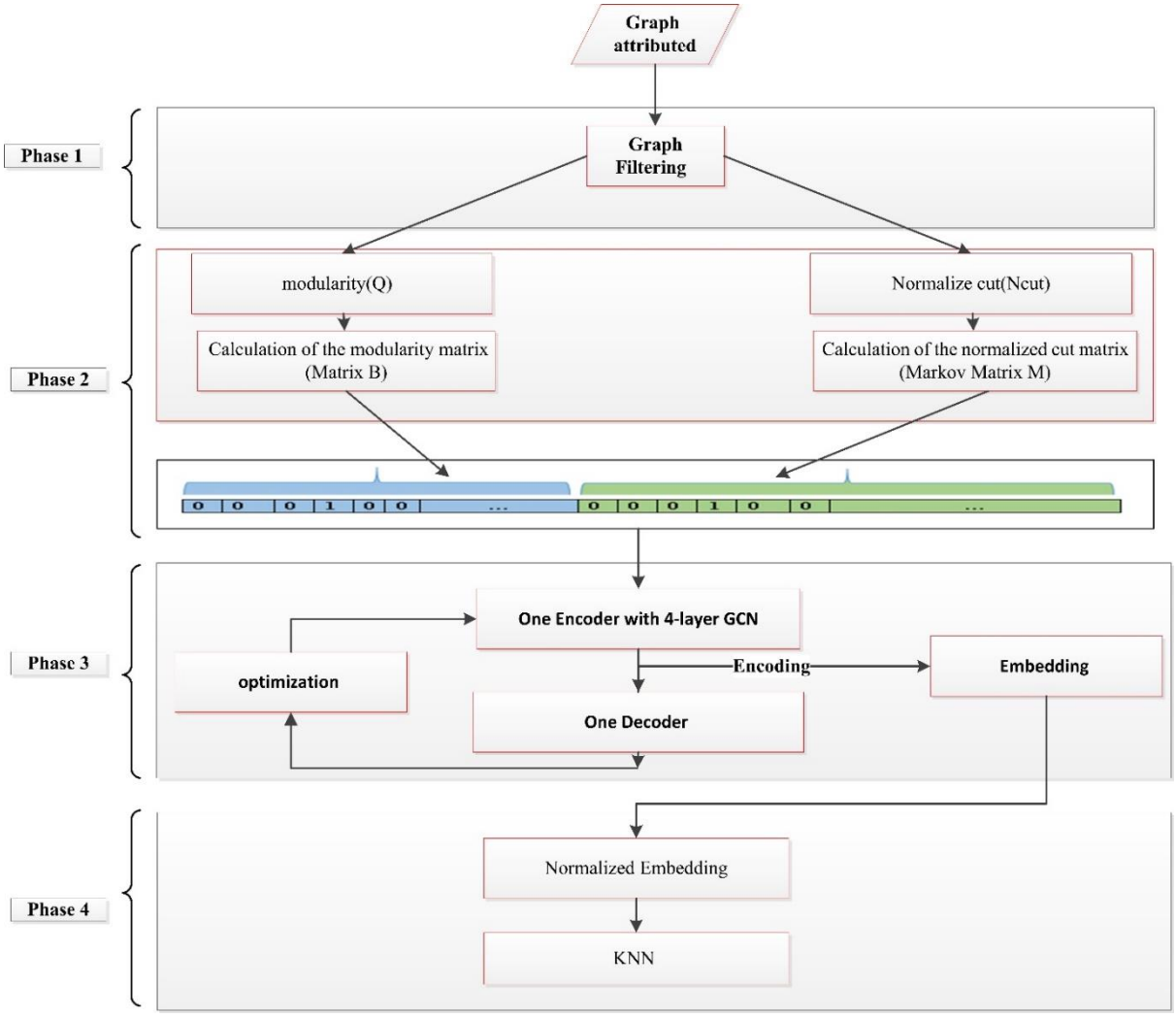

Fig. 1: Flowchart of the proposed method VGAEE

4.3. Network embedding

The learning phase aims to achieve a robust embedding of the data graph G= (V, E, A, X). To accomplish this, we employ a Variational Graph Autoencoder (VGA), which processes the entire graph to learn an effective embedding. As depicted in Figure 2, the workflow for this processing method involves two primary components: the encoder and the decoder.

Encoder: In a Variational Graph Autoencoder, the encoder's role is pivotal. It takes two inputs: the adjacency matrix A, representing the graph's structure, and the node features matrix X. The encoder's task is to map this high-dimensional input data into a lower-dimensional latent representation Z. This latent space Z captures the essential features of the nodes while preserving the structural and feature-based relationships inherent in the graph. Typically, the encoder uses layers of graph convolution to aggregate and transform the input data into this compact representation. This step is crucial as it determines how well the encoder can identify and encode community-specific features into the latent space.

Decoder: Following the encoding process, the decoder takes the latent representation Z and aims to reconstruct the original graph's structure. The primary objective of the decoder is to validate the effectiveness of the learned embeddings by attempting to regenerate the adjacency matrix A from Z. This process tests the encoder's ability to embed nodes in such a way that the original graph structure can be predicted from the embeddings. A successful reconstruction indicates that the latent space Z contains meaningful and comprehensive information about the graph's structure and node interactions.

The Variational Graph Autoencoder's effectiveness hinges on its ability to reduce the dimensionality of the graph data while retaining significant structural and feature-related information. This capability is crucial for tasks such as community detection, where the goal is to cluster similar nodes more effectively. By embedding nodes into a lower-dimensional space that emphasizes community-specific features, the Variational Graph Autoencoder facilitates more accurate and efficient community clustering. This method not only streamlines computations but also enhances the interpretability of the results, allowing for clearer insights into the underlying community structure of the graph.

4.3.1. Encoder Model

The encoder (inference model) of VGAE consists of graph convolutional networks (GCNs) [51]. It takes an adjacency matrix A and a feature matrix X as inputs and generates the latent variable Z as output. The first GCN layer transforms the feature matrix into a lower-dimensional form as defined by Equation 11:

$$\bar{X} = GCN(X, A) = ReLU(\widetilde{A}\, XW_0) \qquad (11)$$

$$\widetilde{A} = D^{-\frac{1}{2}} A\, D^{-\frac{1}{2}}$$

A-tilde is the symmetrically normalized adjacency matrix. The second GCN layer generates μ and $\log\sigma^2$, which are defined by Equation 12:

$$\mu = GCN_{\mu}(X, A) = \widetilde{A}\, \bar{X} W_1 \qquad (12)$$

$$\log\sigma^2 = GCN_{\sigma}(X, A) = \widetilde{A}\, \bar{X} W_1$$

Now if we combine the math of two-layer GCN as defined in Equation 13, yields:

$$GCN(X, A) = \widetilde{A}\, ReLU\big(\widetilde{A}\, XW_0\big) W_1 \qquad (13)$$

Which generates μ and $\log\sigma^2$. Subsequently, Z can be determined using the parameterization trick, as specified in Equation 14:

$$Z = \mu + \boldsymbol{\sigma} * \varepsilon \quad \text{Where } \varepsilon \sim N(0, 1). \qquad (14)$$

