



## Quarterly Journal of Optimization In Soft Computing

Vol. 3, Issue 1, Spring 2025

- **Variational Graph Autoencoder for Unsupervised Community Detection in Attributed Social Networks**  
Omid Rashnodi, Maryam Rastegarpour, Azadeh Zamanifar, Parham Moradi
- **Using Machine Learning to Discover Traffic Patterns in Software Defined Networks**  
Abdulrazzaq Mosa Al-Mhanna, Pouya Khosravian Dehkordi
- **Review of Machine Learning Algorithm in Medical Health**  
Zahra Ghorbani, Sahar Behrouzi-Moghaddam, Shahram Zandiyan, Babak Nouri-Moghaddam, Nasser Mikaeilvand, Sajjad Jahanbakhsh, Ailin Khosravani, Fatemeh Tahmasebizadeh, Abbas Mirzaei
- **Mathematical Modeling For Relocation Of Terminal Facilities In Location Problems**  
Mehdi Fazli, Somayyeh Faraji
- **Fuzzy Logic-Based Vector Control Method for Induction Motors**  
Gholam Reza Aboutalebi
- **Learning Rate Optimization of U-Net Architecture Using Grasshopper Optimization Algorithm to Enhance Accuracy in CT Image Segmentation of COVID-19 Patients**  
Alireza Mehravin, Mostafa Zaare, Reza Mortazavi



Paper Type (Research paper)

## Variational Graph Autoencoder for Unsupervised Community Detection in Attributed Social Networks

Omid Rashnodi<sup>1</sup>, Maryam Rastegarpour<sup>\*2</sup>, Azadeh Zamanifar<sup>1</sup>,  
Parham Moradi<sup>3</sup>

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	Abstract
<p><b>Article History:</b> Received: 2025/02/21 Revised: 2025/03/08 Accepted: 2025/04/12</p> <p><b>DOI:</b></p> <p><b>Keywords:</b> Community Detection, Attributed Social Networks, Variational Graph Autoencoder, Graph Convolutional Networks, Deep Learning, Node Embeddings, Network Topology</p> <p>Omid Rashnodi <a href="mailto:omid.rashnodi@iau.ac.ir">omid.rashnodi@iau.ac.ir</a></p> <p>* Maryam Rastegarpour <a href="mailto:m.rastgarpour@gmail.com">m.rastgarpour@gmail.com</a></p> <p>Azadeh Zamanifar <a href="mailto:azamanifar@srbiau.ac.ir">azamanifar@srbiau.ac.ir</a></p> <p>Parham Moradi <a href="mailto:p.moradi@uok.ac.ir">p.moradi@uok.ac.ir</a></p>	<p>This paper introduces a novel approach named VGAE (Variational Graph AutoEncoder Embedding), an innovative deep-learning framework for detecting communities in attributed social networks. By synergistically integrating node content with network topology, VGAE 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 VGAE to create a meaningful representation of the graph. In the decoder component of VGAE, 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 VGAE 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, VGAE reached 70.46% on Cora, 55.60% on PubMed, and 57.06% on Citeseer, consistently outperforming existing methods. These results confirm the superiority of VGAE in accurately capturing community structures within large, complex networks, making it a highly effective tool for unsupervised community detection.</p>

## 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 VGAE (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 VGAE, 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 VGAE 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, VGAE 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 VGAE 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, VGAAE, 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 topological 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  $\mathcal{C} = \{C_1, C_2, \dots, C_r\}$ . 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  $f: v \rightarrow \{1, 2, 3, \dots, r\}$  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  $\forall v \in C: \text{degree } H(v) \geq k$ , 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  $A = [a_{ij}] \in R^{n \times n}$  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.  $S = [s_{ij}] \in R^{n \times n}$  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:

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

Where  $\psi_i$  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):



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

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

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

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  $\psi_i$  to take any integer value, the problem can be easily solved as equation (4):

$$\max \Phi = \max \text{Tr}(\Psi^T B \Psi) \quad (4)$$

Where  $\Psi = [\psi_{ij}] \in R^{n \times p}$  is the matrix that hints at membership in the community, and  $\text{Tr}(\cdot)$  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  $\Psi$  reconstruction of network topology from a community structure viewpoint. Therefore, any row of the  $\Psi$  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  $\text{Cut}(A, B)$ , represents the total number of edges that connect nodes in different clusters. The volume of cluster AA, represented as  $\text{Vol}(A)$ , is the sum of the degrees of all nodes within cluster A [52]. These values are determined using equations (5) and (6):

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

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

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  $C_1, C_2 \dots C_k$ .

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

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

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

$$\begin{aligned} \min \quad & \text{Tr}(\Phi^T L \Phi) \\ \Phi \in & R^{n \times k} \\ \text{s.t.} \quad & L = D - S \\ D = & \text{diag}(d_1, d_2, \dots, d_n) \\ \Phi_{ij} = & \begin{cases} \frac{1}{\sqrt{\text{vol}(C_j)}} & \text{if } v_i \in C_j \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (9)$$

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

In the case of this problem, the solution matrix  $\Phi$  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  $\Phi$  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  $V = \{v_i\} \mid i = 1, 2, \dots, n$  is formed of a set of graph nodes and  $e_{ij} = \langle v_i, v_j \rangle \in E$  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  $A_{ij} = 0$ .  $X \in R^{n \times d}$  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  $\bar{z}_i \in R^d$ , with a formal format  $f: (A, X) \rightarrow Z$ , where  $z_i^T$  is the  $i$ -th row of the  $Z \in R^{n \times d}$  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: VGAE

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.

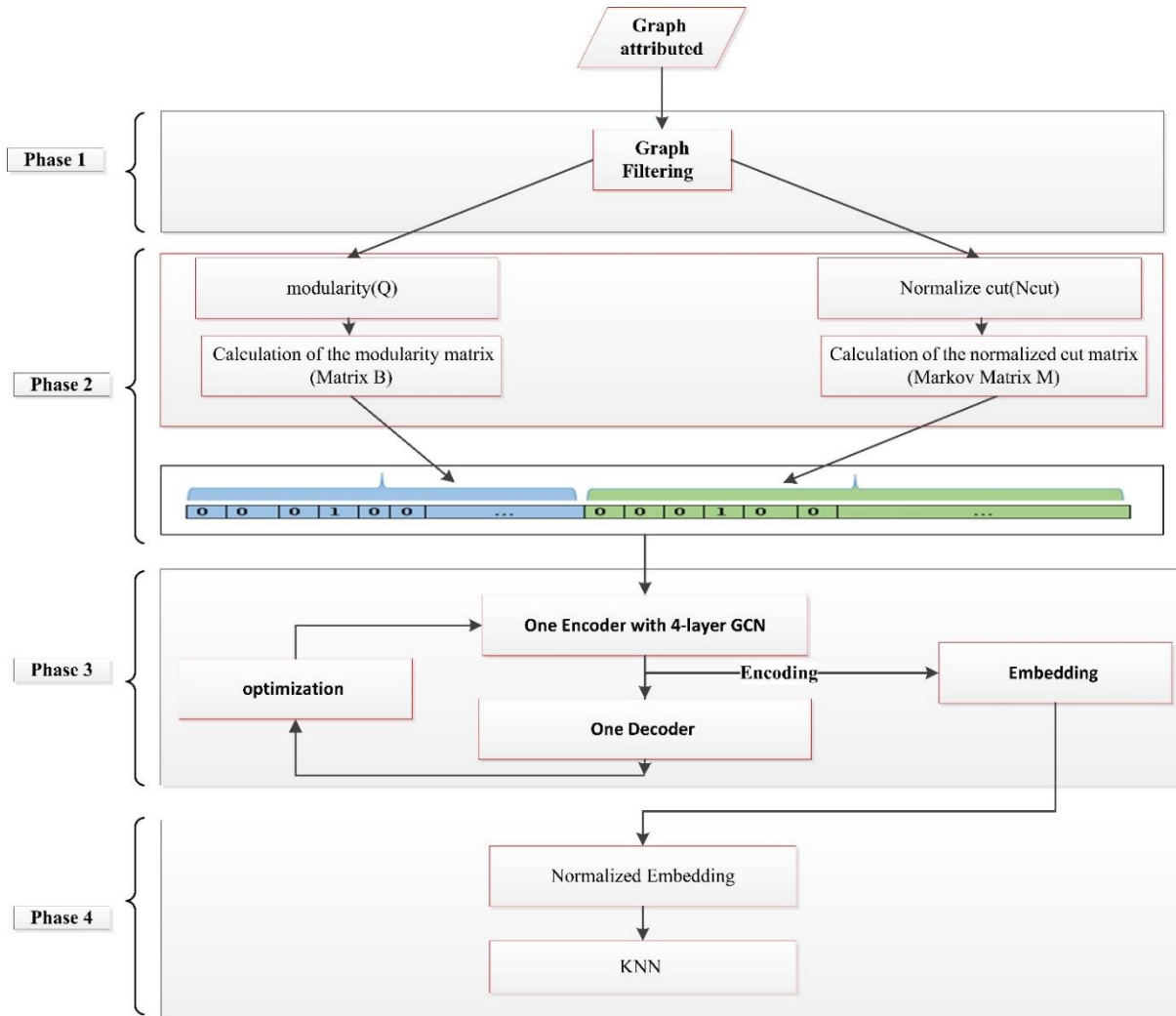
Through successive iterations, the graph is methodically reduced by excluding nodes lacking sufficient connectivity, ultimately yielding a simplified core that depicts the most interconnected nodes. As delineated in this section, the k-core algorithm inherently defines a community based on its density, 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 VGAE**

### 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:

$$\bar{X} = GCN(X, A) = ReLU(\tilde{A} X W_0) \quad (11)$$

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

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

$$\mu = GCN_{\mu}(X, A) = \tilde{A} \bar{X} W_1 \quad (12)$$

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

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

$$GCN(X, A) = \tilde{A} ReLU(\tilde{A} X W_0) W_1 \quad (13)$$

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

$$Z = \mu + \sigma * \epsilon \quad \text{Where } \epsilon \sim N(0, I). \quad (14)$$

#### 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  $\hat{A}$ , which is defined as Equation 15:

$$\hat{A} = \sigma(z z^T) \quad (15)$$

Where  $\sigma(\bullet)$  is the logistic sigmoid function. In summary, the encoder is represented as Equation 16:

$$q(z_i | X, A) = N(z_i | \mu_i, \text{diag}(\sigma^2)) \quad (16)$$

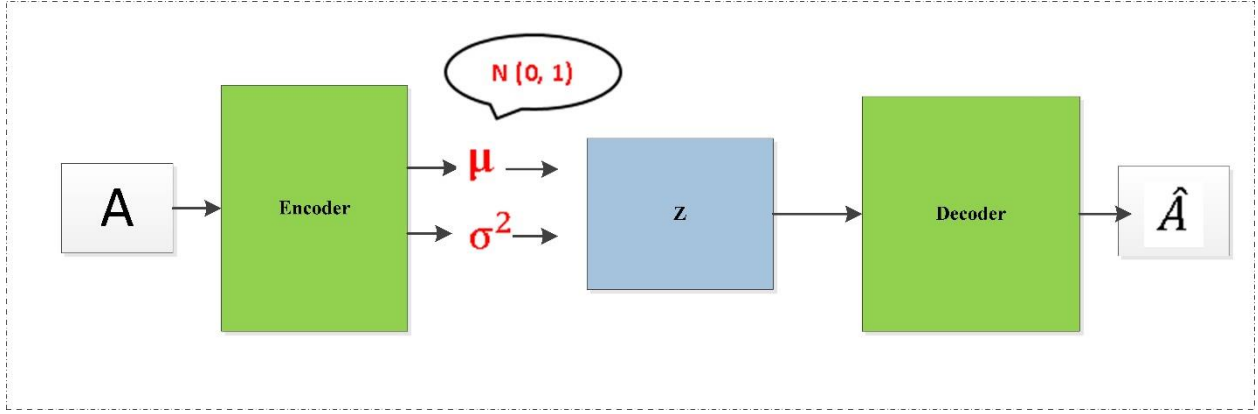


Fig. 2: The workflow scheme of the Variational graph autoencoder in the proposed method

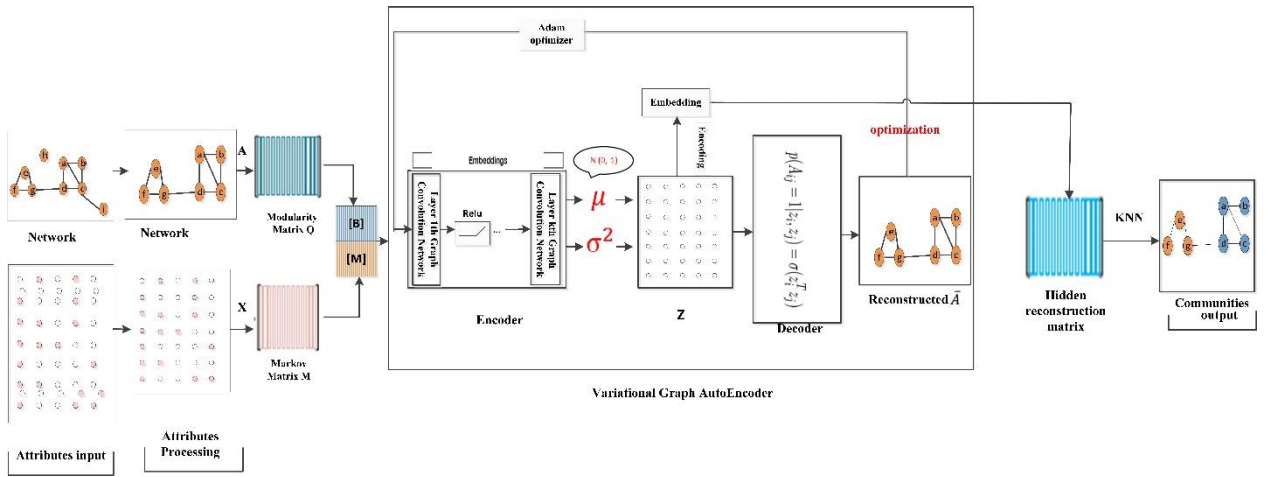


Fig. 3: The VGAAE Framework for Community Detection in Attributed Social Networks.

The decoder is represented in Equation 17:

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

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  $X_{new}$ :

$$X_{new} = \alpha Q + \beta M \quad (18)$$

Where  $\alpha$  and  $\beta$  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  $\mu$  and  $\log \sigma^2$ :  $\mu = GCN_\mu(X', A) = \tilde{A}XW_1$

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

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

$A' = \text{sigmoid}(\tilde{A}XW_2)$  Where  $W$  are the weights associated with the decoder. Using the reparameterization trick:  $Z = \mu + \sigma \delta \sim N(0, 1)$  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.

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

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.

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

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

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

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

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  $Z_{\text{final}}$  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  $Z_{\text{final}}$  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 VGAAE.

## 5. Experiment

In this section, we describe the comprehensive experiments conducted to evaluate the performance of the Variational Graph Autoencoder Embedding Enhancer (VGAAE) 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].

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

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].

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

Where  $n$  is the number of groups, and for a specific group,  $i$  and  $C_i$ ,  $\hat{C}_i$  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 VGAE 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. VGAE'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 VGAE 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 VGAE 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 VGAE 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 VGAE with other advanced graph-based methods, it still maintains a leading position. For example, the semi-supervised MRFGCN 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 VGAE. Additionally, methods like AGC and CommDGI, while competitive, do not achieve the same balance between ACC and NMI, suggesting that VGAE's method of integrating features and graph topology potentially offers a more robust model for understanding complex network structures. Overall, the superiority of VGAE 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 VGAE method relative to other community detection algorithms across various learning paradigms for the Citeseer dataset. VGAE, an unsupervised method, stands out with an ACC% of 75.60 and an NMI% of 57.06. Notably, VGAE 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 MRFGCN and supervised methods such as BernNet GCN and WC-GCN, VGAE demonstrates competitive or superior performance, particularly in accuracy. This highlights VGAE'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, VGAE excels in handling unlabeled and complex datasets while maintaining a balance between accuracy and information preservation.

The proposed VGAE 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 VGAE 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	<b>7</b>
Citeseer [58]	3,312	4,715	3,703	<b>6</b>
PubMed [59]	19,717	44,338	500	<b>3</b>

**Table 5: Detailed parameter setting**

Datasets	Training	Learning	Activation	Weight	Optimizer	GCN	Dropout	#Train/Validation
	Epoch	rate	Function	Decay		layers	rate	/Test Node
<b>Cora</b>	200	0.01	Relu	5e-3	Adam	64/32	0.5	140/500/1000
<b>Citeseer</b>	200	0.01	Relu	5e-3	Adam	64/32	0.5	120/500/1000
<b>PubMed</b>	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%
<b>K-means</b>	Unsupervised	Feature	49.2	<b>32.1</b>
<b>Spectral-F [77]</b>	Unsupervised	Feature	34.7	<b>14.7</b>
<b>Spectral-G [77]</b>	Unsupervised	Graph	31.46	<b>9.69</b>
<b>DeepWalk [14]</b>	Unsupervised	Graph	56.20	<b>39.87</b>
<b>Graph Encoder [78]</b>	Unsupervised	Graph	32.5	<b>10.9</b>
<b>vGraph[63]</b>	Unsupervised	Graph	28.7	<b>34.5</b>
<b>TADW [71]</b>	Unsupervised	Feature & Graph	55.00	<b>36.59</b>
<b>VGAE [65]</b>	Unsupervised	Feature & Graph	63.56	<b>47.45</b>
<b>MGAE [18]</b>	Unsupervised	Feature & Graph	63.43	<b>45.57</b>
<b>ARGE [66]</b>	Unsupervised	Feature & Graph	60.84	<b>42.21</b>
<b>ARVGA [66]</b>	Unsupervised	Feature & Graph	62.83	<b>45.93</b>
<b>DGVAE [74]</b>	Unsupervised	Feature & Graph	64.42	<b>47.64</b>
<b>AGC [55]</b>	Unsupervised	Feature & Graph	68.92	<b>53.68</b>
<b>CommDGI [11]</b>	Unsupervised	Feature & Graph	69.8	<b>57.9</b>
<b>DAEGC [67]</b>	Unsupervised	Feature & Graph	70.4	<b>52.8</b>
<b>GC-VGE [70]</b>	Unsupervised	Feature & Graph	70.67	<b>53.57</b>
<b>GALA [69]</b>	Unsupervised	Feature & Graph	72.42	<b>53.96</b>
<b>DBGAN [68]</b>	Unsupervised	Feature & Graph	74.6	<b>57.7</b>
<b>GMIM [73]</b>	Unsupervised	Feature & Graph	74.8	<b>56.0</b>
<b>AGE[56]</b>	Unsupervised	Feature & Graph	76.8	<b>60.7</b>
<b>MRFasGCN[27]</b>	Semi-supervised	Feature & Graph	84.3	<b>66.2</b>
<b>BernNet GCN[75]</b>	Supervised	Feature & Graph	41.06	<b>68.78</b>
<b>LGNN[35]</b>	Supervised	Feature & Graph	79.04	-
<b>WC-GCN[76]</b>	Supervised	Feature & Graph	79.39	-
<b>VGAEE(proposed method)</b>	<b>Unsupervised</b>	<b>Feature &amp; Graph</b>	<b>84.5</b>	<b>70.46</b>

**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	-
<b>VGAAE(proposed method)</b>	Unsupervised	Feature & Graph	<b>80.50</b>	<b>55.60</b>

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

<b>AGE[56]</b>	Unsupervised	Feature & Graph	70.2	44.8
<b>MRFaSGCN[27]</b>	Semi-supervised	Feature & Graph	73.2	46.3
<b>BernNet GCN[75]</b>	Supervised	Feature & Graph	72.32	58.01
<b>LGNN[35]</b>	Supervised	Feature & Graph	73.15	-
	Supervised	Feature & Graph	73.2	46.3
<b>WC-GCN[76]</b>	Supervised	Feature & Graph	75.18	-
<b>VGAEE</b> (proposed method)	<b>Unsupervised</b>	<b>Feature &amp; Graph</b>	<b>75.60</b>	<b>57.06</b>

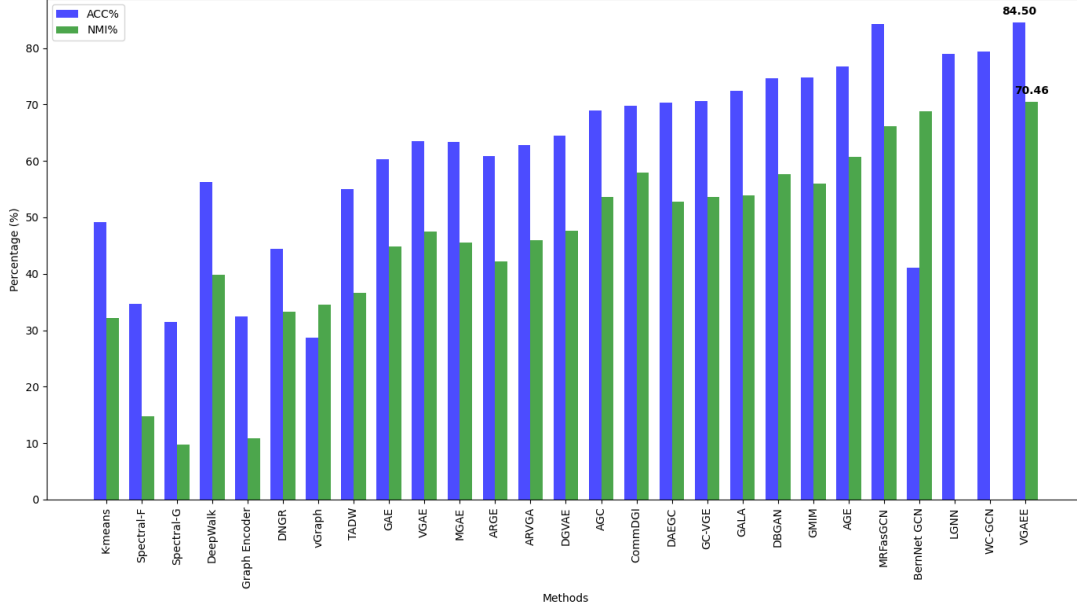


Fig. 4. Performance comparison of different community detection methods on the Cora dataset