4.3.2. Decoder Model

The decoder (generative model) is defined by an inner product between latent variable Z. The output of our decoder is a reconstructed adjacency matrix A-hat, which is defined as Equation 15:

$$\hat{A} = \sigma(zz^T) \qquad (15)$$

Where σ(•) is the logistic sigmoid function. In summary, the encoder is represented as Equation 16:

$$q(z_i|X, A) = N(z_i|\mu_i, diag(\sigma^2)) \qquad (16)$$

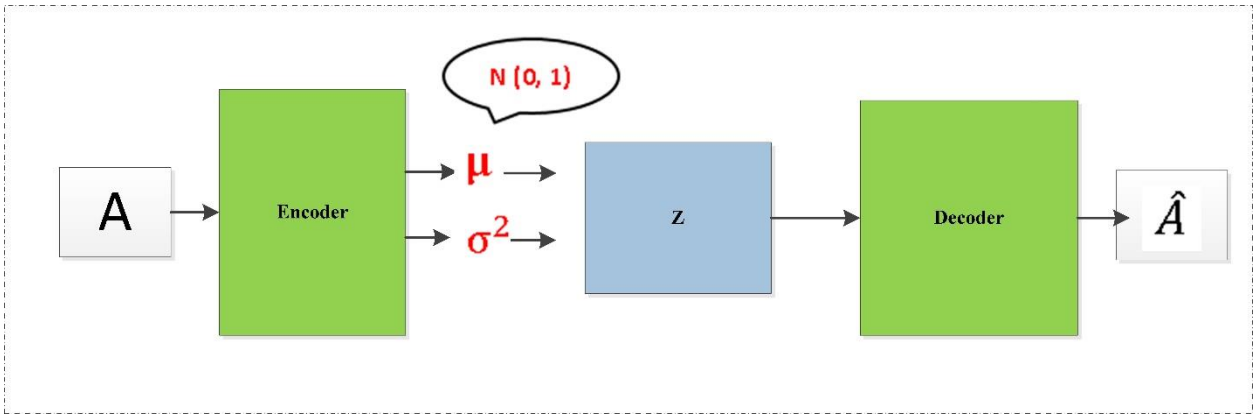

Fig. 2: The workflow scheme of the Variational graph autoencoder in the proposed method

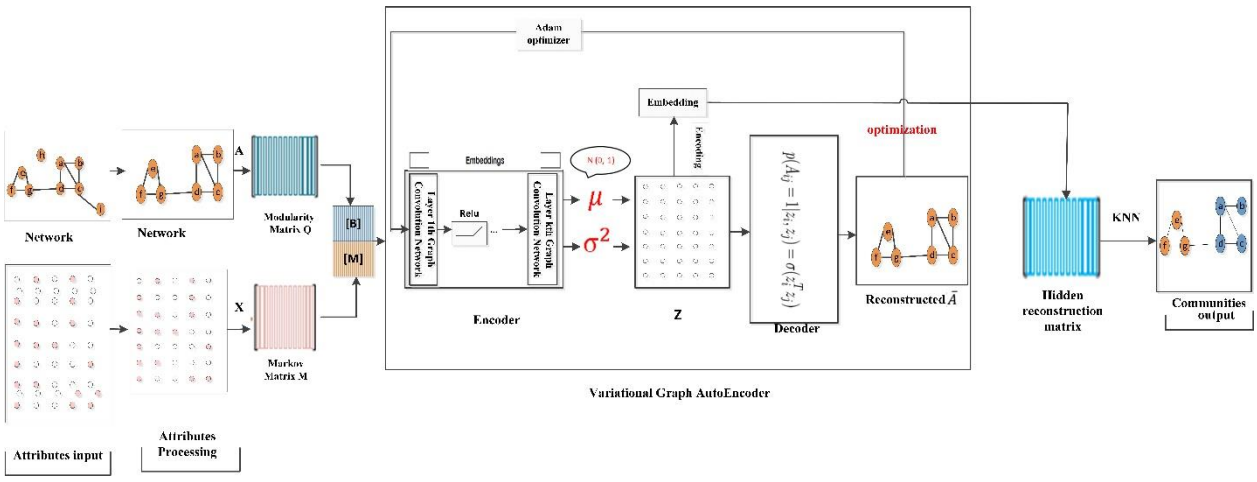

Fig. 3: The VGAEE Framework for Community Detection in Attributed Social Networks.

The decoder is represented in Equation 17:

$$p(A_{ij} = 1|z_i, z_j) = \sigma(z_i{}^T z_j) \qquad (17)$$

In this paper, the encoder, a linear combination of the matrices Q and M is initially computed, which can be considered as the new input feature matrix X_{new}:

$$Xnew = \alpha Q + \beta M \qquad (18)$$

Where α and β are coefficients for the combination. This X_{new} is then fed into Graph Convolutional Networks (GCN): The first GCN layer produces a lower-dimensional feature representation:

$$X' = GCN\,1(X_{new}, A) = ReLU\,(AX_{new}W_0)$$

where A is the symmetrically normalized adjacency matrix.

The second GCN layer generates the values μ and

$$\log\sigma^2 : \mu = GCN_{\mu}(X', A) = \tilde{A}X\,W_1$$

$$log\sigma^2 = GCN_{\sigma}(X, A) = \tilde{A}\,\bar{X}\;W_1$$

The decoder then uses these parameters to reconstruct the adjacency matrix:

$A' = sigmoid\,(\tilde{A}X\,W_2)$ Where W are the weights associated with the decoder. Using the reparameterization trick: $Z = \mu + \sigma\,\dot{o} \sim N\,(0,1)$ is a random variable from the standard normal distribution. These adjustments ensure that the combined inputs are accurately reflected in the model, allowing for more precise and complex community structure identification.

4.3.3. Loss function and Optimization

The loss function for the Variational Graph Autoencoder remains largely unchanged and is defined in Equation 18. It comprises primarily of the reconstruction loss between the input adjacency matrix and the reconstructed adjacency matrix.

More specifically, this involves the binary cross-entropy between the target (A) and the output (A′) logits. The second part is the KL divergence between q(Z | X, A) and p (Z), where p (Z) = N (0, 1). It measures how closely our q(Z | X, A) matches p (Z).

After we get the latent variable Z, we want to find a way to learn the similarity of each row in the latent variable (because one row represents one vertex) to generate the output adjacency matrix. The inner product could calculate the cosine similarity of two vectors, which is useful when we want a distance measure that is invariant to the magnitude of the vectors. Therefore, by applying the inner product on the latent variable Z and Z^T, we can learn the similarity of each node inside Z to predict our adjacency matrix.

$$L = E_{q(Z|X,A)}[logp(A|Z)] - KL[q(Z|X,A)||p(Z)]$$

The proposed decoding model is used to reconstruct graph data. We can reconstruct a graph structure, content information X, or both. Here, reconstruction of the graph structure is recommended, which gives us a higher level of flexibility so our algorithm preserves its functionality even if content information X is unavailable. Decoder $p(\hat{A}|Z)$ predicts whether there is a connection between the two nodes of a connection. Specifically, we trained a connection prediction layer based on graph embedding as per Equation 19 and Equation 20.

$$p(\hat{A}|Z) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\hat{A}_{ij}|z_i, z_j) \quad (19)$$

$$p(\hat{A}_{ij} = 1|z_i, z_j) = sigmod({z_i}^T, z_j) \quad (20)$$

The embedding of Z and $\hat{A}$ Reconstructed graphs are given in Equation 21:

$$\hat{A} = sigmod(ZZ^T), here\ Z = q(Z|X,A) \quad (21)$$

The graph data reconstruction error for a self-encoder graph is minimized using Equation 22.

$$\mathcal{L}_0 = E_{q(Z|(x,A))}[log_p(\hat{A}|Z) \quad (22)$$

4.4. Node clustering:

In this phase of processing, min-max scaling is applied to normalize the Z_final feature vectors that were obtained in the previous phase. This normalization technique adjusts the data values so that they range between zero and one. The objective of using min-max scaling in this context is to standardize the range of the feature vectors, thus ensuring that no single feature dominates due to its scale. This uniform scaling across all features is essential for several reasons:

1. Enhanced Algorithm Performance: Uniformity in feature scale helps machine learning algorithms converge more quickly. This is particularly important for algorithms like K-nearest neighbors (KNN), which rely on distance calculations between points. If the scales are not uniform, features with larger ranges could disproportionately influence the outcome, leading to biased results.
2. Improved Stability: Algorithms that depend on distance measurements or gradients are less likely to exhibit erratic behavior during learning when all features contribute equally. Stability in algorithm performance leads to more reliable and reproducible results.
3. Optimized Learning Efficiency: When all features are scaled uniformly, each feature has an equal opportunity to influence the learning process, potentially increasing the efficiency and effectiveness of the model.

Applying min-max scaling to the Z_final feature vectors ensures that the subsequent steps, especially those involving algorithms like KNN for clustering or classification, operate under optimal conditions. This preprocessing step is crucial for achieving accurate and efficient outcomes in the analysis of data, particularly in complex machine-learning tasks that involve large and diverse datasets. The decoder is represented in Equation 17:

$$p(A_{ij} = 1|z_i, z_j) = \sigma({z_i}^T z_j) \quad (17)$$

Fig. 4 illustrates the architecture of our proposed community detection model using VGAEE.

5. Experiment

In this section, we describe the comprehensive experiments conducted to evaluate the performance of the Variational Graph Autoencoder Embedding Enhancer (VGAEE) against state-of-the-art methods in real-world scenarios using valid datasets. These experiments are designed to provide a fair and rigorous comparison, focusing on several critical aspects:

5.1. Experimental settings

5.1.1. Datasets

In our study, we utilized datasets derived from real-world applications to test our community detection methods, ensuring a thorough evaluation. Statistical information about the three datasets employed is presented in Table 4, reference [57]. These datasets comprise citation networks where the nodes symbolize papers and the edges denote the citations between them. Each node is associated with attributes that represent word packet summaries of the paper abstracts, while the labels indicate the topics of the papers.

5.1.2. Evaluation metrics

This section presents various qualitative metrics for evaluating community detection approaches, classified into performance and goodness measures. Performance measures assess the quality of the communities identified by the algorithm relative to real-world communities. Additionally, goodness measures focus on the structural characteristics of the communities that have been detected [60]. Our evaluation of the proposed method utilized two key metrics: normalized mutual information and accuracy. Higher values in these metrics signify better performance. Subsequent sections will provide a detailed discussion of these measures.

-Normalized Mutual Information

The normalized mutual information, calculated using equation (26), measures the similarity between the community set identified by the proposed algorithm and the actual community [60].

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij} ln(n_{ij}.n/n_i^c.n_j^c)}{\sqrt{(\sum_{i=1}^{k} n_i^c \, ln\left(\frac{n_i^c}{n}\right))(\sum_{j=1}^{k} n_j^c \, ln(n_j^c/n))}} \quad (26)$$

Where k is the number of communities, n is the number of nodes, n_{ij} is the number of nodes in the optimized community set i such that the proposed community set is in community j, n_i^c is the number of nodes in the community i, which is in the optimized community set, and n_j^c is the number of nodes in community j.

-Accuracy

It assesses the authenticity of the community structure. Similar to NMI, computing this measure necessitates the use of an optimal community setting, as outlined in equation (27) [60].

$$ACC = \frac{\sum_{i=1}^{n} k(C_i, PM(\acute{C}_i))}{n} \quad (27)$$

Where n is the number of groups, and for a specific group, i and C_i ،$\acute{C}_i$ are the communities of node i in optimum and recommended community settings. $K(x, y)$ is a function equal to 1 when $x=y$ and 0 otherwise.

5.1.3. Parameter Settings:

For our study, we structured the training set by selecting 20 nodes from each class, resulting in a total of 500 nodes for the validation set and 1,000 nodes for the test set. Our experiments were conducted using a two-layer Graph Convolutional Network (GCN) setup. The initial layer included 64 neurons, with each subsequent layer in the contracting path halving the neuron count from the previous layer. The training was facilitated using the Adam optimizer, a popular choice due to its efficiency, and the experiments were carried out using both TensorFlow and PyTorch frameworks.

The learning rate was initially set at 0.01, adjusted dynamically by a scheduler that reduced the rate upon encountering a loss plateau, which helped achieve more stable convergence. We implemented a dropout rate of 0.5 to prevent overfitting and capped the training at a maximum of 200 epochs. The Relu activation function was applied following each graph convolutional operation. Training was halted if there was no decrease in the loss function over 10 consecutive epochs.

Initialization of the initial weights for the two GCN layers was done randomly, selected from a uniform distribution. To ensure the robustness of our results, each experiment was repeated ten times, with the average scores reported subsequently. Detailed parameter settings for these experiments are summarized in Table 5, which includes parameter names and their respective values.

5.1.4. Experimental results and analysis

This subsection presents the experimental results analyzed from various evaluation angles to validate the efficiency of our proposed model. We conducted experiments using medium-scale datasets including Cora, Citeseer, and PubMed, and compared our model against three established baseline categories to provide a thorough analysis. The comparison categories are detailed as follows:

1. **Node Feature-Based Methods**: This category focuses on the unique attributes

or characteristics of individual nodes. Methods such as k-means and spectral clustering, referred to here as spectral_f, are prominent in this category. These methods construct a similarity matrix primarily using a linear kernel based on node features.

2. **Graph Structure-Based Methods**: This category emphasizes the intrinsic structure of the graph. Techniques like spectral clustering (Spectral_g) utilize the node adjacency matrix to build the similarity matrix. Notable methods in this group include DeepWalk [14], which excels in learning graph embeddings, and DNGR [62], which merges spectral graph clustering with deep neural networks for complex graph representation. Additionally, vGraph [63] is a probabilistic generative model that learns community membership and node representation collaboratively, while Graph Encoder [64] focuses on learning graph embedding for spectral graph clustering.
3. **Hybrid Methods**: These methods integrate both node attributes and graph structure, typically resulting in enhanced community detection outcomes despite increased computational complexity. Various graph autoencoder variants fall within this category, including:
    - **GAE [65]**: Utilizes neural networks for learning graph representations.
    - **VGAE [65]**: Advances GAE by implementing a Variational inference framework.
    - **MGAE [18]**: Enhances representation by marginalizing specific graph properties.
    - **ARGA [66]** and **ARVGA [66]**: Employ adversarial and vibrational regularization, respectively, to refine graph embeddings.
    - **DAEGC [67]**: Uses deep autoencoders to reconstruct the graph's adjacency matrix.
    - **AGE [56]**: Enhances graph-based learning tasks through a two-stage process.
    - **AGC [55]**: Leverages high-order graph convolution to effectively understand a graph's global structure.
    - **DBGAN [68]** and **GALA [69]**: New approaches using graph neural networks for clustering and embedding node features.
    - **CommDGI [11]** and **GC-VGE [70]**: Optimize the simultaneous learning of node embeddings and cluster assignments.
    - **TADW [71]**: Employs matrix factorization for network representation learning.
    - **RMSC [72]** and **RTM [72]**: Focus on robust multi-view spectral clustering and learning topic distributions from text and citations, respectively.
    - **GMIM [73]**: Utilizes a mutual information maximization approach for node embedding.
    - **DGVAE [74]**: Introduces a graph Variational generative model with Dirichlet distributions as priors on latent variables.
    - **BernNet GCN [75]** and **WC-GCN [76]**: Utilize graph convolutional network frameworks, with the former based on Bernstein polynomial approximation.
    - **LGNN [35]** and **MRFasGCN [27]**: Specialized neural network models for graph data, with MRFasGCN combining GCN with a Markov random field model for community detection.