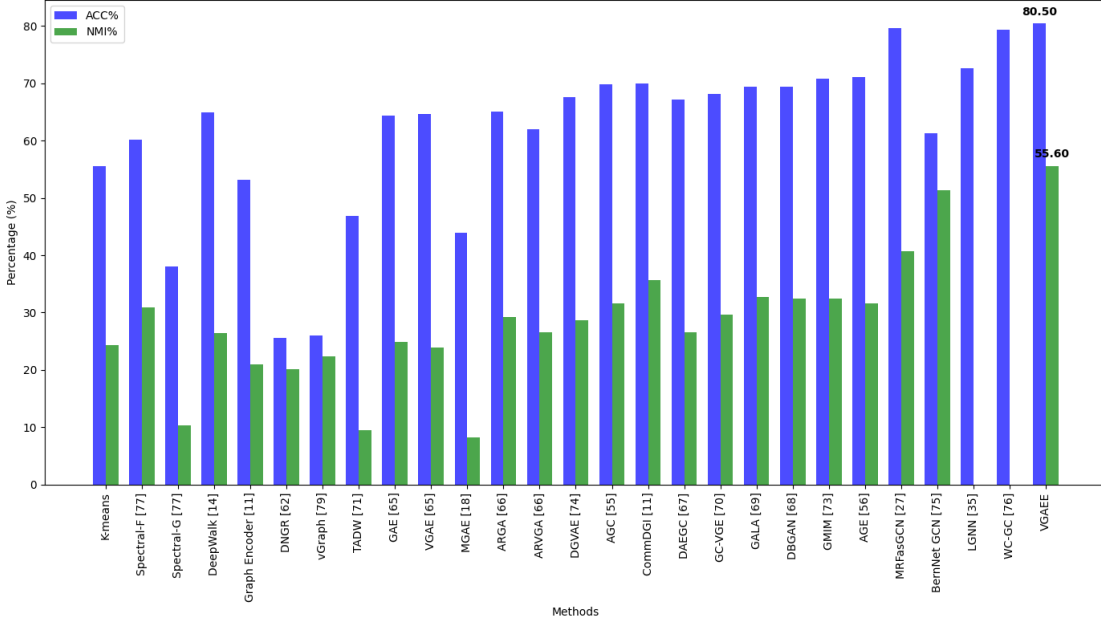
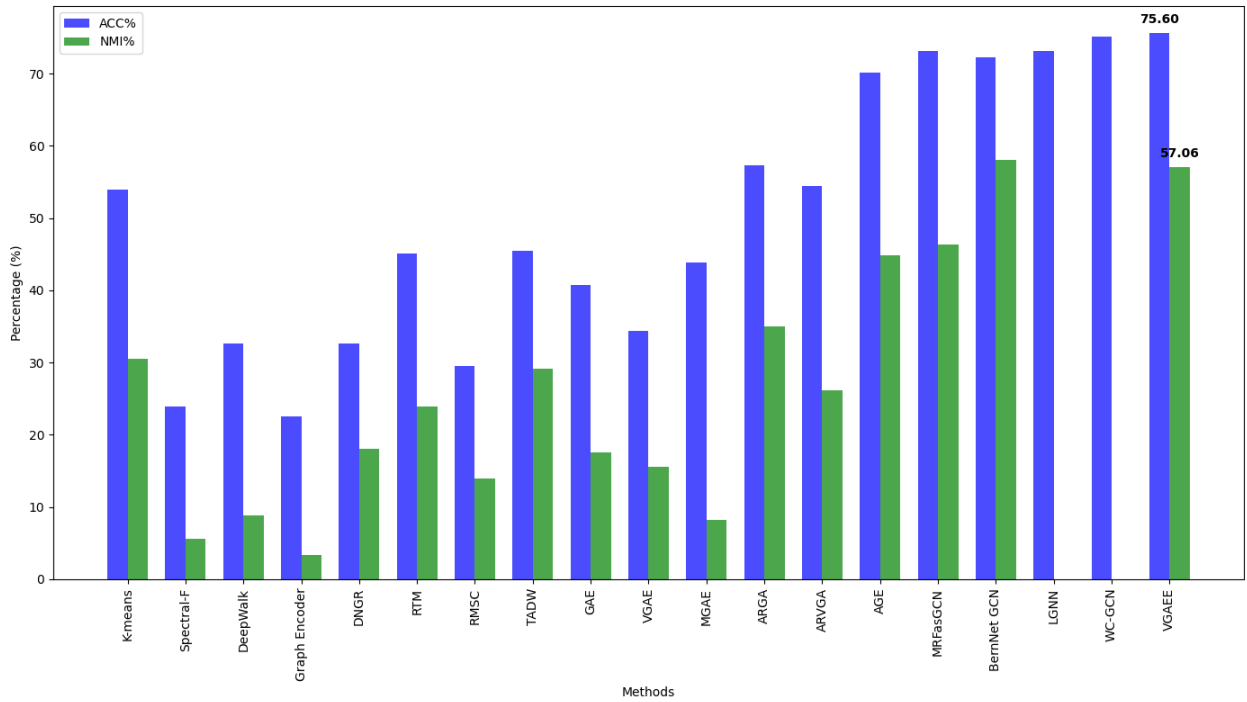


Fig. 5. Performance comparison of different community detection methods on the PubMed dataset



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

## 6. Conclusion and future work

This study introduced VGAE, an innovative unsupervised approach leveraging Variational Graph AutoEncoders to enhance community detection in attributed social networks. By integrating node content with network topology, VGAE effectively captures complex community structures, achieving superior performance metrics across diverse datasets like Cora, Citeseer, and PubMed. Notably, VGAE 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 VGAE 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 VGAE 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 VGAE 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. Domain adaptation 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





Paper Type (Research paper)

## Using Machine Learning to Discover Traffic Patterns in Software Defined Networks

Abdulrazzaq Mosa Al-Mhanna<sup>1</sup> and Pouya Khosravian Dehkordi<sup>1,\*</sup>

1. Department of Computer Engineering, Shahrekord Branch, Islamic Azad University, Shahrekord, Iran.

### Article Info

#### Article History:

Received: 2024/08/17

Revised: 2025/04/14

Accepted: 2025/05/23

DOI:

#### Keywords:

Network Traffic, Software-Defined Networks, SDN, Machine Learning

\*Corresponding Author's Email  
Address: [drkhosravian@iau.ac.ir](mailto:drkhosravian@iau.ac.ir)

### Abstract

In this research, we introduce a deep learning model based on Convolutional Neural Networks (CNNs) along with the Bird Swarm Optimization algorithm to identify and discover traffic patterns in Software-Defined Networks (SDNs). The main objective of this study is to investigate the capability of deep learning models in analyzing traffic data and identifying unique patterns present in SDNs. Using a diverse and comprehensive dataset, the proposed model is trained and evaluated. The use of CNNs, due to their layered structure and deep learning capabilities, enables the identification of unique traffic patterns that are prominently visible in SDNs. The proposed model, with high accuracy and good generalization ability, can serve as an effective tool in enhancing the accuracy and efficiency of traffic pattern identification systems in software-defined networks. This research not only demonstrates the superiority of deep learning models in traffic pattern recognition but also provides practical and effective solutions for traffic analysis and management in SDNs. The results of this study indicate that the proposed model achieves an accuracy of 96.5%, suggesting that the proposed method can significantly contribute to the development and improvement of security systems and performance optimization in software-defined networks.

## 1. Introduction

Management and configuration of computer networks has become a difficult and vital task due to their complexity and dynamics. These networks consist of a collection of switches, routers, firewalls, and other intermediate devices that work simultaneously. Proper implementation of these networks is possible by operators dealing with a limited set of configuration commands in command-line environments and with complex administrative tasks and policies. These policies and complexities are not enough to react to the continuous changes of the network. For this reason, network configuration modifications are done manually to adapt the network to the changes. Operators use external tools to overcome

these limitations, and these constant changes may lead to more configuration errors [1,24].

### 1.2. problem Statement

For network management, service measurement, and network monitoring, traffic classification is an intelligent process that involves categorizing traffic into multiple groups. In addition, traffic classification enables the configuration of access restrictions, quality of service, and other network security features efficiently and allocates resources. Deep packet inspection and port-based methods are popular methods for traffic classification [2]. However, both of these methods are currently less used, as most applications use dynamic ports and the network communication is encrypted. Therefore, it is very important to

develop a new classification method that is more suitable for today's operational environment. The purpose of this research is to discover network traffic patterns with high accuracy. To extract the patterns, a deep learning based approach is proposed.

### 1.3 innovation in research

This research pushes the boundaries by exploring and applying advanced deep learning architectures such as deep neural networks (DNN), convolutional neural networks (CNN), recurrent neural networks (RNN), and attention mechanisms. By doing so, an attempt is made to provide a pioneering approach to modeling and understanding network traffic patterns. In summary, the innovative aspect of this research lies in its pioneering use of deep learning models to achieve high accuracy in discovering and analyzing traffic patterns in software-defined networks [25]. This approach has the potential to transform network management, security, and performance optimization, making it a cornerstone for further advancements in this field.

## 2. Software networks

Organizations have invested heavily in virtualization and hybrid clouds, but they still face challenges, including quickly allocating network connections while systems are running. Often these problems arise due to issues related to policy or implementation processes.

These problems can be partially solved by creating virtual network infrastructure. This infrastructure is easily reassigned, such as when a new SAN or server is implemented. The idea behind this software-defined network management infrastructure, or SDN, is not that new and has been around for over a decade. One efficient definition of SDN is the separation of data and control functions of routers and other layer 2 infrastructure of conventional networks using a programming interface.

### 2.1 PSO algorithm

The PSO algorithm is an optimization method based on probability rules that was first invented in 1995 by Kennedy and Aberhart [3] inspired by the behavior of birds when searching for food. In this algorithm, first a set of initial answers is generated. Then, to find the optimal answer in the space of possible answers, or to time the generations, the answer search is done. Each particle is defined multidimensionally with two along with their descriptions are presented in Table 1-3. Meta-heuristic algorithms are known as effective techniques to improve the performance of CNN architectures by optimizing their meta-parameters.

values of position and velocity, and at each stage of the particle's movement, with two indices of velocity and position, the best answers are determined in terms of merit for all particles.

$$\vec{X}_i(t) = (x_i^1(t), x_i^2(t), \dots, x_i^d(t), \dots, x_i^n(t)).$$

$$\vec{V}_i(t) = (v_i^1(t), v_i^2(t), \dots, v_i^d(t), \dots, v_i^n(t)).$$

### 2.2 Related works

Basic machine learning methods that enable traffic classification in SDN are reviewed in this section. Through the use of artificial intelligence (AI), machine learning enables computers to recognize complex patterns from massive data sets on their own. Operationally, machine learning is divided into two steps: 1) training, which involves providing the machine learning algorithms with a subset of the data set (called the training set) so that the system model can learn from it, and 2) decision making, which is capable of The system is trained to predict the result of the new input using the model. Supervised, unsupervised, semi-supervised and reinforcement learning categories are used to group machine learning algorithms [4], [5] and [6].

Numerous machine learning methods have been developed over the years as a result of research efforts. For problems with large data sets, machine learning is often the most effective approach. Considering that machine learning techniques are designed for pattern recognition and data identification, they are suitable for solving problems in SDNs.

### 3. Proposed method

Optimizing the parameters of convolutional neural networks includes determining the appropriate parameters, which results in significant accuracy in each task. However, the task of optimizing a large number of parameters is very difficult and computationally expensive. Therefore, it is necessary to implement optimization algorithms that reduce the number of iterations. The present study is based on the Particle Swarm Optimization (PSO) technique to find the CNN model with the highest accuracy for breast cancer detection. The development of a convolutional neural network (CNN) involves the optimization of several parameters and the precise choice of architecture. Choosing the optimal parameters is very important to obtain accurate results when using Convolutional Neural Networks (CNN). Therefore, it is a challenging task that requires a considerable level of expertise.

The effectiveness of a CNN model depends on its meta-parametric parameters, so some researchers emphasize the necessity of fine-tuning these meta-parameters to obtain positive results. Hence, it is a challenging task that requires a considerable amount of skill. The meta-parameters of the CNN architecture

### 3.2 Implementation of neural network algorithm optimized with particle swarm optimization algorithm

In this section, we build a CNN from scratch (a new model) to train a Convolutional Neural Network (CNN). It consists of three convolution layers with three maximum localization layers, one dropout layer, one flattening layer, and two fully connected (FC) layers. The activation function for each layer is the ReLU function, except for the last layer which is output and uses the sigmoid function. The output layer uses a sigmoid function that maps the output value to the interval [0, 1]

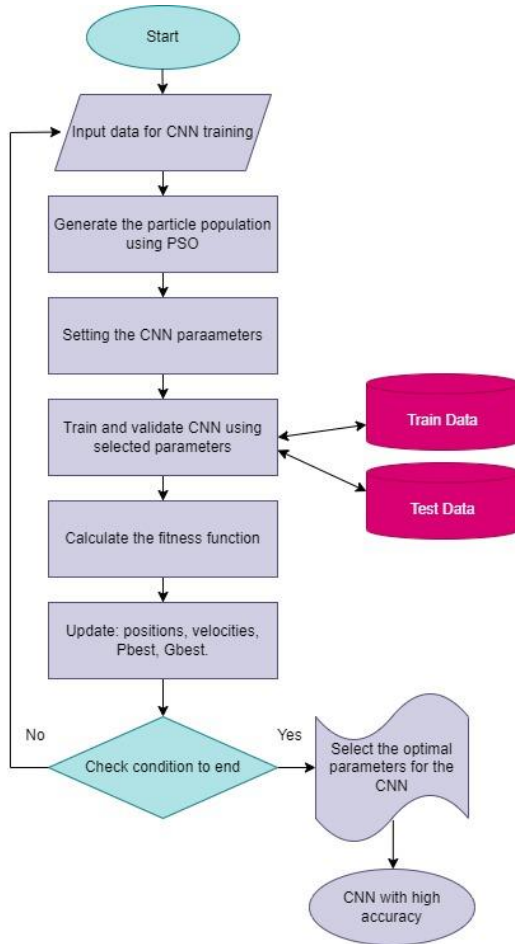


Figure 1-3: Flowchart of neural network optimized by PSO

#### 4.1 evaluation criteria

In this section, various evaluation criteria used to measure the performance of machine learning models in discovering traffic patterns in software-centric networks are described in detail. These criteria include accuracy, recall, and F1 score. Each of these metrics evaluates different aspects of model performance and provides a deeper understanding of model performance.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

$$Sensitivity(Recall) = \frac{TP}{TP+FN} \quad (2)$$

$$F1\ score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)} \quad (3)$$

#### 4.2 Data collection

Software-oriented networks (SDN) are one of the leading technologies in network management and control, which enable centralized control and higher flexibility by separating the control layer from the data layer. In the field of SDN, multiple datasets are used for various purposes, including network traffic analysis, attack detection, and network performance optimization.

The NSL-KDD dataset is one of the most popular and comprehensive datasets in the field of network security and intrusion detection. This dataset is an improved dataset of KDD Cup 99 and is designed to address its problems and limitations. The KDD Cup 99 dataset was introduced as one of the first and most comprehensive datasets in the field of intrusion detection.

- Duplicate data: The presence of a large number of duplicate samples in the dataset, which caused the machine learning models to mistakenly perform very well.

- Imbalance in the data: unbalanced distribution of different samples in the data set, which caused the models to tend to oversampled classes.

#### 4.3 The size of the parameters

In this project, our main goal is to use Convolutional Neural Networks (CNN) and Particle Swarm Optimization (PSO) algorithm to discover traffic patterns in software-oriented networks. For this purpose, a set of parameters for convolutional neural network and PSO algorithm are considered.

#### 4.4 Training settings:

1. Initial learning rate for updating network weights. A low value of this parameter allows the network to be trained with smaller and more accurate steps.
2. The number of complete training iterations on the training data. Increasing this value helps the model to reach higher accuracy.
3. The number of examples in each small training package. This parameter helps balance between training speed and stability.

#### 4.5. Simulation results

To evaluate the performance of the proposed method, accuracy, recall and F1-Score criteria have been used. 60% of the data was used for training and 40% for testing. The results obtained from the evaluation of the proposed method and its comparison with two other references are as follows:

Table 4-1: Comparison of the proposed method with other methods

	evaluation criteria		
	accuracy	recall	F1-score
Proposed method	96.5	94.86	95.85
[38]	87.6	89.6	90.01

[40]	92.38	93.11	94.87
------	-------	-------	-------

The results show that the proposed method has the highest accuracy and F1 criterion compared to the other two references, and it is close to the highest value in readout. This shows that the proposed method has been able to establish a good balance between correctly identifying attack samples and preventing false positive samples.

The proposed method using convolutional neural networks and particle swarm optimization algorithm has been able to show better performance than the previous methods in detecting penetration and traffic analysis of SDN networks.

In this section, the results of the evaluation of different machine learning algorithms to discover traffic patterns in software-oriented networks (SDN) are analyzed. The following table shows the accuracy results of different algorithms:

**Table 4-2: Comparison of the proposed method with other algorithms**

Algorithm	Accuracy
<b>proposed method</b>	96.5
<i>KNN</i>	71.47
<i>DT</i>	95.76
<i>SVM</i>	95.74

The proposed algorithm, which uses convolutional neural networks (CNN), has the best performance among the investigated algorithms with an accuracy of 96.5%. This result shows that CNN, with its capabilities in extracting complex features and deep learning, has been able to identify traffic patterns well and achieve higher accuracy than other algorithms.

The K-Nearest Neighbor (KNN) algorithm with 71.47% accuracy has the lowest accuracy among the investigated algorithms. This result shows that KNN may perform poorly when dealing with complex and high-dimensional data. Due to the simplicity of this algorithm and the inability to extract complex features, it provides less accuracy.

The decision tree (DT) algorithm has performed very well with an accuracy of 95.76% and is known as one of the efficient algorithms in identifying traffic patterns. Decision tree using tree structure and decision rules has been able to achieve high accuracy and work well with traffic data.

The support vector machine (SVM) algorithm has also performed well with 95.74% accuracy. SVM has been able to detect traffic patterns with high accuracy by using feature spaces and optimal separators. Although the accuracy of SVM is slightly lower than decision tree, it is still in the high performance range.

The results show that the proposed method using convolutional neural networks (CNN) has been able to achieve the best accuracy among the investigated

algorithms. This shows the high power of CNN in identifying and learning complex patterns in traffic data. On the other hand, more traditional algorithms such as KNN, DT and SVM have also performed significantly, but could not reach the accuracy of the proposed method.

According to these results, the use of convolutional neural networks (CNN) as the proposed method in this research is a suitable choice and can help improve the accuracy and efficiency of traffic pattern detection systems in software-based networks. This method has many capabilities in analyzing complex data and extracting important features, which has made it a powerful tool in the field of machine learning.

## 5. Conclusion

the data set used in this research included various network traffics, including normal and abnormal traffics. The data has been collected from various sources to have high diversity and realism. The data pre-processing process has included cleaning, normalization and extraction of important features, which has greatly helped to improve the quality of the data and the accuracy of the models.

In this research, four main algorithms have been evaluated: k-nearest neighbor (KNN), decision tree (DT), support vector machine (SVM) and convolutional neural networks (CNN).

In addition to accuracy, other criteria such as recall and F1 criteria have also been used to evaluate the performance of models.

The proposed algorithm, which uses convolutional neural networks (CNN), has the best performance among the investigated algorithms with an accuracy of 96.5%. This result shows that CNN, with its capabilities in extracting complex features and deep learning, has been able to identify traffic patterns well and achieve higher accuracy than other algorithms. The readability of 94.86% and the F1 criterion equal to 95.85% also indicate the high ability of this algorithm to correctly detect positive samples and reduce positive and negative errors.

The K-Nearest Neighbor (KNN) algorithm with 71.47% accuracy has the lowest accuracy among the investigated algorithms. This result shows that KNN may perform poorly when dealing with complex and high-dimensional data. Due to the simplicity of this algorithm and the inability to extract complex features, it provides less accuracy.

The decision tree (DT) algorithm has performed very well with an accuracy of 95.76% and is known as one of the efficient algorithms in identifying traffic patterns. The decision tree has been able to categorize the traffic data well and achieve high accuracy by using the tree structure and decision rules. This algorithm is one of the popular methods in traffic data analysis due to its simplicity and high efficiency. But it has a weaker performance than the proposed method.

One of the main advantages of a decision tree is that it is naturally interpretable. This feature allows network administrators and analysts to easily understand the

reasons behind the decisions and classifications made by the model. This interpretability is especially valuable in cases where there is a need to explain the results to non-technical managers.

Decision tree can also work well with data with different features and incomplete data. However, one of the weaknesses of this algorithm may be the creation of overly complex trees and overfitting in the training data. To reduce this problem, techniques such as pruning are used to reduce the complexity of the tree and improve the model.

The support vector machine (SVM) algorithm has also performed well with an accuracy of 95.74%. Using feature spaces and optimal separators, SVM has been able to recognize traffic patterns with high accuracy. Although the accuracy of SVM is slightly lower than decision tree, it is still in the high performance range. The recall and F1 criterion for SVM are not available in this table, but it can be expected that this algorithm also performs well in these criteria.

One of the main advantages of SVM is its ability to work with high-dimensional data and determine optimal decision boundaries for separating classes. This feature makes SVM perform very well, especially in cases where the data is not linearly separable. However, one of the challenges of using SVM is the need to fine-tune its various parameters, such as the tuning parameter (C) and choosing the appropriate kernel type.

The results obtained from sources [38] and [40] have also been used as a comparative measure. These results show that other algorithms with accuracy, recall and F1 criterion have had acceptable performance of 87.6%, 89.6% and 90.01% respectively in [38] and 92.38%, 93.11% and 94.87% in [40], but still their performance It was less than the proposed method (CNN).

The results show that the proposed method using convolutional neural networks (CNN) has been able to obtain the best accuracy, readability and F1 criterion among the investigated algorithms. This shows the high power of CNN in identifying and learning complex patterns in traffic data. On the other hand, more traditional algorithms such as KNN, DT and SVM have also performed significantly, but could not reach the accuracy of the proposed method.

According to these results, the use of convolutional neural networks (CNN) as the proposed method in this research is a suitable choice and can help improve the accuracy and efficiency of traffic pattern detection systems in software-based networks. This method has many capabilities in analyzing complex data and extracting important features, which has made it a powerful tool in the field of machine learning.

Finally, the results of this research can pave the way for future research in the field of improving machine learning models to analyze and manage software-based network traffic. Using more advanced deep learning techniques, combining different models and improving data preprocessing processes can lead to achieving higher accuracy and efficiency in identifying traffic

patterns. In this way, the security and efficiency of SDN networks will be significantly improved.

## 5.2 Future works

can lead to the development and improvement of current methods and open new horizons in the field of using machine learning and deep learning in software-based network traffic analysis and management. Therefore, continuing research in this field and applying new techniques can have positive effects on the security and efficiency of SDN networks.

In general, the findings of this research show the importance and efficiency of using advanced deep learning models, especially convolutional neural networks (CNN) in analyzing and identifying traffic patterns in software-based networks (SDN). In this section, research limitations and suggestions for future research are discussed.