These methods provide a broad spectrum of approaches for analyzing and detecting community structures within networks, facilitating a comprehensive comparison against our proposed model.

Tables 6-8 comprehensively compare the proposed method with baseline community detection methods based on their performance metrics. These metrics include accuracy (ACC %) and normalized mutual information (NMI %). The compared approaches are often categorized into three groups based on the type of learning: supervised, semi-supervised, and unsupervised. Furthermore, these strategies are classified into three groups based on the input type: Features, graph topology, or a hybrid of both.

Table 6 presents a comprehensive comparison of various graph-based learning methods used for community detection in the Cora dataset, highlighting their performance in terms of accuracy (ACC %) and normalized mutual information (NMI %). Among the methods listed, the proposed VGAEE stands out with the highest performance metrics, achieving an ACC% of 84.5 and an NMI% of 70.46. This represents a significant improvement over both supervised and unsupervised approaches. For instance, the closest competitors, MRFasGCN

and AGE, which are also unsupervised, recorded ACC% of 84.3 and 76.8 and NMI% of 66.2 and 60.7, respectively. VGAEE's superior performance suggests that its methodology for integrating graph topology in an unsupervised learning framework effectively captures the nuanced structures within the community more accurately than other methods. Furthermore, the results from VGAEE are particularly notable when compared to supervised methods such as LGNN and WC-GCN, which, despite their structured learning paradigms, do not achieve the same level of ACC or NMI. Overall, the data underscores the efficacy of VGAEE in community detection, setting a new benchmark for future studies in this area.

To make a fair comparison with other related works, we repeated the experiments on two different datasets, the PubMed dataset and the Citeseer dataset. We present the results and figures of this new evaluation in Tables 7 and 8, respectively.

In Table 7, the proposed VGAEE method outshines both unsupervised and supervised learning algorithms for the PubMed dataset, registering an ACC% of 80.50 and an NMI% of 55.60. This significantly distances it from traditional unsupervised methods like K-means, Spectral-F, and Spectral-G, which show considerable variability in their results. When comparing VGAEE with other advanced graph-based methods, it still maintains a leading position. For example, the semi-supervised MRFasGCN achieves a higher NMI% at 40.7 but falls short in ACC%, illustrating that while it effectively captures mutual information within the data, it does not necessarily translate to outright accuracy. Similarly, the supervised BernNet GCN scores an impressive NMI% of 51.40 but with a lower ACC% of 61.25, indicating potential overfitting to mutual information at the cost of general accuracy. Among unsupervised competitors, AGE and GMIM perform well, with AGE reaching an ACC% of 71.1 and GMIM peaking at 70.87, yet neither approaches the combined performance metrics of VGAEE. Additionally, methods like AGC and CommDGI, while competitive, do not achieve the same balance between ACC and NMI, suggesting that VGAEE's method of integrating features and graph topology potentially offers a more robust model for understanding complex network structures. Overall, the superiority of VGAEE in this dataset underscores its effectiveness in handling the nuances of community detection in large, complex networks. Its ability to outperform existing algorithms, particularly in unsupervised settings, sets a new benchmark and indicates promising directions for future research and application in social network analysis and beyond.

Based on the analysis presented in Table 8, the table showcases the performance of the VGAEE method relative to other community detection algorithms across various learning paradigms for the Citeseer dataset. VGAEE, an unsupervised method, stands out with an ACC% of 75.60 and an NMI% of 57.06. Notably, VGAEE surpasses popular unsupervised algorithms like K-means, Spectral-F, and DeepWalk, which present considerably lower metrics in both accuracy and mutual information. Even when compared to the semi-supervised MRFasGCN and supervised methods such as BernNet GCN and WC-GCN, VGAEE demonstrates competitive or superior performance, particularly in accuracy. This highlights VGAEE's efficacy in effectively capturing and preserving the intrinsic community structures in complex networks without requiring labeled data. Positioned as a robust tool in the unsupervised learning landscape for graph-based community detection, VGAEE excels in handling unlabeled and complex datasets while maintaining a balance between accuracy and information preservation.

The proposed VGAEE method demonstrated outstanding results across all three datasets: Cora, PubMed, and Citeseer, with its performance being particularly notable on the Citeseer dataset. On Citeseer, it achieved the highest accuracy and NMI percentages among all methods evaluated, with scores of 75.60% and 57.06% respectively. While it also ranked among the top performers on the Cora and PubMed datasets, with accuracies of 84.5% and 80.5% respectively, the Citeseer results highlight its superior capability in community detection within various network analyses. This underscores the VGAEE method's robust adaptability and effectiveness across diverse and complex datasets, marking it as a potent tool for intricate network analysis tasks. Figures 4, 5, and 6 illustrate the performance of the proposed method on the Cora, PubMed, and Citeseer datasets, respectively, based on the ACC (classification accuracy) and NMI (normalized mutual information) metrics, compared to baseline methods. In all three figures, the ACC and NMI values for the proposed method are highlighted in bold above the corresponding bars to clearly demonstrate its superiority over other methods.

Table 4: Summary of real-world benchmarks on datasets.

Dataset	#Nodes	#Edges	#Node Attributes	Num. of Communities
Cora [58]	2,708	5,429	1,433	**7**
Citeseer [58]	3,312	4,715	3,703	**6**
PubMed [59]	19,717	44,338	500	**3**

Table 5: Detailed parameter setting

Datasets	Training Epoch	Learning rate	Activation Function	Weight Decay	Optimizer	GCN layers	Dropout rate	#Train/Validation /Test Node
Cora	200	0.01	Relu	5e-3	Adam	64/32	0.5	140/500/1000
Citeseer	200	0.01	Relu	5e-3	Adam	64/32	0.5	120/500/1000
PubMed	200	0.01	Relu	5e-3	Adam	64/32	0.5	60/500/1000

Table 6: Performance comparison of different community detection methods on the Cora dataset; the best results are in bold.

Name of methods	Learning type	Input	ACC%	NMI%
K-means	Unsupervised	Feature	49.2	**32.1**
Spectral-F [77]	Unsupervised	Feature	34.7	**14.7**
Spectral-G [77]	Unsupervised	Graph	31.46	**9.69**
DeepWalk [14]	Unsupervised	Graph	56.20	**39.87**
Graph Encoder [78]	Unsupervised	Graph	32.5	**10.9**
vGraph[63]	Unsupervised	Graph	28.7	**34.5**
TADW [71]	Unsupervised	Feature & Graph	55.00	**36.59**
VGAE [65]	Unsupervised	Feature & Graph	63.56	**47.45**
MGAE [18]	Unsupervised	Feature & Graph	63.43	**45.57**
ARGE [66]	Unsupervised	Feature & Graph	60.84	**42.21**
ARVGA [66]	Unsupervised	Feature & Graph	62.83	**45.93**
DGVAE [74]	Unsupervised	Feature & Graph	64.42	**47.64**
AGC [55]	Unsupervised	Feature & Graph	68.92	**53.68**
CommDGI [11]	Unsupervised	Feature & Graph	69.8	**57.9**
DAEGC [67]	Unsupervised	Feature & Graph	70.4	**52.8**
GC-VGE [70]	Unsupervised	Feature & Graph	70.67	**53.57**
GALA [69]	Unsupervised	Feature & Graph	72.42	**53.96**
DBGAN [68]	Unsupervised	Feature & Graph	74.6	**57.7**
GMIM [73]	Unsupervised	Feature & Graph	74.8	**56.0**
AGE[56]	Unsupervised	Feature & Graph	76.8	**60.7**
MRFasGCN[27]	Semi-supervised	Feature & Graph	84.3	**66.2**
BernNet GCN[75]	Supervised	Feature & Graph	41.06	**68.78**
LGNN[35]	Supervised	Feature & Graph	79.04	-
WC-GCN[76]	Supervised	Feature & Graph	79.39	-
VGAEE(proposed method)	**Unsupervised**	**Feature & Graph**	**84.5**	**70.46**

Table 7: Performance comparison of different community detection methods on the PubMed dataset; the best results are in bold.