## Reference

- [1] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. V. Vasilakos, "An effective network traffic classification method with unknown flow detection," *IEEE transactions on network and service management*, vol. 10, no. 2, pp. 133-147, 2013. [10.1109/TNSM.2013.022713.120250](https://doi.org/10.1109/TNSM.2013.022713.120250)
- [2] J. Frank, "Artificial intelligence and intrusion detection: Current and future directions." pp. 1-12. <http://dx.doi.org/10.14514/BYK.m.26515393.2022.10/1.78-87>
- [3] Cotton, Michelle, et al. Internet assigned numbers authority (IANA) procedures for the management of the service name and transport protocol port number registry. No. rfc6335. 2011.
- [4] J. V. Gomes, P. R. Inácio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Detection and classification of peer-to-peer traffic: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, pp. 1-40, 2013. <https://doi.org/10.1145/2480741.2480747>
- [5] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms." pp. 2451-2455. [10.1109/CompComm.2016.7925139](https://doi.org/10.1109/CompComm.2016.7925139)
- [6] S. Katal, and A. P. H. Singh, "A Survey of Machine Learning Algorithm in Network Traffic Classification," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 9, no. 6, 2014. [10.14445/22312803/IJCTT-V9P157](https://doi.org/10.14445/22312803/IJCTT-V9P157)
- [7] Y. D. Goli, and R. Ambika, "Network traffic classification techniques-a review." pp. 219-222. [https://doi.org/10.1007/978-981-19-8493-8\\_29](https://doi.org/10.1007/978-981-19-8493-8_29)
- [8] T. T. Nguyen, and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56-76, 2008. [10.1109/SURV.2008.080406](https://doi.org/10.1109/SURV.2008.080406)
- [9] J. Kennedy, and R. Eberhart, "Particle swarm optimization." pp. 1942-1948. [10.1109/ICNN.1995.488968](https://doi.org/10.1109/ICNN.1995.488968)



- [10] M. Kubat, An introduction to machine learning: Springer, 2017. <https://doi.org/10.1007/978-3-319-63913-0>
- [11] H. Wu, and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," IEEE Transactions on Image Processing, vol. 27, no. 3, pp. 1259-1270, 2017. <https://doi.org/10.1109/TIP.2017.2772836>
- [12] H.-K. Lim, J.-B. Kim, K. Kim, Y.-G. Hong, and Y.-H. Han, "Payload-based traffic classification using multi-layer lstm in software defined networks," Applied Sciences, vol. 9, no. 12, pp. 2550, 2019. <https://doi.org/10.3390/app9122550>
- [13] C. Xu, H. Lin, Y. Wu, X. Guo, and W. Lin, "An SDNFV-based DDoS defense technology for smart cities," IEEE Access, vol. 7, pp. 137856-137874, 2019. <https://doi.org/10.1109/ACCESS.2019.2943146>
- [14] B. Park, J. W.-K. Hong, and Y. J. Won, "Toward fine-grained traffic classification," IEEE Communications Magazine, vol. 49, no. 7, pp. 104-111, 2011. <https://doi.org/10.1109/MCOM.2011.5936162>
- [15] X. Chang, F. Nie, Y. Yang, and H. Huang, "A convex formulation for semi-supervised multi-label feature selection." <https://doi.org/10.1609/aaai.v28i1.8922>
- [16] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway," IEEE Access, vol. 6, pp. 55380-55391, 2018. <https://doi.org/10.1109/ACCESS.2018.2872430>
- [17] C. Zhang, X. Wang, F. Li, Q. He, and M. Huang, "Deep learning-based network application classification for SDN," Transactions on Emerging Telecommunications Technologies, vol. 29, no. 5, pp. e3302, 2018. <https://doi.org/10.1002/ett.3302>
- [18] Z. Fan, and R. Liu, "Investigation of machine learning based network traffic classification." pp. 1-6. <https://doi.org/10.1109/ISWCS.2017.8108090>
- [19] M. M. Raikar, S. Meena, M. M. Mulla, N. S. Shetti, and M. Karanandi, "Data traffic classification in software defined networks (SDN) using supervised-learning," Procedia Computer Science, vol. 171, pp. 2750-2759, 2020. <https://doi.org/10.1016/j.procs.2020.04.299>
- [20] F. A. M. Zaki, and T. S. Chin, "FWFS: Selecting robust features towards reliable and stable traffic classifier in SDN," IEEE Access, vol. 7, pp. 166011-166020, 2019. <https://doi.org/10.1109/ACCESS.2019.2953565>
- [21] A. Malik, R. de Fr  in, M. Al-Zeyadi, and J. Andreu-Perez, "Intelligent SDN traffic classification using deep learning: Deep-SDN." pp. 184-189. <https://doi.org/10.1109/ICCCI49374.2020.9145971>
- [22] P. Wang, S.-C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs." pp. 760-765. <https://doi.org/10.1109/SCC.2016.133>
- [23] M. Amiri, H. Al Osman, and S. Shirmohammadi, "Game-aware and sdn-assisted bandwidth allocation for data center networks." pp. 86-91. <https://doi.org/10.1109/MIPR.2018.00023>
- [24] F. Kiyomarsi , B. Zamani ,Extending the Lifetime of Wireless Sensor Networks Using Fuzzy Clustering Algorithm Based on Trust Model, Journal of Optimization in Soft Computing , Issue 1 , 2023. <https://doi.org/10.82553/josc.2023.14020714783332>
- [25] P. Khosravian Dehkordi A Comprehensive Review on Service Function Chaining in Network Environments Journal of Optimization in Soft Computing , Issue 3, 2024. <https://doi.org/10.82553/josc.2024.140212161104657>



## Review of Machine Learning Algorithm in Medical Health

Zahra Ghorbani<sup>1</sup>, Sahar Behrouzi-Moghaddam<sup>1</sup>, Shahram Zandiyan<sup>2</sup>, Babak Nouri-Moghaddam<sup>1</sup>, Nasser Mikaeilvand<sup>3</sup>, Sajjad Jahanbakhsh<sup>4</sup>, Ailin Khosravani<sup>1</sup>, Fatemeh Tahmasebizadeh<sup>1</sup>, Abbas Mirzaei<sup>1\*</sup>

*1 Department of Computer Engineering, Ard.C., Islamic Azad University, Ardabil, Iran*

*2 Department of Computer Science and Mathematics, Central Tehran branch, CT.C., Islamic Azad University, Tehran, Iran*

*3 Department of Computer Engineering, ST.C., Islamic Azad University, Tehran, Iran*

*4 Department of Computer Engineering, Germi.C., Islamic Azad University, Germi, Iran*

### Article Info

#### Article History:

Received: 2025/04/13

Revised: 2025/06/10

Accepted: 2025/06/20

DOI:

#### Keywords:

Machine Learning; Medical;

Supervised Learning;

Classification.

\*Corresponding Author's Email  
Address: [a.mirzaei@iau.ac.ir](mailto:a.mirzaei@iau.ac.ir)

### Abstract

Recently, health-related data has been analyzed using a variety of cutting-edge methods, including artificial intelligence and machine learning. The application of machine learning technologies in the healthcare industry is enhancing medical professionals' proficiency in diagnosis and treatment. Researchers have extensively used medical data to identify patterns and diagnose illnesses. Nevertheless, little research has been done on using machine learning algorithms to enhance the precision and usefulness of medical data. An extensive analysis of the many machine learning methods applied to healthcare applications is given in this work. We first examine supervised and unsupervised machine learning techniques, and then we investigate the applicability of time series tasks on historical data, evaluating their appropriateness for datasets of varying sizes.

### 1. Introduction

The healthcare service system plays a crucial role in the medical domain, addressing significant demands on human life. To advance, healthcare providers in developing countries are increasingly adopting intelligent technologies such as artificial intelligence (AI) and machine learning. The integration of AI has spurred advancements in human-centered healthcare systems. AI technologies have notably influenced the development of intensive care and supervisory activities in hospitals and clinics [1-3].

Extensive research by Jafar Abdollahi since 2019 has highlighted the successful application of AI, including machine learning and deep learning, in medical image and healthcare analysis. His research covers a range of conditions such as bupropion, diphenhydramine, breast cancer, medicinal plants, epidemics, stroke, lung cancer, social networks, diabetes, suicides, coronary artery

disease, and more, demonstrating promising results [2-5].

Machine learning, an automated process that enables computers to learn and improve performance without explicit programming, is central to these advancements. Unlike systems reliant on preset rules, machine learning utilizes complex algorithms and statistical techniques to analyze data and make accurate predictions. The dataset's quality is critical for machine learning accuracy, leading to more precise forecasts as the data improves [3,18].

Machine learning has found applications across various industries, including banking, retail, and healthcare. In healthcare, it offers significant opportunities for disease detection and treatment. One of its key benefits is enhancing the accuracy of data forecasting and classification, which is particularly valuable in medical analysis. As more data is collected, the prediction model's ability to make precise decisions improves.

Overall, the healthcare service system is vital in addressing human needs, and advanced technologies like AI and machine learning are instrumental in advancing and refining healthcare services. The integration of AI has led to significant developments in human-centered healthcare systems. Since 2019, Jafar Abdollahi's research has demonstrated the successful application of AI in diagnosing various diseases and analyzing medical images, yielding encouraging results across multiple conditions.

## 2. OVERVIEW OF MACHINE-LEARNING IN HEALTHCARE

Machine learning, a branch of artificial intelligence, focuses on using data to train algorithms so they can act or anticipate without explicit programming. Machine learning has the ability to completely change how the healthcare sector recognizes, treats, and prevents diseases, as seen in Figure 1. The following are a few possible uses of machine learning in the medical field [4,19]:

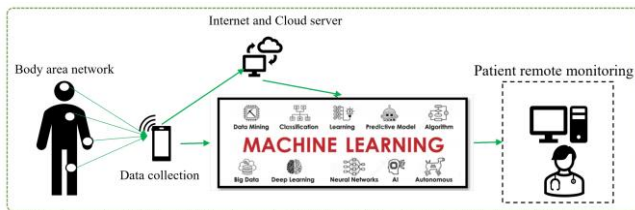


Fig 1 illustrates, machine learning has the power to radically alter how we identify, manage, and prevent diseases in the healthcare industry [19].

- A. *Predictive analytics:* Utilizing information from claims data, electronic health records, and other sources, machine learning algorithms can forecast the probability of certain health outcomes, like hospital readmissions or the onset of chronic illnesses. This capacity permits early preventative treatments and enables medical practitioners to identify individuals who are more likely to have negative effects [6,20].
- B. *Diagnosis and treatment:* Diagnoses and the best course of treatment for a patient can be made with the assistance of machine learning algorithms that have been trained on medical images, such as CT scans and X-rays [5,21].
- C. *Personalized medicine:* Using unique patient variables like genetics and medical history, machine learning may be used to forecast which drugs a patient is most likely to react to [7, 22].
- D. *Clinical decision support:* Clinical decision support systems may incorporate machine

learning algorithms to help medical personnel make better decisions about patient care [8, 23].

- E. *Population health management:* Data from huge populations may be analyzed using machine learning to find trends and patterns that can guide the creation of public health programs. All things considered, using machine learning to healthcare might lead to better patient outcomes, lower costs, and increased system efficiency [9, 24].

## 3. REVIEW OF MACHINE LEARNING

The two main subcategories of machine learning are supervised learning and unsupervised learning, as seen in Figure 2. In order to forecast future results, supervised learning algorithms are trained with input and output data from previous occurrences. Unsupervised learning algorithms, on the other hand, find underlying structures or hidden patterns in the given data without the need for pre-existing labels. Unsupervised learning mostly concentrates on clustering tasks, whereas supervised learning is appropriate for both classification and regression problems [10- 13].

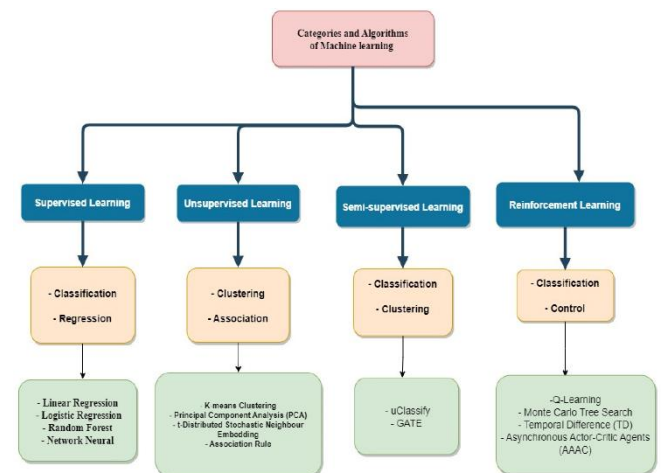


Fig 2. Supervised learning and unsupervised learning are the two primary subcategories of machine learning [13].

Classification algorithms, which predict categorical outcomes, are a subset of supervised machine learning approaches. Unlike unsupervised learning, supervised learning relies on known and labeled training data. The data is divided into training and testing sets [14-17]. Classification algorithms sort incoming data into distinct categories to make predictions. Supervised machine learning is commonly applied in fields such as speech recognition, medical image interpretation, and heart attack prediction [7, 18].



Using the supplied training data, categorization models are created in supervised learning. Then, more unlabeled data can be classified by these models. One output variable from the training dataset needs to be categorized. In order to categorize the test data, classification algorithms first recognize unique patterns in the training data [13]. Neural networks, decision trees, naïve Bayes, K-nearest neighbors, and support vector machines are examples of common classification techniques.

### A. Supervised Machine Learning

#### • Decision Trees(DT)

A decision tree classifier uses a tree-like diagram to illustrate possible outcomes, final values, and options. This method involves computing the probabilities of selecting different actions through a computer algorithm. The process begins with samples of training data and their associated category labels. The decision tree method recursively partitions the training data into subsets based on feature values, resulting in subgroups with more homogeneous data compared to the parent set [19, 25].

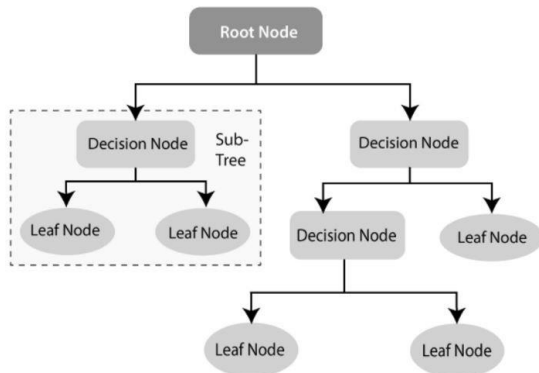


Fig 3. Visual illustration of the DT algorithm [13]

In a decision tree, every internal node denotes a test feature, every branch node shows the test's outcome, and every leaf node shows the class label. The decision tree classifier classifies an unknown sample using the route from a root node to a leaf node, and it utilizes this path to derive the category label [15–17].

#### • Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a classical machine learning method used for addressing classification problems. SVM plays a vital role in supporting a wide range of applications in extensive data mining environments [30]. It leverages specific characteristics of a model to train data and generate accurate predictions from a given dataset [20–22].

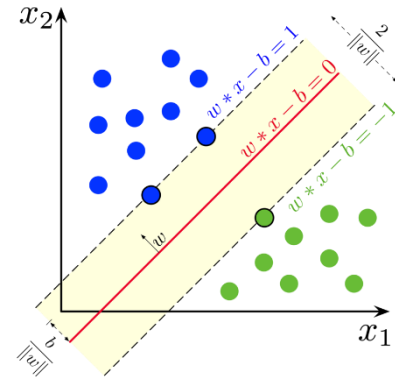


Fig 4. Visual illustration of the SVM algorithm [22].

Support Vector Machine's mathematical intuition: Think of a binary classification task where there are two classes, denoted by the labels +1 and -1. The input feature vectors (X) and the matching class labels (Y) comprise our training dataset. The equation for the linear hyperplane can be written as:

$$w^T x + b = 0 \quad (1)$$

The direction perpendicular to the hyperplane, or the normal vector, is represented by the vector W. The offset, or distance, of the hyperplane from the origin along the normal vector w is represented by the parameter b in the equation. The distance between a data point  $x_i$  and the decision boundary can be calculated as:

$$d_i = \frac{w^T x_i + b}{\|w\|} \quad (2)$$

where  $\|w\|$  represents the Euclidean norm of the weight vector w. Euclidean norm of the normal vector W.

#### • Naïve Bayes (NB)

An approach that is frequently used for classification jobs is the Naïve Bayes algorithm. It is one of the most basic types of Bayesian networks since it is predicated on the idea that there is a single parent node with a finite number of independent child nodes. As shown in Figure 6, the Naïve Bayes technique multiplies the individual probabilities of each attribute-value combination to determine the probability of a classification. This approach works incredibly well in cases where the qualities are independent. The Naïve Bayes method's effective computational training time is one of its main benefits. The classification performance of the algorithm can also be improved by eliminating unnecessary characteristics [23-26].

$$P(y = 1 | x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{n-1} x_{n-1})}} \quad (8)$$

- $P(c | x)$  = Posterior probability the probability of class C given the features X.
- $P(X | C)$  = Likelihood the probability of the features X given the class C.

- $P(c)$  = Class Prior Probability the probability of the class  $C$  occurring.
- $P(x)$  = Predictor Prior Probability the probability of the features  $X$  occurring.

- **K-Nearest Neighbours (K-NN)**

In data mining classification technology, the K-nearest neighbors (K-NN) classification technique is a straightforward and intuitive method. The K-NN algorithm operates on the principle that an unknown pattern can be classified by considering the K nearest neighbors. By specifying a value for K, the algorithm identifies the category based on the majority class of the K training samples most similar to the unknown pattern. Factors such as the chosen K-value and the distance metric play crucial roles in the performance of the classifier [27].

Euclidean=

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (4)$$

$$\text{Manhattan} = \sum_{i=1}^k |X_i - y_i| \quad (5)$$

$$\text{Minkowski} = \left\{ \sum_{i=1}^k (|X_i - y_i|) \right\}^{1/q} \quad (6)$$

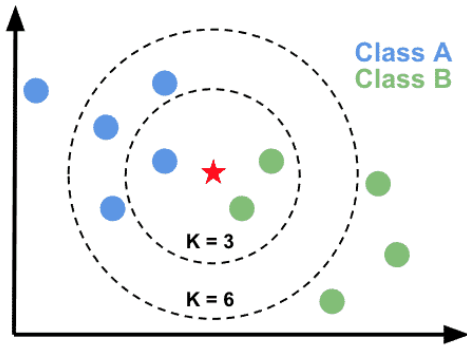


Fig 5. Visual illustration of the KNN algorithm [27].

One advantage of the K-NN method is its relatively low training time compared to other machine learning algorithms. However, it may require more computational time during classification. Despite this, K-NN is favored for its simplicity and ease of use in classification tasks. It is particularly effective when dealing with datasets that have multiple class labels. Additionally, the data training phase of K-NN tends to be faster than that of other machine learning algorithms [27, 28].

- **Linear Regression (LR)**

Linear regression is a straightforward and commonly used method for quantifying the relationship

between response variables and continuous predictors. Its simplicity makes it an optimal choice for analyzing small datasets with high accuracy, as it is relatively easy to understand and interpret. However, if there is an excessive number of predictor variables, the model may struggle to produce reliable results and might not provide the desired outcome [29-31].

$$y = \beta_0 + \beta_1 x + \epsilon \quad (7)$$

Where:

- The dependent variable, often known as the target or outcome variable, is  $Y$ .
- The independent variable, often known as the predictor or feature, is  $x$ .
- The value of  $y$  when  $x=0$  is the regression line's intercept, or  $\beta_0$ .
- The regression line's slope, or the change in  $y$  for every unit change in  $x$ , is  $\beta_1$ .
- The error term, denoted by  $\epsilon$ , is the discrepancy between the observed and model-predicted values.

- **Logistic Regression (LR)**

Unlike linear regression, which predicts continuous data, logistic regression is primarily used for predicting discrete class labels. In classification problems, logistic regression estimates the probability of a sample belonging to one of two possible categories. This is achieved by applying a logistic function, which maps the predicted values to a binary outcome of either 0 or 1. Consequently, logistic regression can indicate the category to which a sample belongs based on the output variable. Researchers have utilized logistic regression to predict health-related behaviors [32-35].

$$P(y = 1 | x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}} \quad (8)$$

Where:

- $P(y=1|x)$  is the probability that the dependent variable  $y$  is 1 given the independent variables  $x_1, x_2, \dots, x_n$ .
- $\beta_0$  is the intercept (the bias term).
- $\beta_1, \beta_2, \dots, \beta_n$  are the coefficients for the independent variables  $x_1, x_2, \dots, x_n$ .
- $e$  is the base of the natural logarithm (approximately equal to 2.718).

- **Ensemble Methods**

Ensemble methods leverage the strengths of multiple machine learning algorithms rather than relying on a single algorithm. By combining and integrating various models, ensemble approaches enhance the overall learning process. One key advantage of ensemble methods is their ability to achieve high

predictive accuracy, which can be superior to that of individual models. However, this increased accuracy often comes at the cost of a more complex training process, which can impact efficiency [36, 37].

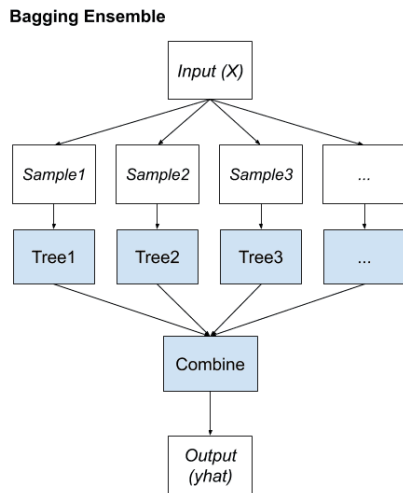


Fig 6. Visual illustration of the Ensemble algorithm [36].

Currently, two common types of ensemble learning techniques are bagging-based methods and boosting-based methods. For instance, Random Forest is a representative algorithm of bagging, while Adaboost, Gradient Boosting Decision Trees (GBDT), and XGBoost are examples of boosting-based algorithms [38, 39].

### • Support Vector Regression (SVR)

An examination of the connection between one or more independent variables and a continuous dependent variable is done using the supervised regression approach known as support vector regression (SVR).

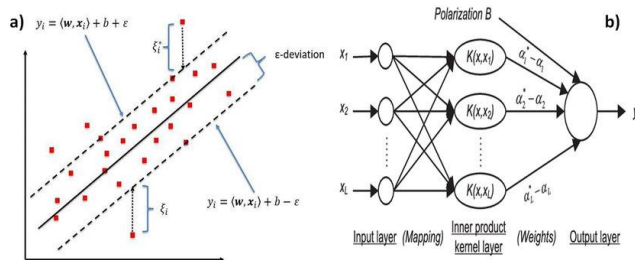


Fig 7. Visual illustration of the SVR algorithm.

While linear regression techniques depend on certain model assumptions, support vector regression (SVR) focusses on identifying the significance of variables in order to describe the connection between inputs and outputs. By keeping the inaccuracy within a certain tolerance margin, this method improves the modelling and prediction of continuous data [33].

### B. Unsupervised Machine Learning

Unsupervised machine learning techniques use sophisticated models with millions of parameters to

analyze vast quantities of unlabeled data in a highly non-linear manner. These methods are popular tools for clustering and exploratory data analysis, allowing for the discovery of hidden patterns within the data. Unlike supervised learning, which relies on labeled data, unsupervised learning draws inferences from datasets that lack explicit output labels. Key applications of unsupervised learning include market research, item recognition, and DNA sequence analysis [34].

Putting incoming data into meaningful categories based on similarities and features rather than predetermined labels is the fundamental idea behind unsupervised learning. This is grouping data according to innate patterns instead of precise categorizations. Hard clustering and soft clustering are the two primary categories of clustering techniques. While soft clustering permits data points to belong to numerous clusters with differing degrees of membership, hard clustering allocates each data point to a single cluster. Popular techniques for unsupervised machine learning are covered in the section that follows [40, 41].

#### A. K-Means

K-means is a well-liked unsupervised learning method that is effective and straightforward for handling clustering issues. By minimizing the total squared distances between each point and the centroid of its designated cluster, the K-means method divides data points into k clusters. This technique is popular for a variety of clustering applications because it effectively divides data into clusters with low intra-cluster variance [42, 43].

#### B. K-Medoids

Unlike K-Means, which uses the mean value of data points in a cluster as a reference point, K-Medoids employs actual data points as the central objects, or medoids, to determine cluster centers. K-Medoids assigns each data point to the nearest medoid and builds clusters around these central objects. Although K-Medoids can produce conflicting results depending on the initial medoids, it is less sensitive to outliers and can adapt cluster memberships more effectively than K-Means [44, 45].

### C. Using Hierarchical Grouping

One popular technique in data mining for cluster analysis is hierarchical cluster analysis (HCA), also referred to as hierarchical clustering. By comparing the traits inside each cluster, it seeks to establish a hierarchical structure of clusters. This methodology creates tiered sets of clusters repeatedly, resulting in a diagram that resembles a tree called a dendrogram. The relationships between data points and clusters are visually represented by the dendrogram, where each level denotes a distinct stage of cluster development [46-49].

**Table 1.** Gives a quick overview of the pros and cons of each algorithm in a clear and concise manner [46-49].

SUPERVISED			UNSUPERVISED		
Algorithms	Pros	Cons	Algorithms	Pros	Cons
DT	Easy to interpret, handles both categorical and numerical data, works well with non-linear data.	Prone to overfitting, especially with noisy data.	K-Means	Fast, scalable, and works well with large datasets. Effective for spherical clusters.	Sensitive to initial centroids and outliers, struggles with clusters of varying sizes or densities, and assumes spherical clusters.
SVM	Effective in high-dimensional spaces, robust to outliers, and works well with clear margin separation.	Computationally intensive, less effective with large datasets, and difficult to interpret.	K-Medoids	More robust to outliers and noise compared to K-Means, as it uses medoids instead of means.	Slower and more computationally intensive than K-Means, especially with large datasets.
KNN	Simple to implement, no training phase, effective with small datasets.	Computationally expensive with large datasets, sensitive to irrelevant features, and storage-intensive.	Using Hierarchical Grouping	Does not require the number of clusters to be specified, provides a hierarchy of clusters, and can capture complex cluster structures.	Computationally expensive, especially for large datasets, and sensitive to noise and outliers.
Linear Regression	Simple and interpretable, works well with linear relationships, and easy to implement.	Assumes linearity, sensitive to outliers, and may underperform with non-linear data.			
Logistic Regression	Interpretable, works well with binary classification, and can handle linear decision boundaries.	Assumes linearity, struggles with complex relationships, and sensitive to outliers.			
Ensemble Methods	Combines multiple models to improve performance, reduces overfitting, and increases accuracy and robustness.	More complex and computationally expensive, less interpretable, and requires careful tuning.			
SVR	Effective for regression with high-dimensional data, robust against overfitting in high-dimensional spaces.	Similar challenges as SVM, including computational complexity, and less intuitive to interpret.			

The table compares various algorithms, highlighting that **Decision Trees** are easy to interpret and handle different data types but are prone to overfitting. **SVM** is effective in high-dimensional spaces but is computationally intensive. **KNN** is simple and effective with small datasets but struggles with large datasets and irrelevant features. **Linear and Logistic Regression** are interpretable and handle linear relationships well but are limited by their assumption of linearity and sensitivity to outliers. **Ensemble Methods** improve accuracy and reduce overfitting by combining models but are more complex and less interpretable. **SVR** shares SVM's strengths in high-dimensional regression but also its computational challenges. **K-Means** is fast and scalable but sensitive to outliers and initial centroids, while **K-Medoids** is more robust to outliers but slower. **Hierarchical Grouping** captures complex structures without needing a preset number of clusters but is computationally expensive and sensitive to noise.

#### 4. EVALUATION MATRIX OF SUPERVISED CLASSIFICATION ALGORITHMS

Three standard measures are used to assess the performance of supervised classification algorithms: specificity, sensitivity, and accuracy. Specificity is the amount of true negative data points identified in actual negative data points (TP = true positive, TN = true negative, FN = false negative, and FP = false positive); accuracy is the percentage of prediction rate in the model; and sensitivity is the amount of true positive data points correctly identified in actual positive data points [3- 5].

**Accuracy:** A category's accuracy is calculated by dividing its "correct predictions made" total by the number of "total predictions made" by a category that is similar.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (9)$$

**Sensitivity:** Real positive rate: If the individual has a positive result, the model will be positive in a tiny fraction of situations, according to the formula below.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (10)$$

**Specificity:** If the person gets a poor result, it will only happen in a tiny portion of situations. This is calculated with the following formula [50-53].

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (11)$$

#### 5. DISSCUSSION

Healthcare has shown considerable promise for both supervised and unsupervised machine learning

technologies. The applications of these approaches vary based on the type of data and the specific tasks at hand, each with its own advantages and limitations.

With supervised learning, a model is trained using labeled data in order to forecast outcomes based on input features [65-67]. It has been widely applied in the medical field to diagnose, classify, and predict prognoses [68-70]. To predict cardiovascular risk, identify malignant cells, and classify medical images, for instance, supervised learning algorithms such as decision trees, logistic regression, and support vector machines have been used [71-75]. Supervised learning is useful, but it needs a lot of labeled data, and it can be biased if the training set isn't typical of the general population.

On the other hand, unsupervised learning makes use of unlabeled data to train a model that, in the absence of explicit guidance, finds patterns and correlations on its own [14, 15, 16]. This method works well in the medical field for tasks like clustering, anomaly detection, and feature extraction [54-57]. For example, K-means clustering techniques have been used to identify uncommon conditions, extract pertinent information from medical images, and classify patients based on shared features [58-60]. Unsupervised learning, however, occasionally yields outcomes that are difficult to interpret and are not clinically relevant.

Thus, there are clear advantages and disadvantages to both supervised and unsupervised learning in the healthcare industry. The particular task at hand, the type of data, and the resources at hand all influence which of these approaches is best. Machine learning will be essential to enhancing patient outcomes and advancing medical research as long as healthcare data is available.

Machine learning algorithms each have distinct strengths and weaknesses in medical health applications. Decision Trees are easy to understand but can overfit and be biased. Random Forests reduce overfitting but can become complex and resource-intensive. Support Vector Machines (SVM) perform well in high-dimensional spaces but are expensive and hard to interpret. Neural Networks capture complex patterns across various data types but require significant resources and are often seen as a "black box." K-Nearest Neighbors (KNN) is simple and flexible but computationally expensive and sensitive to irrelevant features. Logistic Regression is efficient and interpretable but limited to linear relationships. Gradient Boosting Machines (GBMs) offer high accuracy but are prone to overfitting and are complex to implement. Principal Component Analysis (PCA) reduces dimensionality effectively but may lose important information, while Naive Bayes is efficient but struggles with correlated features due to its assumption of independence [61-64].

**Table 2.** summarizing the strengths and weaknesses of machine learning algorithms in medical health, along with references for each aspect [61-64].

ASPECT	STRENGTHS	WEAKNESSES	REF
Decision Trees	<ul style="list-style-type: none"> <li>- Simple to understand and interpret.</li> <li>- Useful for classification tasks with clear decision rules.</li> </ul>	<ul style="list-style-type: none"> <li>- Prone to overfitting, especially with complex data.</li> <li>- Can be biased towards features with more levels.</li> </ul>	[25]
Random Forest	<ul style="list-style-type: none"> <li>- Reduces overfitting by averaging multiple decision trees.</li> <li>- Handles large datasets with higher accuracy.</li> </ul>	<ul style="list-style-type: none"> <li>- Computationally intensive.</li> <li>- Model can become complex and less interpretable.</li> </ul>	[26]
Support Vector Machines (SVM)	<ul style="list-style-type: none"> <li>- Effective in high-dimensional spaces.</li> <li>- Works well for classification problems with clear margins.</li> </ul>	<ul style="list-style-type: none"> <li>- Memory and computationally expensive.</li> <li>- Difficult to interpret results and tune hyperparameters.</li> </ul>	[27]
Neural Networks (ANNs)	<ul style="list-style-type: none"> <li>- Capable of capturing complex patterns in data.</li> <li>- Suitable for various types of data including images and texts.</li> </ul>	<ul style="list-style-type: none"> <li>- Requires large amounts of data and computational resources.</li> <li>- Can be seen as a "black box" with poor interpretability.</li> </ul>	[28]
K-Nearest Neighbors (KNN)	<ul style="list-style-type: none"> <li>- Simple to implement and understand.</li> <li>- Flexible and adaptable to different types of data.</li> </ul>	<ul style="list-style-type: none"> <li>- Computationally expensive with large datasets.</li> <li>- Sensitive to irrelevant features and the choice of 'k'.</li> </ul>	[29]
Logistic Regression	<ul style="list-style-type: none"> <li>- Provides probabilistic outputs and is easy to interpret.</li> <li>- Computationally efficient for binary classification.</li> </ul>	<ul style="list-style-type: none"> <li>- Assumes a linear relationship between features and the log odds of the outcome.</li> <li>- Limited to binary classification without extensions.</li> </ul>	[30]
Gradient Boosting Machines (GBMs)	<ul style="list-style-type: none"> <li>- Provides high prediction accuracy.</li> <li>- Can handle various types of data and feature interactions.</li> </ul>	<ul style="list-style-type: none"> <li>- Prone to overfitting if not tuned properly.</li> <li>- Computationally expensive and complex to implement.</li> </ul>	[31]
Principal Component Analysis (PCA)	<ul style="list-style-type: none"> <li>- Reduces dimensionality while retaining most variance in data.</li> <li>- Simplifies models and speeds up computation.</li> </ul>	<ul style="list-style-type: none"> <li>- May discard useful information.</li> <li>- Difficult to interpret principal components.</li> </ul>	[32]
Naive Bayes	<ul style="list-style-type: none"> <li>- Simple and efficient for large datasets.</li> <li>- Good performance with small to moderate-sized datasets.</li> </ul>	<ul style="list-style-type: none"> <li>- Assumes feature independence, which may not hold true.</li> <li>- Less accurate with highly correlated features.</li> </ul>	[33]



## 6. CONCLUSION

In conclusion, the application of machine learning algorithms in healthcare offers significant potential to enhance diagnostic accuracy and treatment effectiveness. As highlighted in this review, different algorithms come with their own strengths and limitations. For instance, while decision trees and ensemble methods provide interpretability and improved accuracy through model combinations, they can suffer from overfitting and complexity. On the other hand, algorithms like SVM and SVR excel in handling high-dimensional data but require substantial computational resources. Simpler algorithms such as KNN and logistic regression, though effective in specific contexts, face challenges with scalability and handling non-linear relationships.

Moreover, unsupervised techniques like K-Means and hierarchical grouping offer valuable insights into data patterns without requiring labeled datasets, but they can be sensitive to initial conditions and computationally intensive. The review underscores the importance of selecting the appropriate algorithm based on the specific characteristics of the healthcare data at hand, whether it involves small or large datasets, linear or non-linear relationships, or the need for scalability and robustness. Ultimately, the integration of these machine learning models into healthcare systems must consider these trade-offs to optimize patient outcomes and improve the efficiency of medical practices.

## REFERENCES

- [1] Mehrpour, O., Saeedi, F., Vohra, V., Abdollahi, J., Shirazi, F. M., & Goss, F. (2023). The role of decision tree and machine learning models for outcome prediction of bupropion exposure: A nationwide analysis of more than 14 000 patients in the United States. *Basic & Clinical Pharmacology & Toxicology*, 133(1), 98-110.
- [2] Duan, H., & Mirzaei, A. (2023). Adaptive Rate Maximization and Hierarchical Resource Management for Underlay Spectrum Sharing NOMA HetNets with Hybrid Power Supplies. *Mobile Networks and Applications*, 1-17.
- [3] Abdollahi, J. (2022, February). Identification of medicinal plants in ardabil using deep learning: identification of medicinal plants using deep learning. In *2022 27th International Computer Conference, Computer Society of Iran (CSICC)* (pp. 1-6). IEEE.
- [4] Mirzaei, A., & Najafi Souha, A. (2021). Towards optimal configuration in MEC Neural networks: deep learning-based optimal resource allocation. *Wireless Personal Communications*, 121(1), 221-243.
- [5] Abdollahi, J., Davari, N., Panahi, Y., & Gardaneh, M. (2022). Detection of Metastatic Breast Cancer from Whole-Slide Pathology Images Using an Ensemble Deep-Learning Method: Detection of Breast Cancer using Deep-Learning. *Archives of Breast Cancer*, 364-376.
- [6] Shahriyar, O., Moghaddam, B. N., Yousefi, D., Mirzaei, A., & Hoseini, F. (2025). An analysis of the combination of feature selection and machine learning methods for an accurate and timely detection of lung cancer. *arXiv preprint arXiv:2501.10980*.
- [7] Mikaeilvand, N., Ojaroudi, M., & Ghadimi, N. (2015). Band-Notched Small Slot Antenna Based on Time-Domain Reflectometry Modeling for UWB Applications. *The Applied Computational Electromagnetics Society Journal (ACES)*, 682-687.
- [8] Li, X., Lan, X., Mirzaei, A., & Bonab, M. J. A. (2022). Reliability and robust resource allocation for Cache-enabled HetNets: QoS-aware mobile edge computing. *Reliability Engineering & System Safety*, 220, 108272.
- [9] Abdollahi, J., & Mahmoudi, L. (2022, February). An Artificial Intelligence System for Detecting the Types of the Epidemic from X-rays: Artificial Intelligence System for Detecting the Types of the Epidemic from X-rays. In *2022 27th International Computer Conference, Computer Society of Iran (CSICC)* (pp. 1-6). IEEE.
- [10] Somarin, A. M., Barari, M., & Zarrabi, H. (2018). Big data based self-optimization networking in next generation mobile networks. *Wireless Personal Communications*, 101(3), 1499-1518.
- [11] Amani, F., & Abdollahi, J. (2022). Using Stacking methods based Genetic Algorithm to predict the time between symptom onset and hospital arrival in stroke patients and its related factors. *Journal of Biostatistics and Epidemiology*, 8(1), 8-23.
- [12] Jahanbakhsh Gudakahriz, S., Momtaz, V., Nouri-Moghadam, B., Mirzaei, A., & Vajed Khiavi, M. (2025). Link life time and energy-aware stable routing for MANETs. *International Journal of Nonlinear Analysis and Applications*.
- [13] Abdollahi, J., Keshandehghan, A., Gardaneh, M., Panahi, Y., & Gardaneh, M. (2020). Accurate detection of breast cancer metastasis using a hybrid model of artificial intelligence algorithm. *Archives of Breast Cancer*, 22-28.
- [14] PARVAR, M. E., SOMARIN, A. M., TAHERNEZHAD, M. R., & ALAEI, Y. (2015). Proposing a new method for routing improvement in wireless ad hoc networks (optional). *Fen Bilimleri Dergisi (CFD)*, 36(4).
- [15] Barzaki, M. A. J. Z., Abdollahi, J., Negaresh, M., Salimi, M., Zolfaghari, H., Mohammadi, M., ... & Amani, F. (2023, November). Using Deep Learning for Classification of Lung Cancer on CT Images in Ardabil Province: Classification of Lung Cancer using Xception. In *2023 13th International*

- Conference on Computer and Knowledge Engineering (ICCKE) (pp. 375-382). IEEE.
- [16] Narimani, Y., Zeinali, E., & Mirzaei, A. (2022). QoS-aware resource allocation and fault tolerant operation in hybrid SDN using stochastic network calculus. *Physical Communication*, 53, 101709.
- [17] Abdollahi, J. (2023). Evaluating LeNet Algorithms in Classification Lung Cancer from Iraq-Oncology Teaching Hospital/National Center for Cancer Diseases. *arXiv preprint arXiv:2305.13333*.
- [18] Mirzaei, A. (2022). A novel approach to QoS-aware resource allocation in NOMA cellular HetNets using multi-layer optimization. *Concurrency and Computation: Practice and Experience*, 34(21), e7068.
- [19] Khavandi, H., Moghadam, B. N., Abdollahi, J., & Branch, A. (2023). Maximizing the Impact on Social Networks using the Combination of PSO and GA Algorithms. *Future Generation in Distributed Systems*, 5, 1-13.
- [20] Jahandideh, Y., & Mirzaei, A. (2021). Allocating Duplicate Copies for IoT Data in Cloud Computing based on Harmony Search Algorithm. *IETE Journal of Research*, 1-14.
- [21] Abdollahi, J., NouriMoghaddam, B., & MIRZAEI, A. (2023). Diabetes Data Classification using Deep Learning Approach and Feature Selection based on Genetic.
- [22] Mirzaei, A., Barari, M., & Zarrabi, H. (2019). Efficient resource management for non-orthogonal multiple access: A novel approach towards green hetnets. *Intelligent Data Analysis*, 23(2), 425-447.
- [23] Javadzadeh Barzaki, M. A., Negaresh, M., Abdollahi, J., Mohammadi, M., Ghobadi, H., Mohammadzadeh, B., & Amani, F. (2022, July). USING DEEP LEARNING NETWORKS FOR CLASSIFICATION OF LUNG CANCER NODULES IN CT IMAGES. In *Iranian Congress of Radiology* (Vol. 37, No. 2, pp. 34-34). Iranian Society of Radiology.
- [24] Mirzaei, A., Barari, M., & Zarrabi, H. (2021). An Optimal Load Balanced Resource Allocation Scheme for Heterogeneous Wireless Networks based on Big Data Technology. *arXiv preprint arXiv:2101.02666*.
- [25] Abdollahi, J., Aref, S. Early Prediction of Diabetes Using Feature Selection and Machine Learning Algorithms. *SN COMPUT. SCI.* 5, 217 (2024). <https://doi.org/10.1007/s42979-023-02545-y>.
- [26] Mirzaei, A., & Rahimi, A. (2019). A Novel Approach for Cluster Self-Optimization Using Big Data Analytics. *Information Systems & Telecommunication*, 50.
- [27] Narimani, Y., Zeinali, E., & Mirzaei, A. (2025). A new approach in fault tolerance in control level of SDN. *International Journal of Nonlinear Analysis and Applications*, 16(5), 69-76.
- [28] Rad, K. J., & Mirzaei, A. (2022). Hierarchical capacity management and load balancing for HetNets using multi-layer optimisation methods. *International Journal of Ad Hoc and Ubiquitous Computing*, 41(1), 44-57.
- [29] Amani, F., Abdollahi, J., & Amani, P. (2024, February). Identify the Factors Influencing Suicide among Ardabil city People Using Feature Selection: Identify the Factors Influencing Suicide among Ardabil using machine learning. In *2024 10th International Conference on Artificial Intelligence and Robotics (QICAR)* (pp. 17-23). IEEE.
- [30] Barari, M., Zarrabi, H., & Somarin, A. M. (2016). A New Scheme for Resource Allocation in Heterogeneous Wireless Networks based on Big Data. *Bulletin de la Société Royale des Sciences de Liège*, 85, 340-347.
- [31] Abdollahi, J., & Mehrpour, O. (2024, February). Using Machine Learning Algorithms for Coronary Artery Disease (CAD) Prediction Prediction of Coronary Artery Disease (CAD) Using Machine Learning Algorithms. In *2024 10th International Conference on Artificial Intelligence and Robotics (QICAR)* (pp. 164-172). IEEE.
- [32] Mirzaei, A. (2021). QoS-aware Resource Allocation for Live Streaming in Edge-Clouds Aided HetNets Using Stochastic Network Calculus.
- [33] Javaid, M., Haleem, A., Singh, R. P., Suman, R., & Rab, S. (2022). Significance of machine learning in healthcare: Features, pillars and applications. *International Journal of Intelligent Networks*, 3, 58-73.
- [34] NOSRATIP, M., HOSEINIP, M., SHIRMARZP, A., SOMARINP, A. M., HOSEININIAP, N., BARARIP, M., & Ardebil, I. (2016). Application of MLP and RBF Methods in Prediction of Travelling within the city. *Bulletin de la Société Royale des Sciences de Liège*, 85, 1392-1396.
- [35] Habehh, H., & Gohel, S. (2021). Machine learning in healthcare. *Current genomics*, 22(4), 291.
- [36] Somarin, A. M., Nosrati, M., Barari, M., & Zarrabi, H. (2016). A new Joint Radio Resource Management scheme in heterogeneous wireless networks based on handover. *Bulletin de la Société Royale des Sciences de Liège*
- [37] [19] Swain, S., Bhushan, B., Dhiman, G., & Viriyasitavat, W. (2022). Appositeness of optimized and reliable machine learning for healthcare: a survey. *Archives of Computational Methods in Engineering*, 29(6), 3981-4003.
- [38] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). Metaheuristic and Data Mining Algorithms-based Feature Selection Approach for Anomaly Detection. *IETE Journal of Research*, 1-15.
- [39] Eckerson, W. W. (2007). Predictive analytics. Extending the Value of Your Data Warehousing Investment. *TDWI Best Practices Report*, 1, 1-36.
- [40] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2025). Fraud Prediction in Financial Statements through Comparative Analysis of Data Mining