Name of methods	Learning type	Input	ACC%	NMI%
K-means	Unsupervised	Feature	55.59	24.34
Spectral-F [77]	Unsupervised	Feature	60.20	30.90
Spectral-G [77]	Unsupervised	Graph	37.98	10.30
DeepWalk [14]	Unsupervised	Graph	64.98	26.44
Graph Encoder[11]	Unsupervised	Graph	53.1	20.9
DNGR [62]	Unsupervised	Graph	25.53	20.11
vGraph [79]	Unsupervised	Graph	26.00	22.40
TADW [71]	Unsupervised	Feature & Graph	46.82	9.47
GAE [65]	Unsupervised	Feature & Graph	64.43	24.85
VGAE [65]	Unsupervised	Feature & Graph	64.67	23.94
MGAE [18]	Unsupervised	Feature & Graph	43.88	8.16
ARGA [66]	Unsupervised	Feature & Graph	65.07	29.23
ARVGA [66]	Unsupervised	Feature & Graph	62.01	26.62
DGVAE [74]	Unsupervised	Feature & Graph	67.56	28.72
AGC [55]	Unsupervised	Feature & Graph	69.78	31.59
CommDGI [11]	Unsupervised	Feature & Graph	69.90	35.70
DAEGC [67]	Unsupervised	Feature & Graph	67.10	26.60
GC-VGE [70]	Unsupervised	Feature & Graph	68.18	29.70
GALA [69]	Unsupervised	Feature & Graph	69.39	32.73
DBGAN [68]	Unsupervised	Feature & Graph	69.40	32.40
GMIM [73]	Unsupervised	Feature & Graph	70.87	32.43
AGE[56]	Unsupervised	Feature & Graph	71.1	31.6
MRFasGCN[27]	Semi-supervised	Feature & Graph	79.6	40.7
BernNet GCN[75]	Supervised	Feature & Graph	61.25	51.40
LGNN[35]	Supervised	Feature & Graph	72.64	-
WC-GC[76]	Supervised	Feature & Graph	79.41	-
VGAEE(proposed method)	Unsupervised	Feature & Graph	**80.50**	**55.60**

Table 8: Performance comparison of different community detection methods on the Citeseer dataset. The best results are in bold.

Name of methods	Learning type	Input	ACC%	NMI%
K-means	Unsupervised	Feature	54.0	30.5
Spectral-F [77]	Unsupervised	Feature	23.9	5.6
DeepWalk [14]	Unsupervised	Graph	32.7	8.8
Graph Encoder[11]	Unsupervised	Graph	22.5	3.3
DNGR [62]	Unsupervised	Graph	32.6	18.0
RTM [72]	Unsupervised	Graph	45.1	23.9
RMSC [72]	Unsupervised	Graph	29.5	13.9
TADW [71]	Unsupervised	Feature & Graph	45.5	29.1
GAE [65]	Unsupervised	Feature & Graph	40.8	17.6
VGAE [65]	Unsupervised	Feature & Graph	34.4	15.6
MGAE [18]	Unsupervised	Feature & Graph	43.88	8.16
ARGA [66]	Unsupervised	Feature & Graph	57.3	35.0
ARVGA [66]	Unsupervised	Feature & Graph	54.4	26.1

AGE[56]	Unsupervised	Feature & Graph	70.2	44.8
MRFasGCN[27]	Semi-supervised	Feature & Graph	73.2	46.3
BernNet GCN[75]	Supervised	Feature & Graph	72.32	58.01
LGNN[35]	Supervised	Feature & Graph	73.15	-
	Supervised	Feature & Graph	73.2	46.3
WC-GCN[76]	Supervised	Feature & Graph	75.18	-
VGAEE (proposed method)	**Unsupervised**	**Feature & Graph**	**75.60**	**57.06**

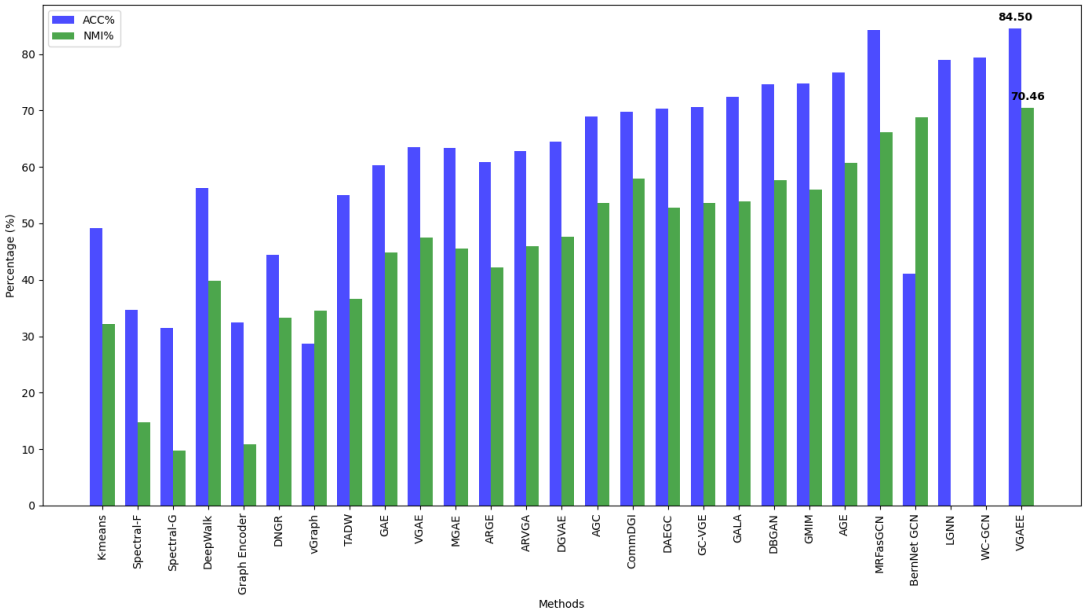

Fig. 4. Performance comparison of different community detection methods on the Cora dataset