- Methods. *International Journal of Finance & Managerial Accounting*, 10(38), 151-166.
- [41] Khanna, N. N., Maindarkar, M. A., Viswanathan, V., Fernandes, J. F. E., Paul, S., Bhagawati, M., ... & Suri, J. S. (2022, December). Economics of artificial intelligence in healthcare: diagnosis vs. treatment. In *Healthcare* (Vol. 10, No. 12, p. 2493). MDPI.
- [42] Nematia, Z., Mohammadia, A., Bayata, A., & Mirzaeib, A. (2024). Predicting fraud in financial statements using supervised methods: An analytical comparison. *International Journal of Nonlinear Analysis and Applications*, 15(8), 259-272.
- [43] Mehrpour, O., Saeedi, F., Abdollahi, J., Amirabadizadeh, A., & Goss, F. (2023). The value of machine learning for prognosis prediction of diphenhydramine exposure: National analysis of 50,000 patients in the United States. *Journal of Research in Medical Sciences*, 28(1), 49.
- [44] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). The impact of financial ratio reduction on supervised methods' ability to detect financial statement fraud. *Karafan Quarterly Scientific Journal*.
- [45] Mathur, S., & Sutton, J. (2017). Personalized medicine could transform healthcare. *Biomedical reports*, 7(1), 3-5.
- [46] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). Fraud Risk Prediction in Financial Statements through Comparative Analysis of Genetic Algorithm, Grey Wolf Optimization, and Particle Swarm Optimization. *Iranian Journal of Finance*, 8(1), 98-130
- [47] Berner, E. S. (2007). *Clinical decision support systems* (Vol. 233). New York: Springer Science+ Business Media, LLC.
- [48] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2023). Financial Ratios and Efficient Classification Algorithms for Fraud Risk Detection in Financial Statements. *International Journal of Industrial Mathematics*.
- [49] Swarthout, M., & Bishop, M. A. (2017). Population health management: review of concepts and definitions. *American Journal of Health-System Pharmacy*, 74(18), 1405-1411.
- [50] Nematollahi, M., Ghaffari, A., & Mirzaei, A. (2024). Task offloading in Internet of Things based on the improved multi-objective aquila optimizer. *Signal, Image and Video Processing*, 18(1), 545-552.
- [51] De Ville, B. (2013). Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6), 448-455.
- [52] Hozouri, A., EffatParvar, M., Yousefi, D., & Mirzaei, A. Scheduling algorithm for bidirectional LPT.
- [53] Webb, G. I., Keogh, E., & Miikkulainen, R. (2010). Naïve Bayes. *Encyclopedia of machine learning*, 15(1), 713-714.
- [54] Babazadeh, Z., & Mirzaei, A. A review on methods, ways of decrease delay duties at calculations cloudy.
- [55] Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2), 1883.
- [56] Movahhedi, R., Effatparvar, M., & Somarin, A. M. A study on load balancing methods in cloud computing environment.
- [57] Cunningham, P., & Delany, S. J. (2021). K-nearest neighbour classifiers-a tutorial. *ACM computing surveys (CSUR)*, 54(6), 1-25.
- [58] Ziaeddini, A., Mohajer, A., Yousefi, D., Mirzaei, A., & Gonglee, S. (2022). An optimized multi-layer resource management in mobile edge computing networks: a joint computation offloading and caching solution. *arXiv preprint arXiv:2211.15487*
- [59] Su, X., Yan, X., & Tsai, C. L. (2012). Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(3), 275-294.
- [60] Nokhostin, P., Mirzaei, A., & Jahanbakhsh, S. Proposed Methods for Establishing Load Balancing in Fog Computing: A Survey.
- [61] LaValley, M. P. (2008). Logistic regression. *Circulation*, 117(18), 2395-2399.
- [62] Mehri, R., & Somarin, A. M. (2020). Designing an Energy-Efficient Mechanism to Regulate the Transmission Power Rate in Wireless Sensor Networks. *World*, 9(S1), 189-194.
- [63] Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [64] Mohammad Zadeh, M., & Mirzaei Somarin, A. (2017). Attack Detection in Mobile Ad Hoc.
- [65] Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11, 169-198.
- [66] Zhang, S., Madadkhani, M., Shafieezadeh, M., & Mirzaei, A. (2019). A novel approach to optimize power consumption in orchard WSN: Efficient opportunistic routing. *Wireless Personal Communications*, 108(3), 1611-1634.
- [67] Derakhshandeh, S., & Mikaeilvand, N. (2011). New framework for comparing information security risk assessment methodologies. *Australian Journal of Basic and Applied Sciences*, 5(9), 160-166.
- [68] Mirzaei, A., & Zandiyan, S. (2023). A Novel Approach for Establishing Connectivity in Partitioned Mobile Sensor Networks using Beamforming Techniques. *arXiv preprint arXiv:2308.04797*.
- [69] Allahviranloo, T., & Mikaeilvand, N. (2011). Non zero solutions of the fully fuzzy linear systems. *Appl. Comput. Math*, 10(2), 271-282.
- [70] Awad, M., Khanna, R., Awad, M., & Khanna, R. (2015). Support vector regression. *Efficient learning machines: Theories, concepts, and*

- applications for engineers and system designers, 67-80.
- [71] Javid, S., & Mirzaei, A. (2021). Highly Reliable Routing In Healthcare Systems Based on Internet of Things
  - [72] Hahne, F., Huber, W., Gentleman, R., Falcon, S., Gentleman, R., & Carey, V. J. (2008). Unsupervised machine learning. Bioconductor case studies, 137-157.
  - [73] Nematollahi, M., Ghaffari, A., & Mirzaei, A. (2024). Task and resource allocation in the internet of things based on an improved version of the moth-flame optimization algorithm. Cluster Computing, 27(2), 1775-1797.
  - [74] Kaur, N. K., Kaur, U., & Singh, D. (2014). K-Medoid clustering algorithm-a review. Int. J. Comput. Appl. Technol, 1(1), 42-45.\
  - [75] Navarro-Cerdán, J. R., Sánchez-Gomis, M., Pons, P., Gálvez-Settier, S., Valverde, F., Ferrer-Albero, A., ... & Redon, J. (2023). Towards a personalized health care using a divisive hierarchical clustering approach for comorbidity and the prediction of conditioned group risks. Health Informatics Journal, 29(4), 14604582231212494.



## Mathematical modeling for relocation of terminal facilities in location problems

Mehdi Fazli<sup>1,\*</sup> and Somayyeh Faraji Amoogin<sup>1</sup>

<sup>1</sup> Department of Mathematics, Ardabil Branch, Islamic Azad University, Ardabil, Iran

### Article Info

#### Article History:

Received: 2025/01/07

Revised: 2025/03/17

Accepted: 2025/04/12

DOI:

#### Keywords:

Simulated Annealing,  
Migratory Bird  
Optimization, Terminal  
Facilities, Optimization,  
Taboo Search

\*Corresponding Author's Email  
Address:

mehdi.fazli.s@gmail.com

### Abstract

The main goal of terminal facility layout is to place parking lots, areas and different units within predefined limits in such a way as to minimize the cost of moving passengers and employees. Especially in large-scale terminals containing several different specialized departments, it is important for the efficiency of the terminal that the interaction units are located close together. Today, meta-heuristic algorithms are often used to solve optimization problems such as facility layout. Organized using three meta-heuristic algorithms which are Migratory Bird Optimization (MBO), Taboo Search (TS) and Simulated Annealing (SA), the results were compared with the existing parking scheme. As a result, the MBO and SA meta-heuristic algorithms have provided similar best results, which improve the efficiency of the existing parking scheme by approximately 58%.

## 1. Introduction

The terminal facility arrangement problem has received less attention than other issues in the literature because many perspectives of customer choice cannot be controlled by transportation directors [1] and demand is highly ambiguous in this area [2]. Therefore, more than presenting the complicity of the methods used to examine the issue of terminal deployment, determining the subject solving method, preparing appropriate data and obtaining optimal data are other important points to achieve optimal deployment. The issue of arranging terminal facilities is an NP-hard problem in terms of mathematical calculations [3,4]. There are various methods to solve the facility design problem in this field, whether it is a one-criteria or multi-

criteria problem. One of them is the quadratic allocation problem (QAP) if it considers approximately equal regions for each section and different locations [5]. Next are approximate and heuristic processes that are effective in solving multiple design problems simultaneously [6]. Another study is mixed programming (MP), which uses a general distance-based aim function to design facilities for segments of unequal or equal area [8]. The facility allocation problem is defined by managers as placing a predestined number of possibilities in possible neighborhoods and sizes [9]. Lee and Lee assumed different possibilities in nearby areas to amend the efficiency of facilities in a border area [10]. Shayan and Chitilapili assumed the issue of facility allocation as an optimization issue.

They tried to achieve the optimal arrangement of facilities by considering the interactions between facilities and material handling costs [11]. In the research of recent years, methods based on technological innovation and crowded intelligence have been used to optimally solve facility allocation problems. Shahin and Turkabi developed a new hybrid meta-heuristic technique for solving multiple instrument design problems based on the simulated annealing (SA) method and based by tabu list [12]. By integrating the honey bee (BA) technique, Cheng and Lin obtained a hybrid method for multi-criteria facility design problems. Their methods had global search capability with local research benefits of Particle Swarm Optimization (PSO) at the same time [13]. Using the Ant Colony (ACO) method, Lu et al presented a new model for designing the layout of emergency medical facilities in the city [14]. Eiler used the ACO to locate labs, radiology units, and hospital polyclinics specifically for applicants to minimize running costs. Huynh et al. reviewed the cost estimate of the laboratory. In their research, they operated Automatic Integrated Moving Average (ARIMA) to appraise the number of laboratory applicants. GIS is also used to show the running reality or situation for the distribution of applicants and laboratory costs [16]. Aydin and Fogarty designed a new technique in which SA evolutionary method was applied to solve the classical clinic planning problem and the optimal location problem of disabled hospital facilities [17]. Kaveh and Sharfi used the Search Engine Optimization (CSS) technique, which is based on the interaction among charged particles, to solve the problem of facility allocation in distribution network management [18]. A special coordination search method is presented by Kaveh et al. To find the location of the optimal tool in a given problem [19]. Chan et al perused the alteration measure of fine-tuning genetic technique and the neighbor function of the SA method to solve the problem of locating specific facilities. They used changeable change functions instead of fixed change function or accidentally selected

genes and increased the efficiency of the technique to an acceptable extent [20] .

Yang et al. modeled the efficiency of several meta-heuristic techniques in terms of theoretical methods. Their problem is considered as a p-median problem for base station routing. As mentioned in the text above, several researches have been done in research activities on the problems of allocation of different possibilities, there are only specific studies on the problems related to the allocation of different terminal locations. The purpose of this study is to create a specific facility design for the terminals in order to minimize the overall costs of the system. The cost of moving travel applicants can be reduced by minimizing travel between different establishments. The optimal arrangement problem of our terminal facilities is to investigate the best possible arrangement of different parts in a hypothetical range. So that the distance between the parts that are logically connected to each other is reduced. In this regard, we created a mathematical method for the optimal location of different parts of the terminal with the lowest possible cost. Different alternative techniques of applied strategies were created and the most possible and best option was used. As a new paper, the simulated annealing (SA) optimization technique is used for the first time to solve the problem of assigning facilities to a terminal. Local search is used in SA method. Aiming to measure the performance of the SA technique in real-world problems, the SA results are measured with effects derived from the Tabu Search (TS) and MBO techniques in the given problem. The MBO technique was chosen due to its simplicity and good application in finding reasonable points. The TS technique avoids local optimization and has a more general search. The results of the assignment of terminal components showed that the MBO technique shows an acceptable performance for solving the problem of assigning components of a single terminal. Also, the rest of the article is organized as follows. The definition of the problem of assigning the components of a terminal and the statistical information of this problem are presented in

Section 2. The meta-heuristic techniques of TS, SA and MBO are detailed in Section 3. In the next section, we will solve the problem and in section 5, the calculation results are described. In Section 6, the paper ends with a conclusion and discussion.

## 2. Model

The design and layout of office facilities should include all necessary requirements such as entities, mathematical modeling, difficulty of movement, vertical movement and arrangement of sections within the pre-defined limits by Illery [15], in a way that takes into account the distance between different components and takes care of the customer's demand. In order to achieve the efficient and optimal positioning of departments in this matter, the distance and the number of guidelines for the distance between different departments should be considered, which depends on the distance traveled between the components. The distance between two points determines the connection coefficient, and the parts with high traffic should be placed next to each other, and the parts with the lowest density can be far from each other. We investigated and modeled these factors to reduce the target performance and overall system cost.

1- multi-station interaction, which depends on the number of passengers moving between two stations, should be considered so that stations with higher traffic are closer to stations that are quieter.

2- The number of customers walking to each station was calculated in such a way that the stations with more customers are closer to the main entrance of the customer stations.

3- The final transportation cost was calculated directly based on the degree of travel difficulty, distance, travel frequency, and basic travel cost, and therefore, changing each of these variables caused a change in the transportation cost.

4- The final acceptable response of the system is obtained by performing simulations with the aim of improving one or a number of variables of the problem.

To solve this position allocation problem, we form a special objective function. While paying attention to the physical dimensions of the terminal, the number of kiosks is proportional to the required area, and the average daily data of a season is considered. In the special terminal system, the number of customers and the number of guides is specified.

The physical structure of the terminal shown in Figure 1 includes three blocks, thirty-one applicable parking areas with an area of 800 square meters, and a terminal entrance. There are twenty-six zones of different sizes in the terminal.

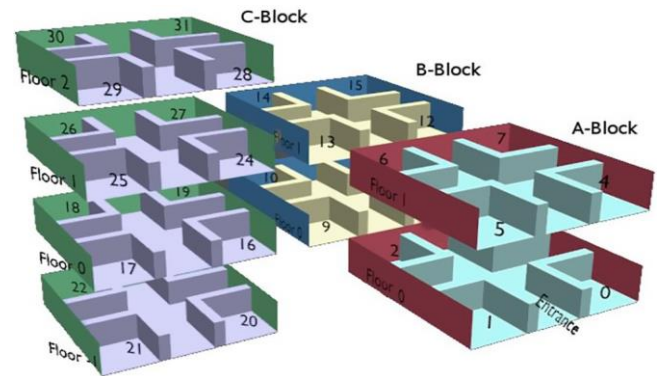


Fig. 1. Physical structure of the terminal.

The main structure of the terminal consists of several blocks, a parking lot with a defined area and a main terminal entrance. So, there are several parts of different sizes in the terminal. The size and distance of the main entrance of the terminal to the areas to be located are given in Table 1.

According to the assumptions explained in the text, Table No. 1 is presented based on the entities of the problem and the objective function of the project is expressed in the form of a special equation.

$$\min(\sum_{i=0}^n x_i y_i + \sum_{j=0}^m \sum_{k=0}^n q_{jk} s_{jk}) \quad (1)$$

In this formula,  $m$  is the number of parking spaces,  $n$  is the number of zones where the parking lots are located,  $x_i$  is the distance between the possible zones and the main entrance of the terminal,  $y_i$  is the number of weekly customers in the parking lot. located up to the  $i$ th place. ,  $q_{jk}$  is the number of customers from the parking lot who were sent

to the location  $j$  to the parking lot that is placed in the return range.  $s_{jk}$  is the distance between region  $j$  and return. The  $q_{jk}$  matrix can be measured as a principal matrix in QAP topics. This matrix consists of different service values, which is the number of trips inside the terminal.

**Table 1**

The size of the areas and the distance to the terminal entrance		
Area Code	Distance to the station	Size
0	20	900
1	20	900
2	30	900
3	30	900
4	70	900
5	70	900
6	80	900
7	80	900
8	120	900
9	120	900
10	130	900
11	130	900
12	170	900
13	170	900
14	180	900
15	180	900
16	220	900
17	230	900
18	230	900
19	230	900
20	250	900
21	260	900
22	260	900
23	250	900
24	270	900
25	280	900
26	280	900
27	270	900
28	320	900
29	330	900
30	330	900
31	320	900

Our problem had two obvious limitations. First, the terminal facility allocation should take into account overall system costs related to customer handling and congestion vehicle handling, but we did not consider the number of vehicles because reliable information was hard to come by. Second, the terminals were

not considered by the staff, because their movement directly depends on the needs of the primary travel establishments.

### 3. Methods

#### 3.1 The MBO migratory bird optimization technique

The MBO migratory bird optimization technique for solving problems was investigated by Duman et al. in 2012 [22]. The MBO technique is inspired by the V formation to optimally use the energy of migratory birds in flight. This technique is often designed for discrete problems and has been implemented and investigated on QAP problems that are based on actual-world questions.

The most optimal method of migration for migratory birds is the V formation so that they can travel longer distances without getting tired. It is clear that in this technique the main instinct of this formation is to save the total energy. The commander bird is the member that consumes the maximum energy in the V organization. The rest of the members can fly for a longer time due to the wind energy generated by the movement of the bird's wings in front of them.

The MBO technique is used in many problems to solve many cases such as the flow storage problem [23-28], also in the backpack problem [33], or in the credit card fraud detection problem [29] and in the case of the traveling salesman [31,32], two-way partitions are used [33], this technique is also used to balance the U-shaped robotic assembly line [34].

The MBO method and algorithm starts with the initial answers that are randomly generated and then tries to improve the obtained answers at each step. The permutation, insertion and inversion process is used to generate candidate solutions for neighbors in order to improve existing paths. The resulting candidate close responses are then compared to the response the technique is looking for. Each current answer is checked against the best adjacent answer. If the adjacent answer is better than the running answer, the neighbor's answer is replaced with the running answer.

Furthermore, leader displacement is performed at defined times to provide optimal responses to both sides of the population. The MBO technique and algorithm include generation of relevant initial community, neighbor sharing operation, appropriate response descendant and leader displacement steps.

### 3.1.1. Generation a main solution

The MBO technique and algorithm examines the initial answers that are randomly selected and tries to optimize the current solution. One of the candidate birds is chosen as the commander bird and the other birds are resting on the left and right side of the commander bird. Therefore, the initial population is marked as described in the method. where the primary congregation of each bird represents an initial response.

### 3.1.2. Generation an acceptable answer

In the MBO technique, an inversion process is used to generate input points for the operation of searching for the appropriate position. In this particular technique, candidate neighboring solutions are obtained completely randomly from the current solution by swapping two selected points. The number of potential suitable answers for the front bird and the number of optimal answers for the candidate bird are different according to the process of the MBO technique. The number of optimal answers is obtained from equation (2) - (4) below [32] .

$$m \geq 3; m = \{3,5,7,9, \dots\}; k \in N^+ \quad (2)$$

$$1 \leq q \leq \frac{m-1}{2}; q \in N^+ \quad (3)$$

$$m - q = r \quad (4)$$

where  $m$  is the number of neighboring answers of the front bird,  $r$  is the number of neighboring answers of other birds, besides for the front bird,  $q$  is the number of neighboring points.

A sample of optimal neighbor answer generation by the permutation process is shown in Figure 2 [32].

### 3.1.3. Neighbor subscription operation

In the MBO technique, the neighbor sharing operation and process is the special

feature of this method that makes the method stand out from similar meta-heuristic techniques. This special neighborhood feature ensures the interaction of all members of the bird flock. The front bird neighbor responses are shared on the left and right sides of the rodents. Neighboring responses of other members are shared only for themselves. According to the corresponding equation (2),  $m$  produces a neighboring solution for the front member. These available answers are measured based on the target performance of the system and descend from the best answer to the worst. The fit point is measured by the running response of the front member. If the neighboring point is better than the running answer, the neighboring answer takes its place. Solutions  $q$  the remaining point is moved to the left bird of the front member. After this operation, the remaining point responses (if such a neighborhood exists) are discarded. The neighborhood responses for the left member of the front point are substituted and the  $p$  responses from the front point are added. These  $q + r$  responses are measured and sorted from the best member to the least favorable member. The best answer of each point's neighbor is compared with the existing answer of this bird. If the neighboring point's answer is better than the current answer, the neighboring answer of that point replaces the existing answer.

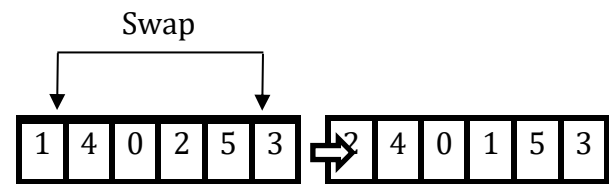


Fig. 2. Neighbor solution generation.

The remaining  $q$  responses are passed to the next bird and other neighboring responses are removed. This action is also performed on the other side or the right side of the group of birds. Neighborhood subscriptions run until the end of the bird collection .

### 3.1.4. Replacing the leader bird

In the MBO algorithm, a flap parameter  $\alpha$  is used to keep indi

sequence for a certain period of time. Then der replace processing is applied. First -lea replacement is applied to the left side of the flock. While the leader is sent behind the left side of the flock, another bird that follows the leader is replaced by the leader. Thus, a new and parameter m is reset. sequence is created The next replacement is applied to the right side of the flock. This process is continued until the algorithm is terminated. Leader .replacement process is shown in Fig3 .

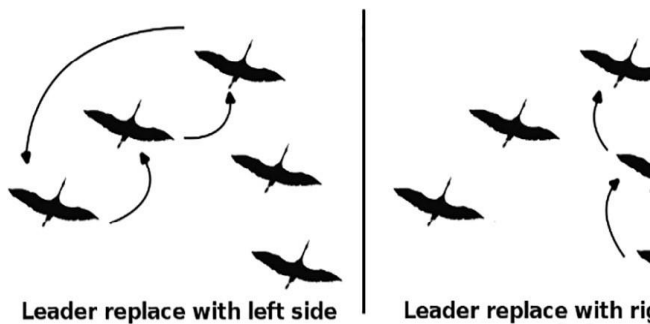


Fig. 3. The leader replacement.

In Algorithm a, the pseudocode of the MBO method is presented as follows:

#### Algorithm a. algorithm MBO

```
Create a random initial set
repetition
repetition
    m Create a neighbor point for the leader member
    Create neighbors of r for the other answer
    Compare the best value of q neighbors with the
    answer of the posterior member
    if (best neighbor answer < current answer value)
        Current answer = Neighborhood answer
    up to the input amount
    Replace the leader member
until termination
Return the best answer in the set
```

### 3.2. Taboo search technique

For use in practical optimization problems, the TS evolutionary technique was first modeled by Glover in 1988 [35]. In the research background, the TS technique has been applied to multitude practical optimization subjects such as the vehicle positioning and routing problem [36] as well

as the traveling salesman project [38] or the special flow storage subject [37] .

In the process related to the TS technique, an initial answer is generated and during the operation, an attempt is made to discover the overall optimal answer using local search methods such as the exchange and insertion operator. In the different memory structure of the TS technique, we have two modes. The structures of short-term memory and long-term memory accept the user to choose the best feasible move to generate the appropriate response and avoid reaching taboo responses. In this method, the TS technique eludes local optimization and seeks the global optimal solution. When the desired amount is attained, the specified taboo is removed and the formerly found answer can be chosen as the new answer. During the search operation for the best point, the optimal answer found by the TS technique is stored in the system memory and all the answers produced by the system are measured with the desired answer. If there is an answer more favorable than the maximum answer, the overall system memory is updated .

In Algorithm b, the pseudocode of the TS method is presented as follows:

#### Algorithm b. algorithm TS

```
Generate the initial answer randomly. This answer
as Choose the current answer and the best optimal
answer.
repetition
    Discover neighboring optimal answers using local
    search techniques.
    Choose the answer that is a non-taboo neighbor.
    Break the norms, even if it is taboo.
    If the resulting answer is better than the existing
    answer, select the new answer as taboo
    If the new answer is better than the best answer, set
    the new answer as the best optimal answer until
    termination
```

### 3.3. Simulated annealing technique

To solve the combined and specific problems, the SA technique was investigated by Kirkpatrick et al. [39]. In the SA technique, it is used to find the optimal solution by avoiding local answers for functions with manifold



variables. The reason for the name of this technique is that it is an example of the complete arrangement of atoms and the minimization of the potential energy of the system when cooling solids. The method of heating a solid body to its melting point and then slowly cooling it to a complete network structure is known as annealing process .