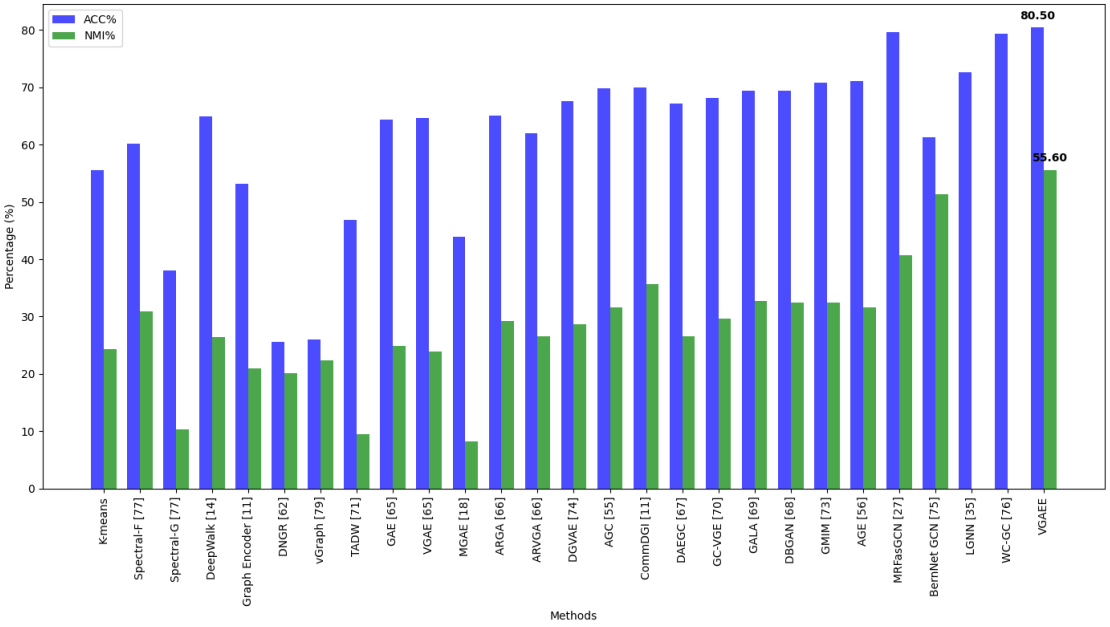

Fig. 5. Performance comparison of different community detection methods on the PubMed dataset

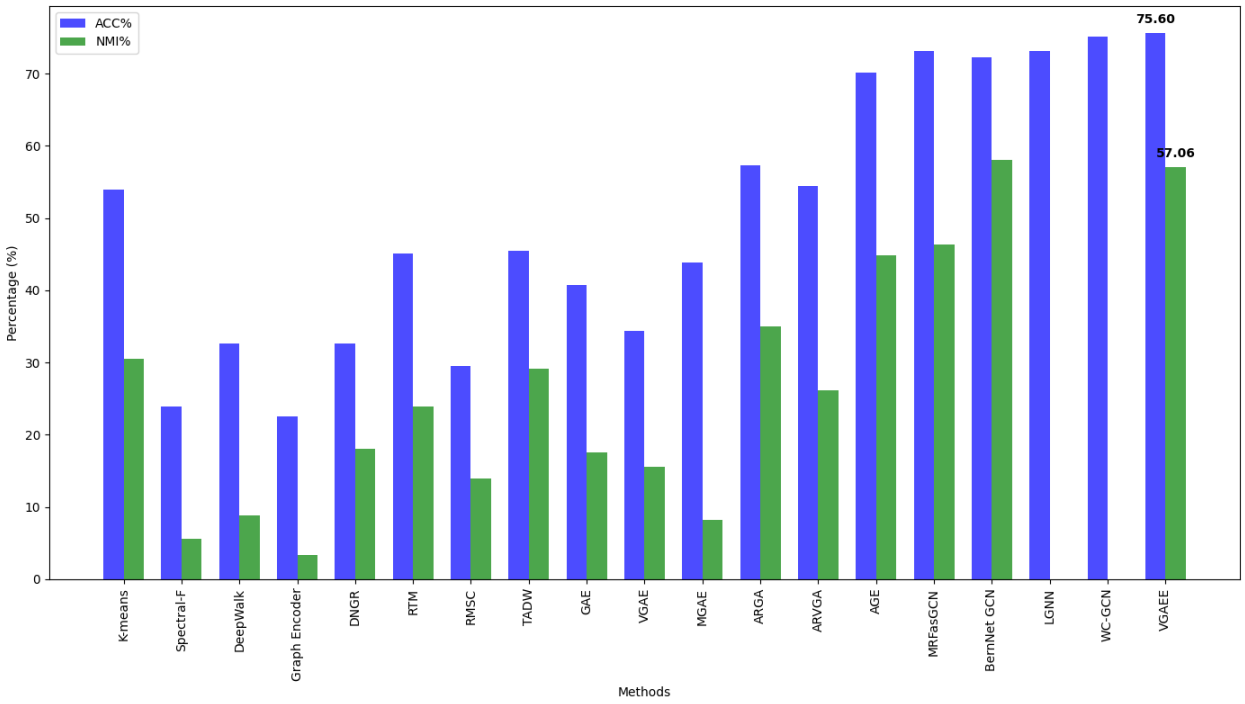

Fig. 6. Performance comparison of different community detection methods on the Citeseer dataset

6. Conclusion and future work

This study introduced VGAEE, an innovative unsupervised approach leveraging Variational Graph AutoEncoders to enhance community detection in attributed social networks. By integrating node content with network topology, VGAEE effectively captures complex community structures, achieving superior performance metrics across diverse datasets like Cora, Citeseer, and PubMed. Notably, VGAEE consistently outperformed both traditional and state-of-the-art methods, demonstrating its robustness and efficiency in handling large-scale network data without the necessity for pre-labeled information. The effectiveness of VGAEE was particularly evident in its ability to maintain high accuracy and mutual information scores, thereby providing a more nuanced understanding of community dynamics within large and complex networks. Looking forward, several avenues could further refine and expand the capabilities of the VGAEE framework. First, exploring the integration of semi-supervised learning protocols could potentially enhance the model's accuracy and applicability to even broader network types, including those with sparse or incomplete labeling. Additionally, adapting the model to dynamically evolving networks where community structures change over time would significantly increase its practical utility in real-world scenarios. Another promising direction involves enhancing the model's scalability and efficiency through the incorporation of more advanced graph neural network architectures or optimization techniques. Lastly, applying the VGAEE framework to other types of data, such as multimodal networks or those with highly heterogeneous attributes, could open new research areas and applications, further cementing its utility and impact in network analysis and beyond.

References

1. Wen, X., et al. (2016). *A maximal clique based multiobjective evolutionary algorithm for overlapping community detection.* IEEE Transactions on Evolutionary Computation. 21(3): p. 363-377. doi.org/10.1109/TEVC.2016.2622695
2. Lu, X., et al. (2018). *Adaptive modularity maximization via edge weighting scheme.* Information Sciences. 424: p. 55-68 . doi.org/10.1016/j.ins.2017.09.040
3. Wu, W., et al. (2018). *Nonnegative matrix factorization with mixed hypergraph regularization for community detection.* Information Sciences. 435: p. 263-281. doi.org/10.1016/j.ins.2017.12.017
4. Altinoz, O.T., K. Deb, and A.E. Yilmaz. (2018). *Evaluation of the migrated solutions for distributing reference point-based multi-objective optimization algorithms.* Information Sciences. 467: p. 750-765. *doi.org/10.1016/j.ins.2018.07.062*
5. Whang, J.J., D.F. Gleich, and I.S. Dhillon. (2016). *Overlapping community detection using*

neighborhood-inflated seed expansion. IEEE Transactions on Knowledge and Data Engineering. 28(5): p. 1272-1284. doi.org/10.1109/TKDE.2016.2528240