The simulated annealing technique is performed in three specific steps: heating the desired object or material to a certain degree (heating), maintaining that degree for a certain period of time (waiting) and controlled and gradual reduction of temperature (cooling). In the heating process, the particles of solid matter slowly turn into liquid, and when they are properly and gradually cooled, crystalline particles with a completely regular structure are formed.

The structure of this problem and the practical annealing operation are based on the Monte Carlo technique by Metropolis et al. [41]. In the assumed degree of T, the amount of energy of the system is determined based on the following relationship.

$$Q(E) = e^{-E/MT} \quad (5)$$

In this formula, M is Boltzmann's constant and E is the energy of the device.

In case of fluctuations in the overall state of the system, the new power of the system is calculated based on the metropolis technique. The overall stability of a substance with energy E1 is physically created by the displacement of a randomly selected small member, and the amount of energy E2 is replaced by another state. If the amount of energy decreases ( $DE = (E2-E1) < 0$ ), the system will change to this new format. If the energy of the system increases ( $DE > 0$ ), we measure in the new format whether to replace the new energy value E1 according to equation (6) or not. A uniform number is generated in a given interval ( $\gamma \in [0,1]$ ).

$$\gamma \leq e^{E\Delta/T} \quad (6)$$

In this sense, DE is the distinction in several energy levels of different states of matter. These selection criteria are known as basic system processes. Also, based on equation (5), for all energy states in different states, Q (E)

converges to the number 1. It is possible that the system has a high energy level even at low temperatures.

The simulated annealing technique has a favorable application in vehicle routing [44-46]. Also, it has obtained good results in different fields, including the selection of specific features of equipment [47], the traveling salesman problem [45], facility allocation problems [46] and workshop scheduling [48].

In Algorithm c, the pseudocode of the SA method is presented as follows:

#### Algorithm c. algorithm SA

```

Randomly generate the initial answer Si
repetition
Create a neighboring point answer Sn
If (new Sn responds better than Si)
Set Sn as the optimal answer
otherwise (random (0,1) < eΔE/T)
Set Sn as the available answer
Update T
until termination
    
```

#### 4. Application of meta-heuristics in the issue of terminal facility allocation

We deal with a discrete problem in this research. Algorithms used in this article are designed and implemented for discrete problems. In this case, there is no fundamental change in the structure of methods for the process of allocating terminal facilities. To solve this problem, the parking lot codes are randomly arranged. The amount in the array is the parking number. A sample of the layout of this parking lot is shown in Figure 4 .

In Figure 2, parking lot number 3 is located in area 1. After placing all the parking spaces in the available areas, the appropriateness of the answer is evaluated according to function (1). These answers are checked by considering the area of the area where the parking lots are located, the proximity to the main entrance of the terminal, the number of customers to the parking lot, customer consultation between the parking lots and the distance between the parking lots.

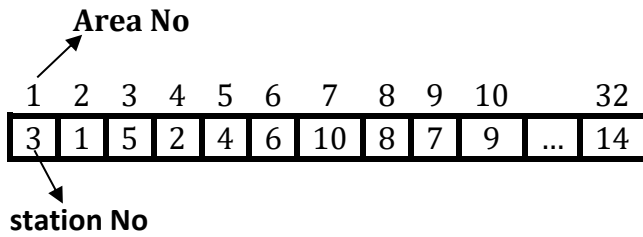


Fig. 4. Permutation coding for an answer

**Example.** Suppose that four different stations (1, 2, 3, 4) are located in five specific regions. Tables 2-4 show the number of station customers, recommendations of inter-station customers, areas to be deployed and main terminal entry time intervals.

In Table 2, the second station is shown as three separate station plots because it has three times the area to be located. The number of customers in Table 2 shows the variable  $y$  in relation (1).

In table number 3, columns 2 and 3 show the same parts of the station (2) and should be placed in the adjacent area. In this case, a very large number of consultations are done for the applicants (600). Table 3 shows the matrix  $z$  in relation (1).

According to the sample answer, station 5, 3, 1, 2 and 4 are located in different regions 1, 2, 3, 4 and 5 respectively.

Table 5 shows matrix  $t$  in the Eq (1). A sample answer is as follows.

5	3	1	2	4
---	---	---	---	---

## 5. Variable settings

In order to reach the best possible solution and cover all aspects in the issue of assigning terminal facilities, we perform parameter settings for all methods.

**Table 2**

The number of customers who come to the station.

Parking No (Name)	Number of applicant (y)	Required size (m2)
(A) 1	14,609	900
(B) 2	55,406	2700
(B) 3		
(C) 4	27,421	900
(D) 5	10,684	900

**Table 3**

The number of consultation clients between stations

	station sending consultations					
	No	1	2	3	4	5
station accepting consultations	1	0	85	85	25	96
	2	14	0	500	50	41
	3	14	500	0	50	41
	4	40	9	9	0	63
	5	33	4	4	5	0

**Table 4**

Distance to the entrance of the station and size of the area

Area No	Size	Distance to the entrance (x)
1	900	15
2	900	15
3	900	25
4	900	25
5	900	35

**Table 5**

Distance between terminal areas

	1	2	3	4	5
1	0	15	20	20	30
2	15	0	20	20	30
3	20	20	0	15	20
4	20	20	15	0	20
5	30	30	20	20	0

In the parameter settings, the meta-heuristic techniques of this research are implemented 30 times and separately from each other, and the variables are selected according to the efficiency of the best effects.

### 5.1. Setting the variables of the MBO technique

In the MBO technique, several variables are checked, including: These variables are, population, neighborhood, and subscription, shake.

Table 6 presents the population parameters with values of 71, 61, 51 and 41. Input,

neighbor, and share variables are set to 30, 13, and 1, respectively. As you can see in Table 6, the best mean value in the population = 71 is obtained. It can also be seen in Table 7 that the best mean value is obtained in input = 30.

Table 7 evaluates the input parameters with values of 60, 50, 40 and 30. The neighborhood, subscription and population parameters are fixed at 1, 13 and 71, respectively.

In Table 8, the neighborhood variables are evaluated with values of 10, 8, 6 and 4. Sharing, shaking, and population parameters are fixed at 1, 30, and 71, respectively.

Based on the performance of MBO technique, the minimum value of the neighboring variable can be attributed to the number 3. According to equation (3), if  $m = 3$ , the sharing variable ( $q$ ) can only be 1. So, the sharing variable does not need to be checked and we set its size to 1.

According to the tables obtained from the analysis of variables and data, the best data of the MBO technique are given in Table No. 9.

## 5.2. TS variable setting

Three different variables are investigated in the TS technique. These numbers and parameters include the length of the taboo, the long-term penalty, and the long-term penalty.

**Table 6**

Variable setting of population.

Fixed measures		Population Value	Fitness Average
Parameter	Value		
Input	30	41	10347907,36
Neighbor	13	51	10381693,10
Share	1	61	10340936,30
		71	10290422,95

**Table 7**

Variable setting of flap.

Fixed measure		Flap Value	Fitness Average
Parameter	Value		
Population	71	30	10237226,00
Neighbor	13	40	10276514,80
Share	1	50	10302385,16
		60	10256843,93

**Table 8**

Neighborhood variable setting

Fixed measures		Neighbor Value	Fitness Average
Measure	Value		
Population	71	4	10224283,63
Input	30	6	10275433,16
Share	1	8	10305913,26
		10	10269533,46

**Table 9**

The best data for MBO.

Measurers	Value
Population	71
Input	30
Neighbor	13
Share	1

**Table 10**

Variabl setting of tabu length.

Fixed measure		Tabu length Value	Fitness Average
Measure	Value		
Penalizing long term	71	20	11324842,17
		30	11162677,17
long term length	110	40	11000656,80
		50	11293773,77

In Table 10, the taboo length data with values of 50, 50, 30 and 20 are tested and evaluated. Long-term fine and long-term data are fixed at 110 and 10 by definition. As you can see from the data in Table 10, the best value of the process is obtained at tabu length = 40.

Based on table number 11, we see that; the long-term penalty data is evaluated at different values of 30, 25, 20 and 15. The long-term and taboo parameters are fixed at 110 and 40, respectively. According to the information in Table 11, the best process value for a long-term penalty is = 30.

Table 12 evaluates and evaluates different long-term length numbers and data with 120, 110, 100 and 90 test values. The data of taboo length and long-term penalty are fixed at constant values of 30 and 40, respectively. As shown in Table 12, the best mean value is obtained in the long run = 90.

The best data related to the TS technique according to the observations and results obtained are given in Table No. 13.

**Table 11**

Measure setting of penalizing long term.

Fixed Measure		Penalizing long term	Fitness
Measure	Value	Value	Average
tabu length	40	15	11163603,20
		20	11391634,70
long term length	110	25	11268883,43
		30	11133662,73

**Table 12**

Measure setting of long term length.

Fixed measure s		long term length	Fitness
Measure	Value	Value	Average
tabu length	40	90	10979242,50
		100	10991153,00
long term length	30	110	11203683,33
		120	11203941,57

**Table 13**

Best measures for the TS.

Measures	Value
tabu length	40
penalizing long term	30
long term length	90

## 6. Practical results

Considering that the problem of allocation of terminal facilities is a special problem, then the effects of this problem cannot be compared with the effects of standard problems in similar literature. However, this issue is not only solved by MBO but also by TS and SA methods to evaluate performance efficiency. Tests are performed with an Intel (R) Core (TM) i5-3330 @ 3.00 GHz processor, 1 GB of RAM and Ubuntu 14.04 (64-bit) Linux operating system. All techniques and algorithms are coded with QT Creator 3.1.1 gcc compiler and C++ language. For each technique, the algorithms are run 30 times separately to obtain the overall results. Each

test runs for 120 seconds. Depending on the data settings, 120 seconds seems sufficient for both techniques. Parking maps obtained from tests, TS, SA and MBO and it is given in table number 14.

According to the observations and parameters of Table No. 14, the best effect obtained from the SA algorithm is equal to 10142858, which is about 1.2% worse than the MBO and SA techniques. For system design and current terminal design, the best results of MBO and SA techniques and their algorithms are given in Table 15 .

Based on the data obtained from Table 15, parking lots 0-3 are located on the 0th floor of Z-Block. Also with the number of consultations, people seem to pour into the lounge from every terminal station.. Considering the number of parking spaces of the stations and the distance to the main hall, it seems logical that the halls should be placed on the 0th floor of the block. Stations 6 and 7 are located on the 0th stage without blocks, parking lots 13 and 14 are located on the 0th bottom of Block A, and parking lots 25 and 26 are located on the 1st stage of Block X. It can also be seen from Table No. 15, the layout of the proposed station and the existing parking layout are generally distinct, except for parking lots 10 and 17. Both are located on the 1st stage of Y-Block. Also, the appropriateness and allocation values of the proposed plan are given in Table 15 . According to the proposal, the allocation of the terminal layout has been improved.

The convergence rate of TS and MBO techniques is closer to each other and faster than SA technique. But the efficiency obtained from the SA technique is better than the TS and MBO methods according to the worst results and the average.

**Table 14**

Results from MBO, SA and TS.

	Method		
	MBO	TS	SA
Min.	10142858,0	10254993,0	10142858,0
Avg.	10236622,3	11054517,2	10158826,4
Worst	0652954,0	11872704,0	10331548,0

**Table 15**

Existing and proposed station design			
Block	Floor	Proposed station Layout	Existing station Layout
X-Block	0	13, 15, 29, 9	29, 23, 31, 22
	1	26, 26, 5, 13	8, 16, 25, 31
Y-Block	0	4, 6, 8, 12	15, 18, 18, 28
	1	10, 28, 17, 24	10, 11, 19, 21
Z-Block	-1	19, 17, 31, 23	12, 21, 4, 5
	0	2, 0, 4, 1	6, 7, 8, 27
	1	18, 9, 16, 25	13, 15, 25, 26
	2	21, 28, 31, 21	0, 1, 4, 3
Fitness Value		10, 141, 847	17, 423, 012

According to the results obtained from all three meta-heuristic techniques that are analyzed and tested practically. To check whether there is a significant difference between the techniques used, we use the Wilcoxon criterion. There is no important difference between the two measured techniques under the assumption of  $H_0$ . There is a fundamental difference between these two techniques measured by the alternative hypothesis  $H_1$ . The acceptable value of  $H_0$  hypothesis is set at 95%. In this way, if the distance of each technique is less than 5% ( $\alpha = 0.05$ ), there is not much distance between two techniques. The results of the techniques are recorded in Table No. 16 at a significance level of 5% .

**Table 16**

Wilcoxon test results.		
p-values		
TS&MBO	SA&MBO	TS &SA
0.001	0,148	0.001

According to the results obtained in table number 16, there is no important difference between SA and MBO techniques ( $0.148 > 0.05$ ). According to the results obtained from the tests, it can be seen that there is an acceptable difference between TS and MBO techniques with SA and TS techniques ( $< 0.05$ ).

## 7. Conclusion and discussion

In the end, we will examine the optimization issues of urban life that we can face in many

areas of our real life. In these cases, meta-heuristic techniques are used to solve practical optimization subjects. Placing different bodies in appropriate areas in the issue of facility allocation increases operational efficiency by improving system costs, implementation and process time. The location of the stations inside the terminal plays an important role in determining the commuting time of customers and terminal employee. The transfer time of a customer is to visit a station without an intermediary for service or to receive service from a station to a secondary station. The lengthening of these transfer times leads to a longer stay in the terminal and a decrease in the terminal's performance. Therefore, the precise designation of these walking stations ensures the optimal use of terminal electricity and minimizes unnecessary separation between customers, employees and tourists, and increases the productivity of the terminal. In this study, we used TS, SA and MBO meta-algorithms to allocate station space for a real-scale terminal. Based on the results obtained from the experiences, the success of TS, SA and MBO techniques in facility allocation problems is tested on a transportation problem and it is observed that the total cost of SA and MBO techniques is very better than TS technique. When we compare the fit value generated from the applied results with the fit value available in a system, we see that our method performs well. As a result, technology can be used to allocate terminal stations. For future work, some constraints and assumptions can be added to the problem. As if a customer has several turns at several stations or some passengers need support to move. The re-completion operation of each section can also be considered .

## References

- [1] J. Tompkins, J. White, W. Bozer, J. Tanchoco, Facilities Planning, John Wiley & Sons, New York, USA, 2003.
- [2] Sarma, Sisira, Demand for outpatient healthcare, Appl. Health Econ. Health Policy 7 (4) (2009) 265–277.
- [3] R.S. Amaral, A new lower bound for the single row facility layout problem, Discrete Appl. Math. 157 (2009) 183–190
- [4] M. Mohammadi, K. Forghani, A novel approach for considering layout problem in cellular manufacturing systems with alternative processing routings and

- subcontracting approach, *Appl. Math. Model.* 38 (14) (2014) 3624–3640.
- [5] A. Barbosa-Povoa, R. Mateus, A. Novais, Optimal two-dimensional layout of industrial facilities, *Int. J. Prod. Res.* 39 (12) (2001) 2567–2593.
- [6] S.P. Singh, R.R. Sharma, A review of different approaches to the facility layout problems, *Int. J. Adv. Manuf. Technol.* 30 (5–6) (2006) 425–433.
- [7] B. Montreuil, A modeling framework for integrating layout design and flow network design, in: *Proceedings of the Material Handling Research Colloquium*, 1990, pp. 43–58.
- [8] T. Lacksonen, Pre-processing for static and dynamic facility layout problems, *Int. J. Prod. Res.* 35 (1997) 1095–1106.
- [9] F. Azadivar, J. Wang, Facility layout optimization using simulation and genetic algorithms, *Int. J. Prod. Res.* 38 (17) (2000) 4369–4383.
- [10] Y.H. Lee, M.H. Lee, A shape-based block layout approach to facility layout problems using hybrid genetic algorithm, *Comput. Ind. Eng.* 42 (2) (2002) 237–248.
- [11] E. Shayan, A. Chittilappilly, Genetic algorithm for facilities layout problems based on slicing tree structure, *Int. J. Prod. Res.* 42 (2) (2002) 237–248.
- [12] R. Sahin, O. Turkbey, A new hybrid heuristic algorithm for the multi objective facility layout problem, *J. Faculty Eng. Archit. Gazi Univ.* 25 (1) (2010) 119–130.
- [13] M.Y. Cheng, L.C. Lien, A hybrid AI-based particle bee algorithm for facility layout optimization, *Eng. Comput.* 28 (1) (2012) 57–69.
- [14] Q. Luo, Q. Su, J. Le, L. Lu, in: *10th International Conference on Service Systems and Service Management*, IEEE, 2013, pp. 224–227.
- [15] Y.Y. Ileri, The Importance of Layout Organization in Hospital Management Efficiency: A Model in S.U. Medical Faculty Hospital, Selcuk University, 2013.
- [16] D.T.T. Huyen, N.T. Binh, T.M. Tuan, T.Q. Trung, N.G. Nhu, N. Dey, L.H. Son, Analyzing trends in hospital-cost payments of patients using ARIMA and GIS: case study at the hanoi medical university hospital, Vietnam, *J. Med. Imaging Health Inf.* 7 (2) (2017) 421–429.
- [17] M.E. Aydin, T.C. Fogarty, A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems, *J. Heuristics* 10 (3) (2004) 269–292.
- [18] A. Kaveh, P. Sharafi, Charged system search algorithm for minimax and minisum facility layout problems, *Asian J. Civ. Eng.* 6 (12) (2011) 703–718.
- [19] A. Kaveh, M.A.A. Shakouri, M.S. Zolfaghari, An adapted harmony search based algorithm for facility layout optimization, *Int. J. Civ. Eng.* 1 (10) (2012) 37–42.
- [20] K.Y. Chan, M.E. Aydin, T.C. Fogarty, Main effect fine-tuning of the mutation operator and the neighbourhood function for uncapacitated facility location problems, *Soft. Comput.* 10 (11) (2006) 1075–1090.
- [21] J. Yang, M.E. Aydin, J. Zhang, C. Maple, UMTS base station location planning: a mathematical model and heuristic optimisation algorithms, *IET Commun.* 1 (5) (2007) 1007–1014.
- [22] E. Duman, M. Uysal, A.F. Alkaya, Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem, *Inf. Sci.* 217 (2012) 65–77.
- [23] V. Tongur, E. Ülker, Migrating birds optimization for flow shop sequencing problem, *J. Comput. Commun.* 2 (04) (2014) 142.
- [24] A. Sioud, C. Gagné, Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times, *Eur. J. Oper. Res.* 264 (1) (2018) 66–73.
- [25] B. Zhang, Q.K. Pan, L. Gao, X.L. Zhang, H.Y. Sang, J.Q. Li, An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming, *Appl. Soft Comput.* 52 (2017) 14–27.
- [26] T. Meng, Q.K. Pan, J.Q. Li, H.Y. Sang, An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem, *Swarm Evol. Comput.* 38 (2018) 64–78.
- [27] Q.K. Pan, Y. Dong, An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation, *Inf. Sci.* 277 (2014) 643–655.
- [28] K.Z. Gao, P.N. Suganthan, T.J. Chua, in: *An Enhanced Migrating Birds Optimization Algorithm for No-Wait Flow Shop Scheduling Problem*, IEEE, 2013, pp. 9–13.
- [29] E. Duman, I. Elikucuk, Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization, 62–71, *International Work-Conference on Artificial Neural Networks*, Springer, Berlin, Heidelberg, 2013.
- [30] E. Ulker, V. Tongur, Migrating birds optimization (MBO) algorithm to solve knapsack problem, *Proc. Comput. Sci.* 111 (2017) 71–76.
- [31] V. Tongur, E. Ülker, The analysis of migrating birds optimization algorithm with neighborhood operator on traveling salesman problem, in: *Intelligent and Evolutionary Systems*, Springer, Cham, 2016, pp. 227–237.
- [32] V. Tongur, E. Ülker, PSO-based improved multi-flocks migrating birds optimization (IMFMBO) algorithm for solution of discrete problems, *Soft. Comput.* 23 (14) (2019) 5469–5484.
- [33] M. Hacıbeyoglu, K. Alaykiran, A.M. Acilar, V. Tongur, E. Ulker, A Comparative Analysis of metaheuristic approaches for multidimensional two-way number partitioning problem, *Arabian J. Sci. Eng.* 43 (12) (2018) 7499–7520.
- [34] Z. Li, M.N. Janardhanan, A.S. Ashour, N. Dey, Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem, *Neural Comput. Appl.* (2019) 1–17.
- [35] F. Glover, Tabu search part I, *ORSA J. Comput.* 1 (3) (1989) 190–206.

- [36] F.A.T. Montané, D.G. Roberto, A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, *Comput. Oper. Res.* 33 (3) (2006) 595–619.
  - [37] J. Grabowski, W. Mieczyslaw, A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion, *Comput. Oper. Res.* 31 (11) (2004) 1891–1909.
  - [38] C.N. Fiechter, A parallel tabu search algorithm for large traveling salesman problems, *Discrete Appl. Math.* 51 (3) (1994) 243–267.
  - [39] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, *Am. Assoc. Adv. Sci.* 220 (4598) (1983) 671–680.
  - [40] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, *J. Chem. Phys.* 21 (6) (1953) 1087–1092.
  - [41] H. Bagherlou, A. Ghaffari, A routing protocol for vehicular ad hoc networks using simulated annealing algorithm and neural networks, *J. Supercomput.* 74 (6) (2018) 2528–2552.
  - [42] L. Wei, Z. Zhang, D. Zhang, S.C. Leung, A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints, *Eur. J. Oper. Res.* 265 (3) (2018) 843–859.
  - [43] K. Karagul, Y. Sahin, E. Aydemir, A. Oral, A simulated annealing algorithm based solution method for a green vehicle routing problem with fuel consumption, in: *Lean and Green Supply Chain Management*, Springer, Cham, 2019, pp. 161–187.
  - [44] M.M. Mafarja, S. Mirjalili, Hybrid whale optimization algorithm with simulated annealing for feature selection, *Neurocomputing* 260 (2017) 302–312.
  - [45] S.H. Zhan, J. Lin, Z.J. Zhang, Y.W. Zhong, List-based simulated annealing algorithm for traveling salesman problem, *Comput. Intell. Neurosci.* 2016 (2016) 8.
  - [46] S. Kulturel-Konak, A. Konak, A large-scale hybrid simulated annealing algorithm for cyclic facility layout problems, *Eng. Optim.* 47 (7) (2015) 963–978.
  - [47] N. Shivasankaran, P.S. Kumar, K.V. Raja, Hybrid sorting immune simulated annealing algorithm for flexible job shop scheduling, *Int. J. Comput. Intell. Syst.* 8 (3) (2015) 455–466.
  - [48] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, *J. Heuristics* 15 (6) (2009) 617.
- in binary computer arithmetic, "IEEE Computer Group Repository, Paper R-67-85.