6. Fortunato, S. and D. Hric. (2016).Community detection in networks: A user guide. Physics reports. 2016. 659: p. 1-44. doi.org/10.1016/j.physrep.2016.09.002
7. Garza, S.E. and S.E. Schaeffer. (2019). *Community detection with the label propagation algorithm: a survey.* Physica A: Statistical Mechanics and its Applications. 534: p. 122058. *doi.org/10.1016/j.physa.2019.122058***.**
8. Cao, J., et al.(2018). *Incorporating network structure with node contents for community detection on large networks using deep learning.* Neurocomputing. 297: p. 71-81. *doi.org/10.1016/j.neucom.2018.02.072*
9. He, C., et al.(2019). *Community detection method based on robust semi-supervised nonnegative matrix factorization.* Physica A: Statistical Mechanics and its Applications. 523: p. 279-291. *doi.org/10.1016/j.physa.2019.02.010*
10. Zhengdao Chen, X.L., Joan Bruna. (2020). *Supervised Community Detection with Line Graph Neural Networks.* International Conference on Learning Representations. p. 1-24.openreview.net/forum?id=H1g0ZpA9FQ
11. Zhang, T., et al., *CommDGI: Community Detection Oriented Deep Graph Infomax*. 2020. p. 1843-1852. *doi.org/10.1145/3340531.3412042*
12. Tang, J., et al. (2015). *Line: Large-scale information network embedding*. *the 24th international conference on world wide web*. *doi.org/10.1145/2736277.2741093*
13. Grover, A. and J. Leskovec. (2016). *node2vec: Scalable Feature Learning for Networks*. p.855-864. doi.org/10.1145/2939672.2939754
14. Perozzi, B., R. Al-Rfou, and S. Skiena. (2014). *Deepwalk: Online learning of social representations*. in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ***doi:10.1145/2623330.2623732***
15. Chen, S. and W. Guo. (2023). *Auto-encoders in deep learning—a review with new perspectives.* Mathematics. 11(8): p. 1777. *doi:10.3390/math11081777*
16. Zhao, S., et al. (2021). *Hierarchical representation learning for attributed networks.* IEEE Transactions on Knowledge and Data Engineering. 35(3): p. 2641-2656. *doi:10.1109/TKDE.2021.3111539*
17. Lu, H.-Y., et al. (2024). *Visual analytics of multivariate networks with representation learning and composite variable construction.* IEEE Transactions on Visualization and Computer Graphics. *doi:10.1109/TVCG.2024.3372078*
18. Wang, C., et al. (2017). *Mgae: Marginalized graph autoencoder for graph clustering*. *Conference on Information and Knowledge Management*. *doi:10.1145/3132847.3132967*
19. Li, B., et al. (2020). *Multi-source information fusion based heterogeneous network embedding.* Information Sciences. 534: p. 53-71. *doi:10.1016/j.ins.2020.05.017*
20. He, C ,.et al. (2021). *Boosting nonnegative matrix factorization based community detection with graph attention auto-encoder.* IEEE Transactions on Big Data. 8(4): p. 968-981. *doi:10.1109/TBDATA.2021.3074253*
21. Yang, C., et al. (2021). *Network Embedding for Graphs with Node Attributes*, in *Network Embedding: Theories, Methods, and Applications*. p. *29-38*. *doi:10.1007/978-981-16-2637-9_3*
22. Zhang, Y., et al. (2022). *Spectral–spatial feature extraction with dual graph autoencoder for hyperspectral image clustering.* IEEE Transactions on Circuits and Systems for Video Technology. 32(12): p. 8500-8511. *doi:10.1109/TCSVT.2022.3171421*
23. Jin, D., et al. (2021). *A survey of community detection approaches: From statistical modeling to deep learning*. IEEE Transactions on Knowledge and Data Engineering. 35(2): p. 1149-1170. *doi:10.1109/TKDE.2021.3124888*
24. Liu, F., et al. (2020). *Deep learning for community detection: progress, challenges and opportunities.* arXiv preprint arXiv:2005.08225. *doi:10.48550/arXiv.2005.08225*
25. Zhou, J., et al. (2020). *Graph neural networks: A review of methods and applications.* AI Open. p. 57-81. *doi:10.1016/j.aiopen.2021.01.001*
26. Su, X., et al. (2022). *A comprehensive survey on community detection with deep learning.* IEEE Transactions on Neural Networks and Learning Systems. *doi:10.1109/TNNLS.2022.3145142*
27. Jin, D., et al. (2019). *Graph Convolutional Networks Meet Markov Random Fields: Semi-Supervised Community Detection in Attribute Networks.* the AAAI Conference on Artificial Intelligence. 33(01): p. 152-159. *doi:10.1609/aaai.v33i01.3301152*
28. Sun, H., et al. (2020). *Network embedding for community detection in attributed networks.* ACM Transactions on Knowledge Discovery from Data (TKDD). 14(3): p. 1-25. *doi:10.1145/3385414*
29. Jin ,D., et al. (2019). *Community detection via joint graph convolutional network embedding in attribute network*. in *International Conference on Artificial Neural Networks*. *doi:10.1007/978-3-030-30493-5_42*
30. Luo, J. and Y. Du. (2020). *Detecting community structure and structural hole spanner simultaneously by using graph convolutional network based Auto-Encoder.* Neurocomputing. 410: p. 138-150. *doi:10.1016/j.neucom.2020.06.035*
31. Veličković, P., et al. (2017). *Graph attention networks.* arXiv preprint arXiv:1710.10903. *doi:10.48550/arXiv.1710.10903*