Paper Type (Research paper)

**Fuzzy Logic-Based Vector Control Method for Induction Motors**

Gholam Reza Aboutalebi\*

*Energy and Environment Research Center, Shahrekord Branch, Islamic Azad University, Shahrekord, Iran***Article Info****Article History:***Received: 2025/03/15**Revised: 2025/06/08**Accepted: 2025/06/12**DOI:***Keywords:***Fuzzy logic, Speed control, Torque control, Induction motors, Motor performance optimization.**\*Corresponding Author's Email Address:  
gh.r.aboutalebi@gmail.com***Abstract**

Vector control methods in induction motors based on proportional-integral (PI) and proportional-integral-derivative (PID) controllers with fixed gains are not effective against changes in system parameters, load changes, temperature changes, magnetic saturation, and other disturbances due to their strong dependence on machine parameters.

In vector control systems, motor flux and torque control are performed by determining the currents and spatial angles of the vectors, which are not very accurate due to instantaneous oscillations in the load and changes in rotor resistance. In many industrial applications, the stable and precise performance of these controllers is challenged. To deal with these problems, there is a need for an adaptive control system that can dynamically adjust the controller gains. The use of fuzzy logic controllers (FLC) due to their high flexibility, adaptability to different operating conditions, and improved dynamic response, without the need for a precise mathematical model of the system, can adjust of control strategies based on linguistic rules and fuzzy sets. In this paper, an induction motor indirect vector control method is replaced with a fuzzy logic controller. The results of simulation and evaluation of the method in different conditions show that the use of fuzzy control leads to improved stability, reduced speed, and torque oscillations, reduced system response delay, and increased control accuracy and can be a suitable alternative to classical controllers in industrial applications in systems requiring precise and stable performance.

**1. Introduction**

AC motor drives require high-efficiency performance due to their numerous industrial applications. In these drives, the motor speed must follow the desired reference speed trajectory with less influence from load changes, parameter changes, and motor model estimation errors. For this purpose, the vector control method was proposed. In this method, the design of an appropriate controller plays a decisive role in the drive performance.

Unfortunately, there are problems such as high sensitivity to machine parameters such as rotor time constant, and the need for accurate flux measurement and estimation in the vector control method.

The fixed-gain PI and PID controllers, which are commonly used in speed control drives, are very sensitive to changes in parameters and load changes, so the parameters of these controllers must be continuously adapted to the prevailing environmental and load conditions. This problem can be solved to some extent by various techniques such as Model Reference Adaptive Control (MRAC)<sup>1</sup>[1], Sliding Mode Control (SMC)<sup>2</sup>[2], Variable Structure Control (VSC)<sup>3</sup>[3], Self-Tuning PI Controllers<sup>4</sup>[4] and some other methods.

Controller design in all the above methods requires a more accurate mathematical model of the system, but determining the exact mathematical model of the system

<sup>1</sup> Model Reference Adaptive Control<sup>2</sup> Sliding-Mode Control<sup>3</sup> Variable Structure Control<sup>4</sup> Self - tuning PI controller

is often difficult due to reasons such as uncertain load changes and uncertain changes in parameters due to conditions such as temperature changes and system disturbances [5, 6].

To overcome these problems, fuzzy logic controllers (FLC)<sup>5</sup> can be used [7-10].

In a general comparison between classical PI and PID controllers and adapted fuzzy controllers, the following advantages are observed [11, 12]:

1. These controllers do not require an accurate mathematical model of the system.
2. They are easy to implement in systems with nonlinear and complex behavior.
3. The structure of this type of controllers is based on the linguistic rules common among humans and can be implemented through "IF-Then" statements, which itself expresses the proximity of this logic to life in human societies.

In order to achieve better performance, the indirect vector control method is simulated with the help of a fuzzy controller, and its results are presented.

## 2. Indirect Vector Control of Induction Motors

The stages of vector control at induction motor by indirect method and by precise tracking of the rotor field are given below [13, 14]:

First step) Sampling of a stator and Calculating the real value of Longer and Transverse Components of Stator current in rotor flow Coordinates:

$$\begin{bmatrix} i_{ds}^s \\ i_{qs}^s \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_{SA} \\ i_{SB} \\ i_{SC} \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} i_{ds}^e \\ i_{qs}^e \end{bmatrix} = \begin{bmatrix} \cos \theta_e & \sin \theta_e \\ -\sin \theta_e & \cos \theta_e \end{bmatrix} \begin{bmatrix} i_{ds}^s \\ i_{qs}^s \end{bmatrix} \quad (2)$$

Second step) Calculate the amount of the rotor flux linkage by estimated  $\left| \bar{\psi}_r \right|_{est}$ , angular slip angle frequency  $\omega_{sl}$ , and rotor angle position  $\theta_e$ . For  $\omega_{sl}$ :

$$\omega_{sl} = \frac{L_m}{\left| \bar{\psi}_r \right|_{est}} \cdot \left( \frac{R_r}{L_r} \right) i_{qs}^* \quad (3)$$

$$\theta_e = \int (\omega_m + \omega_{sl}) dt \quad (4)$$

And for the rotor flux linkage, we can write:

$$\left| \bar{\psi}_r \right|_{est} = \frac{L_m i_{ds}^*}{1 + T_R S} \quad (5)$$

Third Step) Determine the Reference Current  $i_{ds}^*$ :  
The speed control in this study is below the base speed  $\omega_b$ . Therefore,  $i_{ds}^*$  is calculated using the following relationship:

$$i_{ds}^* = \frac{\left| \bar{\psi}_r \right|^*}{L_m} \quad (6)$$

Where  $\left| \bar{\psi}_r \right|^*$  is the reference value of the rotor flux space phasor and its nominal value can be obtained through the steady-state model of the induction machine in the constant torque region of the induction machine's speed-torque curve.

Fourth Step) Determine the Reference Current :  $i_{qs}^*$   
The torque-producing component in an induction motor is the reference current,  $i_{qs}^*$ , and can be calculated from  $T_e^*$  as follows:

$$i_{qs}^* = \frac{2}{3P} \cdot \frac{L_r}{L_m} \cdot \frac{T_e^*}{\left| \bar{\psi}_r \right|_{est}} \quad (7)$$

In Equation (7),  $T_e^*$ , represents the reference electromagnetic torque,  $P$ , denotes the number of pole pairs in the machine, and  $i_{qs}^*$  corresponds to the reference value of the transverse (quadrature-axis) component of the stator current."

To achieve  $T_e^*$ , the motor speed is initially sampled. Then, the error between the desired reference speed and the actual motor speed is processed through a proportional-integral (PI) speed controller, which generates the reference torque  $T_e^*$ .

Fifth Step) Converting reference currents  $i_{ds}^*$  and  $i_{qs}^*$  into three-phase currents  $i_a^*$ ,  $i_b^*$ , and  $i_c^*$  through equations (8) and (9):

$$\begin{bmatrix} i_{ds}^{s*} \\ i_{qs}^{s*} \end{bmatrix} = \begin{bmatrix} \cos \theta_e & -\sin \theta_e \\ \sin \theta_e & \cos \theta_e \end{bmatrix} \begin{bmatrix} i_{ds}^{e*} \\ i_{qs}^{e*} \end{bmatrix} \quad (8)$$

<sup>5</sup> Fuzzy Logic Controller

$$\begin{bmatrix} i_a^* \\ i_b^* \\ i_c^* \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_{ds}^* \\ i_{qs}^* \end{bmatrix} \quad (9)$$

Sixth Step) Applying three-phase reference currents to a current-controlled PWM inverter:

At this stage, the current error resulting from the three-phase reference currents and the sampled currents is applied to a hysteresis controller with a

specific hysteresis band to generate the necessary pulses for the inverter.

The general block diagram of indirect control of an induction motor using the FOC method based on the above six steps with Current-controlled VSI voltage source inverter will be as shown in Figure (1).

In this block diagram, the speed controller is of the PI type and its role is to keep the actual speed of the motor equal to the reference speed in both steady-state and transient states with good dynamic response.

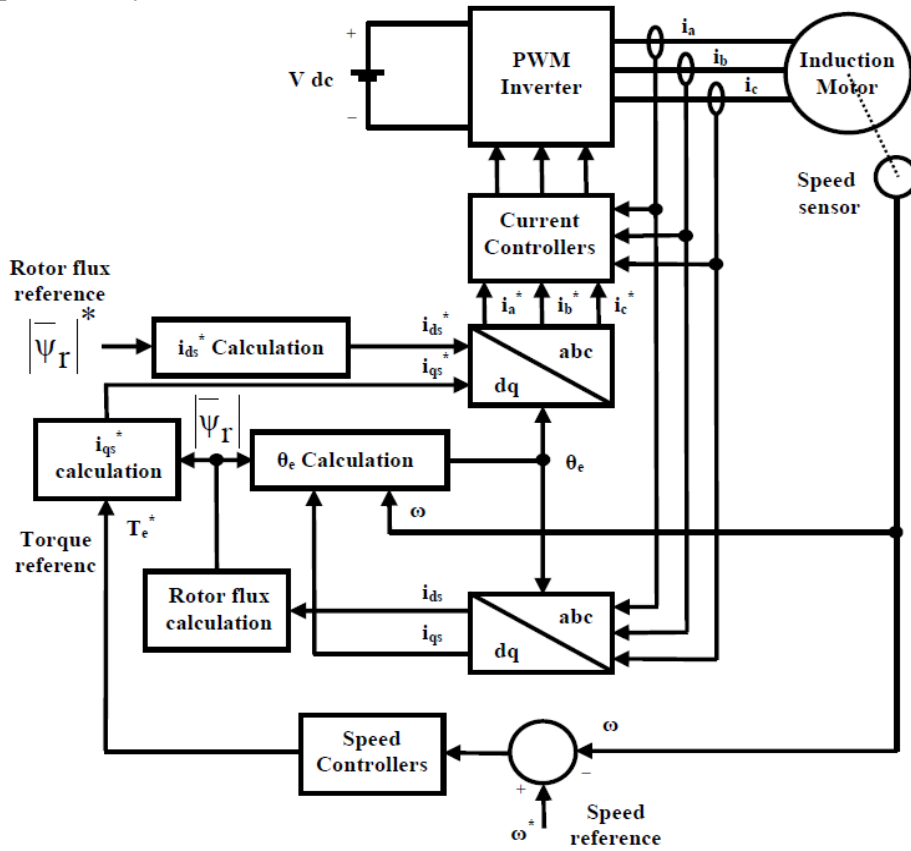


Figure 1. General block diagram of indirect vector control of an induction motor using the FOC method

The curves related to the start-up of a three-phase squirrel cage induction motor sample without load and at a speed lower than the rated speed are shown in Figure (2). As can be seen, the motor speed has reached the reference speed after approximately 1.9s. The results of the study of the dynamic behavior with respect to changes in load torque and reference speed are also shown in Figure (3). In this figure, the reference speed of the running motor has increased to 150 radians/second, which is the rated speed of the motor. Also, after a certain period of

time, a load torque of 100 N.m has been applied to the motor.

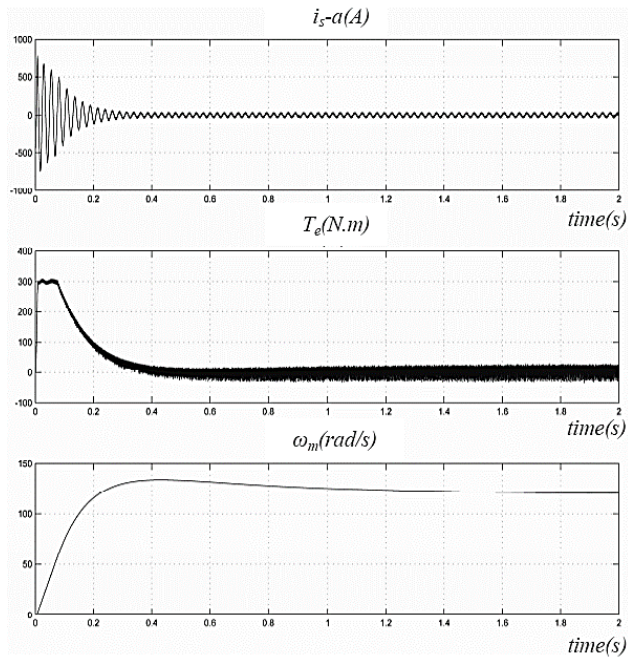


Figure 2. The curves related to the start-up of a three-phase squirrel cage induction motor sample without load and at a speed lower than the rated speed

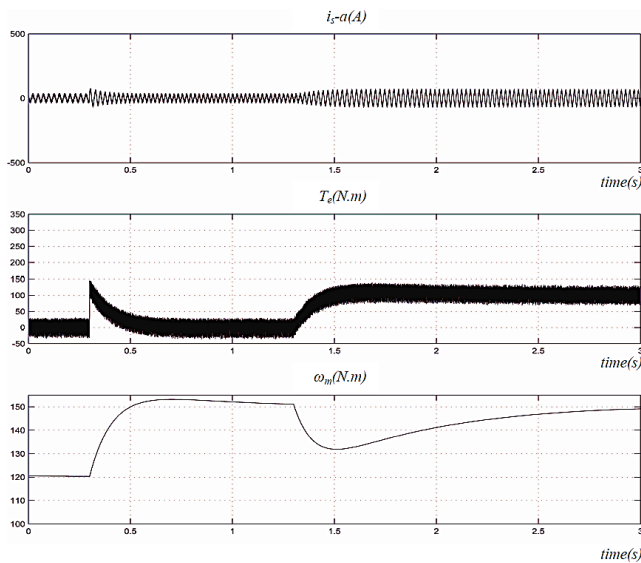


Figure 3. dynamic behavior with respect to changes in load torque and reference speed

### 3. Fuzzy vector control

Fuzzy logic techniques are of significant use in solving many problems in various sciences [15-17]. These techniques play a particularly important role in the control of engineering processes. Fuzzy logic controllers (FLC) allow the setting of control strategies based on linguistic rules and fuzzy sets, due to their high flexibility, adaptability to different operating conditions, and improved dynamic response, without the need for an accurate mathematical model of the system.

Due to the high sensitivity of the vector control method to machine parameters such as the rotor time constant, the need for accurate flux measurement and estimation, etc., classical PI controllers are not very suitable for this method. For this reason, the controller in the vector control method is replaced by a fuzzy PI controller.

PI controllers are used as one of the most important controllers due to their simple structure and robust performance. The transfer function of these controllers is as follows:

$$G = k_p + \frac{k_i}{s} \quad (10)$$

The success of a PI controller depends on the appropriate choice of its gains, A and B. In practice, determining the PI gains that will provide optimal efficiency is not a simple task and must be derived with the help of expert experience and based on a number of general rules.

In a speed control system, the goal is to achieve a fast rise time with the least overshoot.

Therefore, the set of rules of the fuzzy control system is obtained empirically and based on the step response. Figure (4) shows a typical response of a process to a step input.

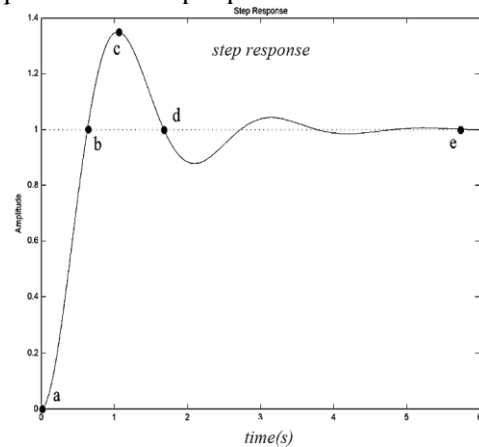


Figure 4 A typical response of a process to a step unit input

Around point “a”, a large control signal is needed, so we can say: If the error between the reference speed and the actual speed of the motor is large, then  $k_p$  should be large and  $k_i$ , small.

At points “b” and “d”, the speed changes with respect to time are large and the error between the reference speed and the actual speed is small, so we can write: If the speed changes with respect to time (acceleration) are large and the speed error is small, Then  $k_p$  and  $k_i$  are both small because large  $k_p$  causes large overshoot and large  $k_i$  causes the system to oscillate.

At point  $e$ , the error between the reference speed and the actual speed is zero and the speed changes with respect to time are also almost zero, so we have: If the speed error is zero and the acceleration is also zero, then  $k_p$  should be small and  $k_i$ , large

So that the steady-state error in the system is reduced. A similar behavior to point “a” can also be proposed for “c” point.

Based on the above expressions, the desired fuzzy controller can be prepared for replacement in the vector control method. For this purpose, one fuzzy controller is considered for determining the value and another fuzzy controller is considered for determining.

The fuzzy inference system used in this paper is of the Mamdani type. This system has features such as its efficiency in ambiguous environments, the use of human knowledge, and the ability to find the optimal solution to the problem from a large number of available solutions.

The membership functions used in these controllers are trapezoidal and triangular, which are defined as follows:

$$f(x; a, b, c, d) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c \leq x \leq d \\ 0 & x \geq d \end{cases} \quad (11)$$

In the above expressions, if  $b = c$ , the triangular membership function will be obtained.

The methods used for combining and summing the rules are also in accordance with the following relation expressions:

min	:	And Method
max	:	Or Method
min	:	Implication
max	:	Aggregation
center of gravity	:	Defuzzification

The defuzzification method used is the center of gravity method, which is defined as follows:

$$output\ i = \frac{\sum_{k=1}^N i \mu_{c(k)}(i)}{\sum_{k=1}^N \mu_{c(k)}(i)} \quad (12)$$

Where  $N$  is the number of fuzzy rules used and  $\mu_{c(k)}(i)$  is the membership degree of the output for the  $k$ th rule.

Figure (5) shows the membership functions and fuzzy rules related to the  $k_p$  fuzzy controller. In this figure, the first input of the fuzzy controller is the error of the reference speed and the measured speed, which is represented by “ $e$ ”, and the second input is the acceleration or change in speed with respect to time, which is represented by “ $a$ ”.

$$e = \omega_{ref} - \omega \quad (13)$$

$$a = \frac{d\omega}{dt} \quad (14)$$

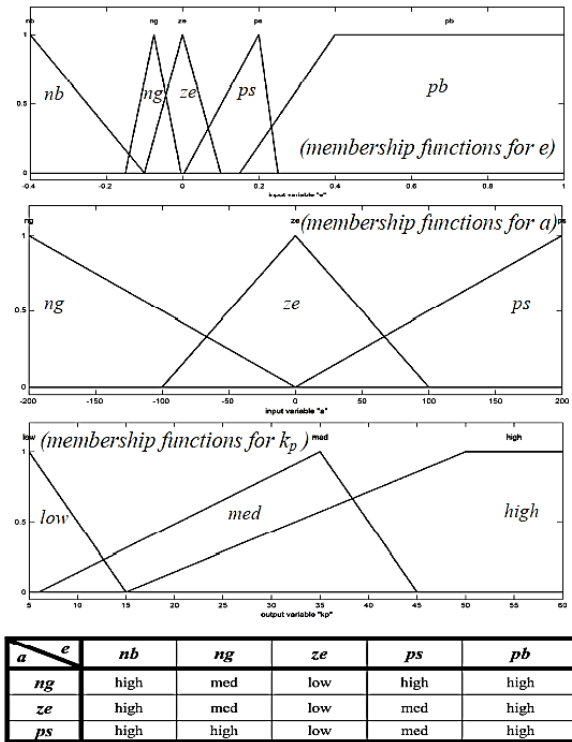


Figure 5. membership functions and fuzzy rules related to the  $k_p$  fuzzy controller

Figure (6) shows the membership functions and fuzzy rules related to the  $k_i$  fuzzy controller. The inputs of this controller are the velocity error “ $e$ ” and acceleration “ $a$ ” and its output is the desired numerical value for  $k_i$  according to the fuzzy rules used.



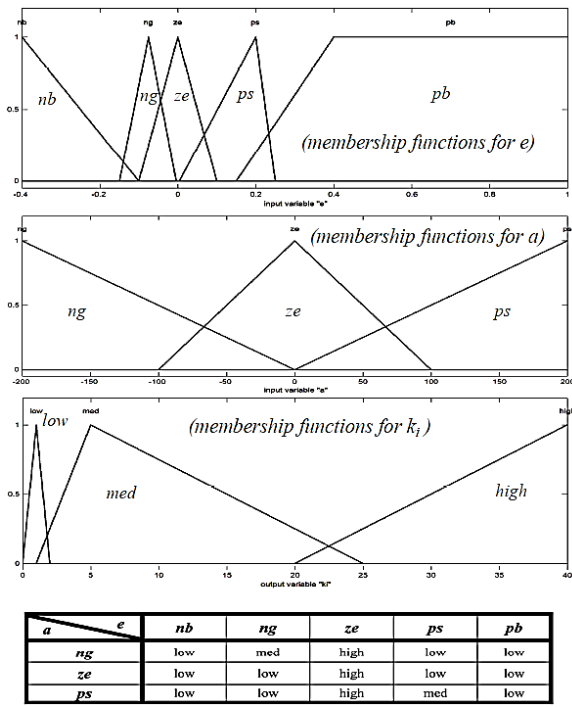


Figure 6. membership functions and fuzzy rules related to the  $k_i$  fuzzy controller

By replacing the block corresponding to the classical controller in the conventional vector control method with the blocks of fuzzy controllers and the necessary sections, the fuzzy vector control method was simulated on a motor with the same specifications and under similar transient conditions. The simulation results are shown in Figures (7), (8), and (9).