32. Goodfellow, I., et al. (2020). *Generative adversarial networks.* Communications of the ACM, 63(11): p. 139-144. *doi:10.1145/3422622*
33. Chen, H., et al. (2019). *Exploiting centrality information with graph convolutions for network representation learning*. *International Conference on Data Engineering*. *doi:10.1109/ICDE.2019.00125*
34. Xin, X., et al. (2017). *Deep community detection in topologically incomplete networks.* Physica A: Statistical Mechanics and its Applications. 469: p. 342-352. *doi:10.1016/j.physa.2016.10.040*
35. Cao, S., et al. (2023). *LGNN: a novel linear graph neural network algorithm.* Frontiers in Computational Neuroscience. *doi:10.3389/fncom.2023.1150105*
36. Zhang, T., et al. (2020). *CommDGI: community detection oriented deep graph infomax*. *International Conference on Information & Knowledge Management*. *doi:10.1145/3340531.3411973*
37. Hu, R., et al. (2020). *Going deep :Graph convolutional ladder-shape networks*. *the AAAI Conference on Artificial Intelligence*. *doi:10.1609/aaai.v34i04.5767*
38. Liu, Y., et al. (2020). *Independence promoted graph disentangled networks*. *AAAI Conference on Artificial Intelligence*. *doi:10.1609/aaai.v34i04.5768*
39. Levie, R., et al. (2018). *Cayleynets: Graph convolutional neural networks with complex rational spectral filters.* IEEE Transactions on Signal Processing. 67(1): p. 97-109. *doi:10.1109/TSP.2018.2879644*
40. Geisler, S., D. Zügner, and S. Günnemann. (2020). *Reliable graph neural networks via robust aggregation.* Advances in neural information processing systems. 33: p. 13272-*13284*. *doi:10.48550/arXiv.2010.15651*
41. Cai, X. and B. Wang. (2023). *A graph convolutional fusion model for community detection in multiplex networks.* Data Mining and Knowledge Discovery. 37(4): p. 1518-1547. *doi:10.1007/s10618-023-00933-9*
42. Li, D., S. Zhang, and X. Ma. (2022). *Dynamic Module Detection in Temporal Attributed Networks of Cancers.* IEEE/ACM Transactions on Computational Biology and Bioinformatics. 19(4): p. 2219-2230. *doi:10.1109/TCBB.2021.3093196*
43. Li, D., Q. Lin, and X. Ma. (2021). *Identification of dynamic community in temporal network via joint learning graph representation and nonnegative matrix factorization.* Neurocomputing. 435: p. 77-90. *doi:10.1016/j.neucom.2021.01.019*
44. Li, D., et al. (2021). *Detecting dynamic community by fusing network embedding and nonnegative matrix factorization.* Knowledge-Based Systems. p. 106961. *doi:10.1016/j.knosys.2021.106961*
45. Li, D., X. Ma, and M. Gong. (2023). *Joint Learning of Feature Extraction and Clustering for Large-Scale Temporal Networks.* IEEE Transactions on Cybernetics. 53(3): p. 1653-1666. *doi:10.1109/TCYB.2021.3128221*
46. Huang.- ,H., et al. (2023). *Diverse Deep Matrix Factorization With Hypergraph Regularization for Multi-View Data Representation.* IEEE/CAA Journal of Automatica Sinica. *doi:10.1109/JAS.2023.123203*
47. Huang, H., et al. (2023). *Exclusivity and consistency induced NMF for multi-view representation learning.* Knowledge-Based Systems. 281: p. 111020. *doi:10.1016/j.knosys.2023.111020*
48. Huang, H., et al. (2024). *Comprehensive Multiview Representation Learning via Deep Autoencoder-Like Nonnegative Matrix Factorization.* IEEE Trans Neural Netw Learn Syst. p. 5953-5967. *doi:10.1109/TNNLS.2022.3200905*
49. Amirfarhad Farhadi, M.M. (2024). Arash Sharifi, and Mohammad Teshnelab. Domaina daptation in reinforcement learning: a comprehensive and systematic study. Frontiers of Information Technology & Electronic Engineering. *doi:10.1631/FITEE.2300356*
50. Kanatsoulis, C.I., N.D. Sidiropoulos, and A.I. Claims. (2022). *GAGE: Geometry Preserving Attributed Graph Embeddings.* Fifteenth ACM International Conference on Web Search and Data Mining. p. 439–448. *doi:10.1145/3488560.3498387*
51. Newman, M.E. (2006). *Modularity and community structure in networks.* Proceedings of the national academy of sciences. 103(23): p. 8577-8582. *doi:10.1073/pnas.0601602103*
52. Jianbo, S. and J. Malik. (2000). *Normalized cuts and image segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(8): p. 888-905. *doi:10.1109/34.868688*
53. Liu, L., et al. (2015). *Community detection based on structure and content: A content propagation perspective*. *IEEE international conference on data mining*. *doi:10.1109/ICDM.2015.153*
54. Shchur, O. and S. Günnemann. (2019). *Overlapping Community Detection with Graph Neural Networks*. doi:10.48550/arXiv.1909.12201
55. Zhang, X., et al. (2019). *Attributed graph clustering via adaptive graph convolution.* arXiv preprint arXiv:1906.01210. doi:10.48550/arXiv.1906.01210
56. Cui, G., et al. (2020). *Adaptive Graph Encoder for Attributed Graph Embedding*. p. 976-985. doi:10.1145/3394486.3403150
57. Huang, W. (2021). *Graph Auto-Encoders with Edge Reweighting.* International Journal of Reconfigurable and Embedded Systems (IJRES). doi:10.33899/rengj.2021.131549.1102
58. Sen, P., et al. (2008). *Collective classification in network data.* AI magazine, 29(3): p. 93-93doi:10.1609/aimag.v29i3.2157.
59. Namata, G., et al. (2012). *Query-driven active surveying for collective classification*. in *10th International Workshop on Mining and Learning with Graphs*. doi:10.1145/2442476.2442482

60. Rice, S.A. (1927). *The identification of blocs in small political bodies.* American Political Science Review. 21(3): p. 619-627. doi:10.2307/1945514
61. Zhu, W., X. Wang, and P. Cui, (2020). *Deep learning for learning graph representations*, in *Deep learning: concepts and architectures*. p. 169-210. doi:10.1007/978-3-030-31756-0_6
62. Cao, S., W. Lu, and Q. Xu. (2016). *Deep neural networks for learning graph representations*. *AAAI Conference on Artificial Intelligence*. doi:10.1609/aaai.v30i1.10105
63. Sun, F.-Y., et al. (2019). *vGraph: a generative model for joint community detection and node representation learning*, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc. doi:10.48550/arXiv.1906.07159
64. Tian, F., et al. (2014). *Learning Deep Representations for Graph Clustering*. Proceedings of the AAAI Conference on Artificial Intelligence. 28(1). doi:10.1609/aaai.v28i1.8889
65. Kipf, T. and M. Welling. (2016). *Variational Graph Auto-Encoders.* doi:10.48550/arXiv.1611.07308
66. Pan ,S., et al. (2019). *Learning Graph Embedding With Adversarial Training Methods.* IEEE Transactions on Cybernetics. p. 1-13. doi:10.1109/TCYB.2019.2932097
67. Wang, C., et al. (2019). *Attributed Graph Clustering: A Deep Attentional Embedding Approach*. 3670-3676. doi:10.24963/ijcai.2019/510
68. Zheng, S., et al. (2020). *Distribution-Induced Bidirectional Generative Adversarial Network for Graph Representation Learning*. p. 7222-7231. doi:10.1109/CVPR42600.2020.00728
69. Park, J., et al. (2019). *Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning*. doi:10.48550/arXiv.1908.02441
70. Guo, L. and Q .Dai. (2021). *Graph Clustering via Variational Graph Embedding.* Pattern Recognition. 122: p. 108334. doi:10.1016/j.patcog.2021.108334
71. Yang, C., et al.(2015). *Network representation learning with rich text information*. in *Twenty-fourth international joint conference on artificial intelligence*. doi:10.5555/2832415.2832492
72. Xia, R., et al. (2014). *Robust Multi-View Spectral Clustering via Low-Rank and Sparse Decomposition.* Proceedings of the AAAI Conference on Artificial Intelligence. 28(1). doi:10.1609/aaai.v28i1.8990
73. Ahmadi, M., M. Safayani, and A. Mirzaei. (2022). *Deep Graph Clustering via Mutual Information Maximization and Mixture Model.* arXiv preprint arXiv:2205.05168. doi:10.48550/arXiv.2205.05168
74. Li, J., et al. (2020). *Dirichlet Graph Variational Autoencoder*. doi:10.48550/arXiv.2005.11578
75. Xie, H. and Y. Ning. (2023). *Community detection based on BernNet graph convolutional neural network.* Journal of the Korean Physical Society. 83(5): p. 386-395. doi:10.1007/s40042-023-00893-9
76. Deng, L., B. Guo, and W. Zheng. (2024). *GCN-based weakly-supervised community detection with updated structure centres selection.* Connection Science. 36(1): p. 2291995. doi:10.1080/09540091.2024.2291995
77. Ng, A., M. Jordan ,and Y. Weiss. (2002). *On Spectral Clustering: Analysis and an algorithm.* Adv. Neural Inf. Process. Syst. doi:10.5555/2980539.2980649
78. Tian, F., et al. (2014). *Learning Deep Representations for Graph Clustering.* Proceedings of the National Conference on Artificial Intelligence. p. 1293-1299. doi:10.1609/aaai.v28i1.8889
79. Sun, F.-Y., et al. (2019). *vGraph: A Generative Model for Joint Community Detection and Node Representation Learning*. doi:10.48550/arXiv.1906.07159