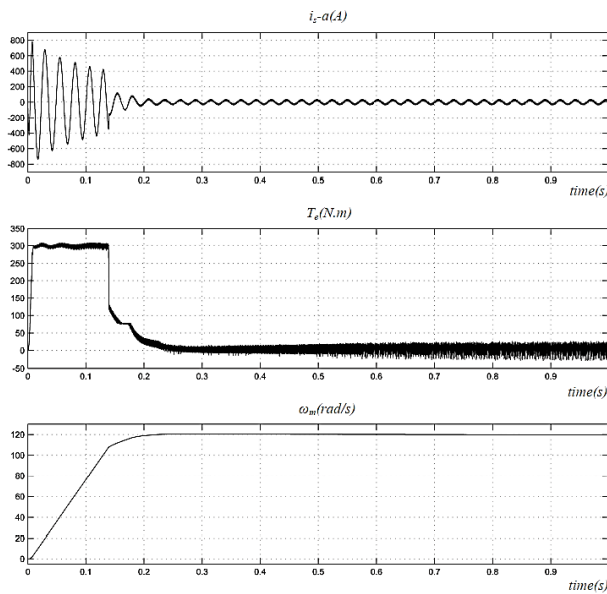


Figure 7. The curves related to the start-up of a induction motor sample without load and at a speed lower than the rated speed with fuzzy logic controller

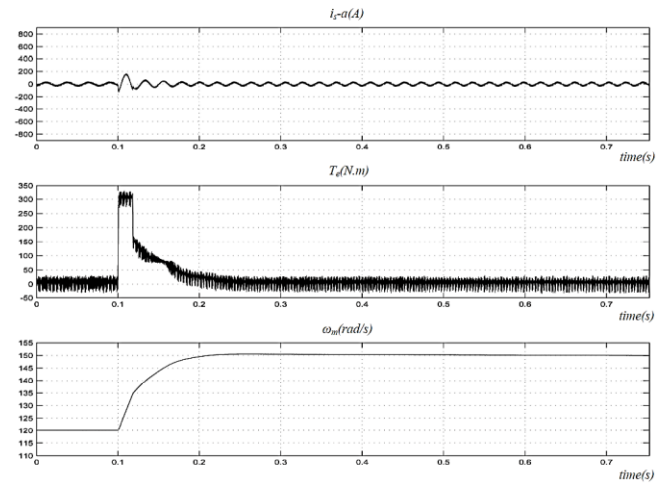


Figure 8. dynamic behavior with respect to the reference speed increase with fuzzy logic controller

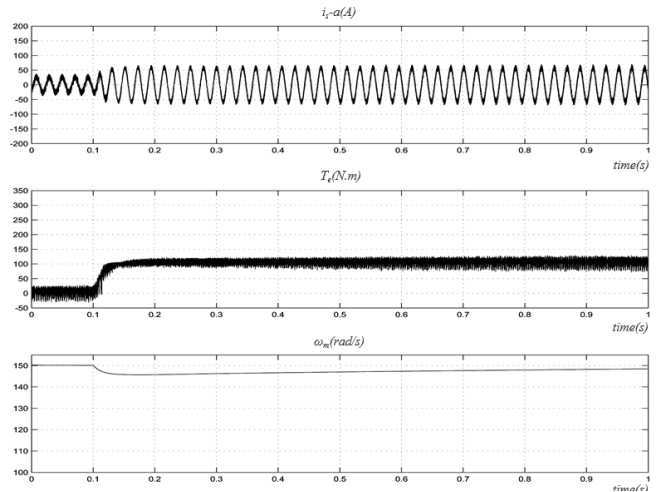


Figure 9. Dynamic behavior with increasing motor load using a fuzzy logic controller

The curves related to the changes in the  $k_i$  and  $k_p$  gains and the ratio of the speed change curve are presented in Figure (10). By observing this figure and the previously presented fuzzy rules, the result of applying fuzzy rules and using fuzzy controllers can be seen.

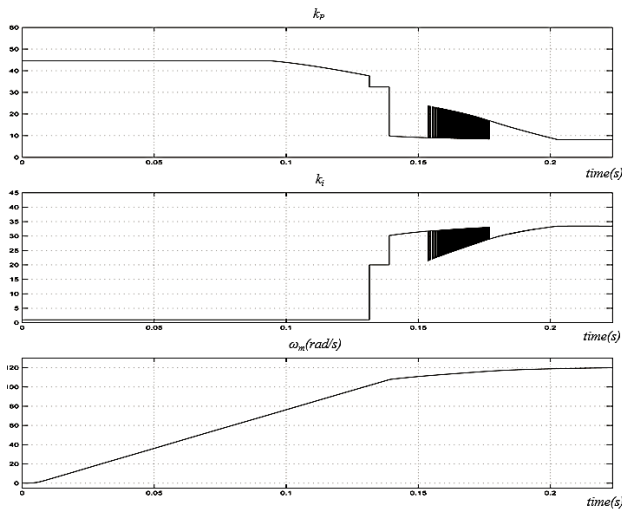


Figure 10.

Figure (11) presents the simultaneous response of the induction motor starting curves using the classical vector control method and the fuzzy vector control method.

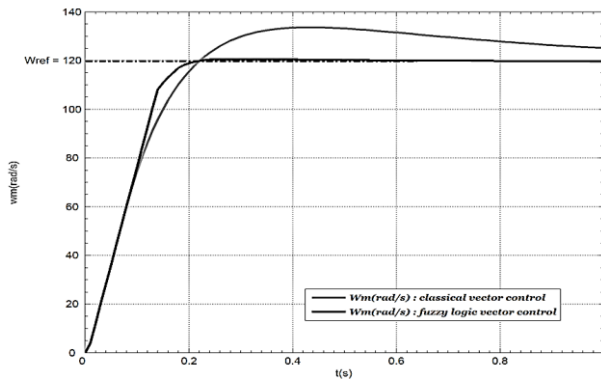


Figure 11.

As can be seen in Figure (11), in the fuzzy method, the speed of the induction motor will reach the desired reference speed in less time. Also, the steady state error in this case is almost zero.

The rate of speed overshoot from the desired reference speed is very small and can be reduced further by better and more appropriate selection of membership functions and their parameters and the use of fuzzy rules.

However, the fuzzy vector control method is one of the reliable application method of speed control of induction motors, which in addition to its simplicity increases the speed and accuracy of the system response under variable environmental conditions and motor parameters. Of course, the practical implementation of this method is possible using fast digital signal processors (DSP<sup>6</sup>).

#### 4. Specifications of the squirrel cage induction motor used in the simulations

The specifications of the induction motor used in the simulations are as follows:

Table 1. Specifications of the induction motor

Parameter	Nominal value
$P_n$	50 HP (37kW)
$V_n$	400 V
$f_n$	50 Hz
$N_n$	1480 rpm
$R_s$	0.08233 $\Omega$
$L_{ds}$	0.724 mH
$R'_{rs}$	0.0503 $\Omega$
$L_{lr}$	0.724 mH
$L_m$	27.11 mH
$J$	0.37 kg.m <sup>2</sup>
$f$	0.02791 N.m.s
$P$	2 (Pair of poles)

#### 5. Conclusion

Using the fuzzy vector control method, on the one hand, improves the speed and accuracy of the system's response to sudden changes in the load torque or the applied reference speed, and on the other hand, by using a fuzzy controller instead of a classic controller, the high sensitivity of the control system to changes in environmental conditions and changes in engine parameters is reduced. So that changing the proportional and integral gains of the controller during system operation tries to create a desired response in following the desired reference speed and responding to changes in the load torque in loads that have uncertain behavior.

The fuzzy control method in determining the controller gains compared to fixed-gain controllers includes other advantages such as simplicity, no need for an accurate mathematical model of the system, and faster response to unwanted changes in the load characteristics and engine model.

#### References

- [1] R. Kumar, S. Das, P. Syam, and A. K. Chattopadhyay, "Review on model reference adaptive system for sensorless vector control of induction motor drives," IET Electr Power Appl, vol. 9, no. 7, pp. 496–511, Aug. 2015, doi: 10.1049/iet-epa.2014.0220.
- [2] S. Di Gennaro, J. Rivera Dominguez, and M. A. Meza, "Sensorless High Order Sliding Mode Control of Induction Motors With Core Loss," IEEE

<sup>6</sup> Digital signal Processing



- Transactions on Industrial Electronics, vol. 61, no. 6, pp. 2678–2689, Jun. 2014, doi: 10.1109/TIE.2013.2276311.
- [3] V. V. Alekseev, A. P. Emel'yanov, and A. E. Kozyaruk, "Analysis of the dynamic performance of a variable-frequency induction motor drive using various control structures and algorithms," Russian Electrical Engineering, vol. 87, no. 4, pp. 181–188, Apr. 2016, doi: 10.3103/S1068371216040027.
- [4] S. Yaacob and F. A. Mohamed, "Real time self tuning controller for induction motor based on PI method," in SICE '99. Proceedings of the 38th SICE Annual Conference. International Session Papers (IEEE Cat. No.99TH8456), Soc. Instrum. & Control Eng, pp. 909–914, doi: 10.1109/SICE.1999.788670.
- [5] F. N. Sarapulov, V. E. Frizen, E. L. Shvydkiy, and I. A. Smol'yanov, "Mathematical Modeling of a Linear-Induction Motor Based on Detailed Equivalent Circuits," Russian Electrical Engineering, vol. 89, no. 4, pp. 270–274, Apr. 2018, doi: 10.3103/S1068371218040119.
- [6] M. Konuhova, "Induction Motor Dynamics Regimes: A Comprehensive Study of Mathematical Models and Validation," Applied Sciences, vol. 15, no. 3, p. 1527, Feb. 2025, doi: 10.3390/app15031527.
- [7] C. W. de Silva, Intelligent Control. CRC Press, 2018. doi: 10.1201/9780203750513.
- [8] Y. A. Almatheel and A. Abdelrahman, "Speed control of DC motor using Fuzzy Logic Controller," in 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), IEEE, Jan. 2017, pp. 1–8. doi: 10.1109/ICCCCEE.2017.7867673.
- [9] L. A. Zadeh, "Fuzzy Logic," 2009, pp. 19–49. doi: 10.1007/978-1-0716-2628-3\_234.
- [10] A. Sagdatullin, "Improving Automation Control Systems and Advantages of the New Fuzzy Logic Approach to Object Real-Time Process Operation," in 2019 1st International Conference on Control Systems, Mathematical Modelling, Automation and Energy Efficiency (SUMMA), IEEE, Nov. 2019, pp. 256–260. doi: 10.1109/SUMMA48161.2019.8947538.
- [11] M. Masoumi, S. Hossani, F. Dehghani, and A. Masoumi, THE CHALLENGES AND ADVANTAGES OF FUZZY SYSTEMS APPLICATIONS A PREPRINT. 2020. doi: 10.13140/RG.2.2.22310.96328.
- [12] A. G. M. A. Aziz, A. Y. Abdelaziz, Z. M. Ali, and A. A. Z. Diab, "A Comprehensive Examination of Vector-Controlled Induction Motor Drive Techniques," Energies (Basel), vol. 16, no. 6, p. 2854, Mar. 2023, doi: 10.3390/en16062854.
- [13] A. Mishra and P. Choudhary, "Speed control of an induction motor by using indirect vector control method," International Journal of Emerging Technology and Advanced Engineering, vol. 2, no. 12, pp. 144–150, 2012.
- [14] G. Satyanarayana, M. Karthikeyan, R. Mahalakshmi, and T. Vandarkuzhali, "Vector Control of an Induction Motor for Speed Regulation," in 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), IEEE, Feb. 2023, pp. 1621–1625. doi: 10.1109/ICCMC56507.2023.10084248.
- [15] F. Kiyomarsi and B. Zamani, "Extending the Lifetime of Wireless Sensor Networks Using Fuzzy Clustering Algorithm Based on Trust Model," Journal of Optimization of Soft Computing (JOSC), vol. 1, no. 1, pp. 12–22, Sep. 2023, doi: 10.82553/josc.2023.14020714783332.
- [16] A. Banitalebidehkordi, "Using the fuzzy methods to examine changes in brain lesions and atrophy from MRI images for rapid diagnosis of MS," Journal of Optimization of Soft Computing (JOSC), vol. 2, no. 1, pp. 19–25, Jun. 2024, doi: 10.82553/josc.2024.140302141118861.
- [17] A. Jahanbakhsh and M. Jahangiri, "Bridging Technology and Language: Exploring Soft Computing Solutions for Effective English Language Teaching in Iran," Journal of Optimization of Soft Computing (JOSC), vol. 2, no. 3, pp. 1–6, Nov. 2024, doi: 10.82553/josc.2024.140305181128542.

Paper Type (Research paper)

# Learning Rate Optimization of U-Net Architecture Using Grasshopper Optimization Algorithm to Enhance Accuracy in CT Image Segmentation of COVID-19 Patients

Alireza Mehravin<sup>1\*</sup>, Mostafa Zaare<sup>1</sup>, Reza Mortazavi<sup>2</sup>

1. School of Mathematics and Computer Science, Damghan University, Damghan, Iran.

2. School of Engineering, Computer Department, Damghan University, Damghan, Iran.

## Article Info

### Article History:

Received: 2025/04/28

Revised: 2025/06/11

Accepted: 2025/06/11

DOI:

### Keywords:

Covid-19, GOA, Image segmentation, Metaheuristic, U-Net.

\*Corresponding Author's Email Address:  
[alireza.mehravin@gmail.com](mailto:alireza.mehravin@gmail.com)

## Abstract

In light of its excellent learning accuracy and rate, rapid data processing, and independence from large databases for network training, the U-Net architecture is a well-known and popular deep learning architecture for image segmentation and feature extraction. Learning rate selection and updating are crucial in network training. As U-Net is a completely nonlinear network, classical mathematical optimization algorithms increase the probability of local optima. This analytical research paper used the grasshopper optimization algorithm (GOA) as a metaheuristic approach to optimize the learning rate of U-Net. The network was trained using 256\*256 CT images of the lungs of COVID-19 infected and uninfected individuals. A total of 400 CT images were employed as the training dataset, whereas 80 CT images were used as the testing data. Coding was implemented in MATLAB. The optimization of the learning rate enhanced image segmentation accuracy by 2.23%. Iterative metaheuristic algorithms would lead to longer network training times. However, the proposed network optimization method could be very useful when large databases are not available for network training and higher accuracy is preferred over time savings.

## 1. Introduction

U-Net is a convolutional neural network proposed by researchers at the University of Fribourg in 2015 for biomedical image segmentation purposes. Designed through convolutional networks, the U-Net architecture accelerates processing and enhances learning accuracy based on limited training data samples. Figure 1 depicts the U-Net architecture and the operations in its different layers [1, 2].

The architecture of U-Net is based on an encoder-decoder model, where the encoder part of the network learns to extract high-level features from the input image, while the decoder part of the network learns to reconstruct the output image from the learned features. The U-Net model also includes skip connections between the encoder and decoder layers. These skip connections allow the decoder layers to use the features learned by the corresponding encoder layers, which helps to

preserve spatial information and improve the accuracy of segmentation [3].

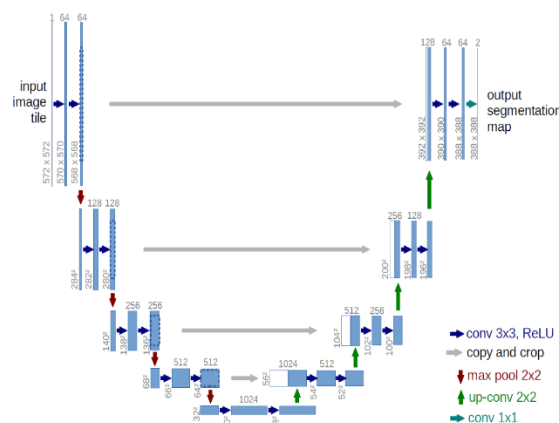


Figure 1. U-Net architecture.

One of the key advantages of U-Net is its ability to perform well with limited training data. This is

achieved by using data augmentation techniques such as rotation, flipping, and scaling during training. Additionally, U-Net can be modified to work with various input sizes, making it a versatile architecture for a wide range of image segmentation tasks. U-Net has been extensively implemented in medical imaging applications, such as tumor detection and cell segmentation, as well as in other image segmentation tasks, including the segmentation of roads and buildings in satellite imagery [4, 5]. The accuracy of the network's learning can be enhanced by employing appropriate parameters for its training.

## 2. Materials and Methods

To obtain optimal convergence in the learning process of a neural network, it is necessary to optimize the loss function [6]. Extensive and useful interactions between optimization and machine learning methods have been important breakthroughs in state-of-the-art computing. Many optimization problems in engineering sciences are complex and cannot be solved through conventional optimization approaches, such as mathematical programming. Heuristic (or approximate) algorithms can be used to solve such problems. These algorithms would not guarantee that the optimal solution is the exact solution to the problem and can only obtain a relatively accurate solution in a long time; the accuracy of the solution varies with time [7, 8].

## 3. Data Description

Preprocessing is necessary to enhance the quality of image segmentation. The preprocessing stage involves various operations such as image size reduction, noise reduction, and histogram equalization [9]. To ensure the network is trained more accurately and to reduce the computational load, the dimensions of the images were scaled down to 256x256. To train the network, a dataset of lung CT images of COVID-19 infected patients and uninfected individuals was used. A total of 400 CT images were utilized as the training dataset, whereas 80 CT images were used as the testing data. This dataset is publicly available on the Kaggle website. MATLAB R2020b was used to implement the codes on a computer equipped with an intel core i7 processor running at 2.8 GHz, along with 16GB of RAM and 4GB GPU.

## 4. Proposed Method

In the U-Net architecture, it has been observed that the initialization of the learning rate among its hyperparameters has a significant impact on

the accuracy of the network [10]. During the training process of the network, a certain scheduling strategy is employed to gradually decrease the initial learning rate, which in turn helps to minimize the loss function. Two hyperparameters affect the performance of the training process, i.e., the learning rate drop factor and the learning rate drop period. The former is a value between 0 and 1 that determines the rate at which the learning rate decreases while the latter is the number of epochs after which the learning rate is decreased [11].

As U-Net is a completely nonlinear network, the use of classical mathematical optimization algorithms increases the probability of local optima. Hence, metaheuristic algorithms can be employed to obtain solutions closer to global optima at the cost of execution time [12]. Learning rate selection and updating are essential. A sub-optimal learning rate excessively lengthens the convergence of the network, leading to trapping in local minima. On the other hand, an over-optimal learning rate would diminish network performance, with the network likely to neglect the most optimal solution to the problem [13]. It should also be noted that the initiation of the learning rate is crucial since different initiation points result in different paths, playing a key role in local and global minima [14]. Equation (1) represents the role of the learning rate ( $\eta$ ) in network training.

$$\theta = \theta - \eta \frac{\partial L(\theta)}{\partial \theta} \quad (1)$$

where  $\theta$  represents the parameter that minimizes the loss function ( $L$ ). This parameter can be assumed to denote network weights [15]. The scheduling of the learning rate can be used to improve stochastic gradient descent performance to update the weights. An optimal learning rate minimizes the iterations of the stochastic gradient descent and, thus, reduces the computational burden [16].

This paper adopted the grasshopper optimization algorithm (GOA) to optimize the learning rate hyperparameter in U-Net training to enhance image segmentation accuracy. A visual representation of the step-by-step process used in our proposed method can be seen in Figure 2.

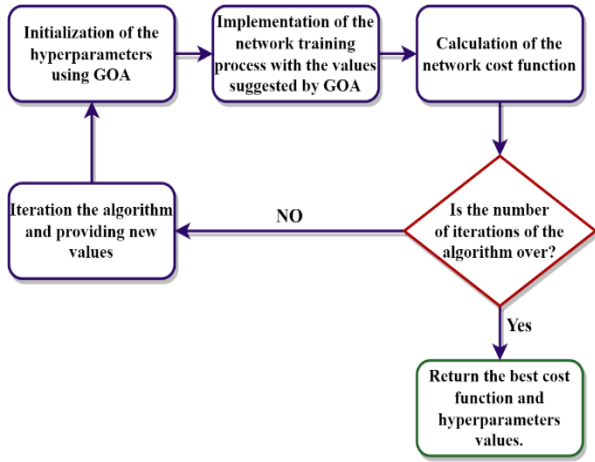


Figure 2. Diagram of the proposed method.

## 5. GOA

The GOA was introduced by Mirjalili et al. (2017). It is a metaheuristic algorithm inspired by the behavior of grasshopper swarms. It exploits swarm intelligence and is a population-based algorithm.

Exploration and exploitation are the two main features of metaheuristic algorithms. The configuration of a metaheuristic algorithm is determined by the interaction of these two features. Mirjalili et al. (2017) claimed that the lifecycle of grasshoppers intrinsically had the exploration and exploitation features; immature grasshoppers (nymphs) have soft, continuous movements and play the exploitation role, while mature ones have completely stochastic and jumpy movements and play the exploration role. These two features simultaneously exist in grasshopper swarms; therefore, efficient modeling of grasshopper behavior would provide a relatively powerful optimization algorithm. The GOA model for position updating of each grasshopper is written as:

$$x_i^d(t+1) = c \left( \sum_{j=1, j \neq i}^{n_{pop}} c \frac{ub_d - lb_d}{2} s(|x_j(t) - x_i(t)|) \frac{x_j(t) - x_i(t)}{d_{ij}(t)} \right) + \hat{T}_d(t) \quad (2)$$

where  $ub_d$  is the upper bound and  $lb_d$  denotes the lower bound of the solution space in dimension  $d$ . Furthermore,  $\hat{T}_d(t)$  denotes the best solution found until iteration  $t$  in dimension  $d$ , and  $c$  is a decreasing factor.

The social force function is defined as:

$$s(d) = fe^{\frac{d}{l}} - e^{-d} \quad (3)$$

where  $d$  is the function input and represents the distance,  $f$  is the intensity of attraction, and  $l$  represents the attractive length scale. Here, function  $s$  stands for the effects of social interactions (attraction and repulsion) among

grasshoppers. The behavior of this function is dependent on  $f$  and  $l$  values.

The adaptive parameter of  $c$  appears twice in equation (2). The outer  $c$  is very similar to the inertia weight  $w$  in particle swarm optimization (PSO) and reduces the movements of grasshoppers around the target [17-19]. In other words, this parameter moderates the exploration and exploitation of the swarm around the target. The inner  $c$  reduces the attraction zone, comfort zone, and repulsive zone between grasshoppers. This factor linearly reduces the space in which grasshoppers are to explore and exploit based on  $c \frac{ub_d - lb_d}{2}$  in equation (2).

Figure 3 depicts the GOA procedure.

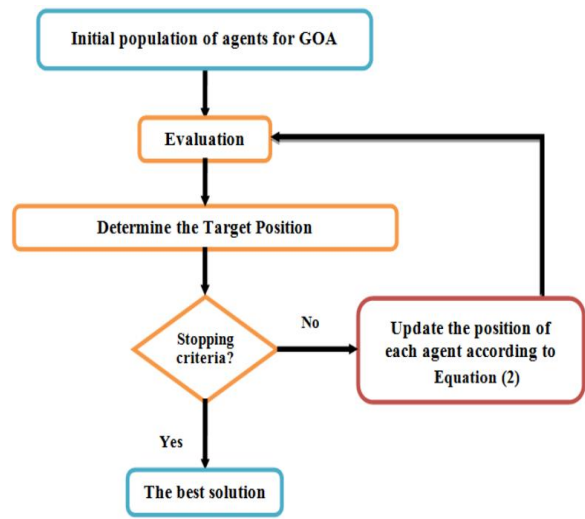


Figure 3. GOA flowchart.

## 6. Results

When initializing a neural network and selecting its associated hyperparameters, there are typically two approaches that can be taken. In the first scenario, values are assigned based on empirical knowledge or prior experience with similar models. The second scenario involves using a metaheuristic algorithm to search for optimal solutions within a range of values that we know empirically has a higher probability of network convergence.

The initial learning rate, learning rate drop factor, and learning rate drop period were empirically set to 0.003, 0.4, and 8 respectively in the first approach. In contrast, these values were determined by the GOA in the proposed method. It is worth mentioning that in both scenarios, the number of epochs was set equal to 80.

Figure 4 shows the values proposed by the GOA for the three hyperparameters and the best cost value in the last ten iterations of the algorithm. It should be noted that 20 grasshoppers and 30 iterations were employed in the GOA.

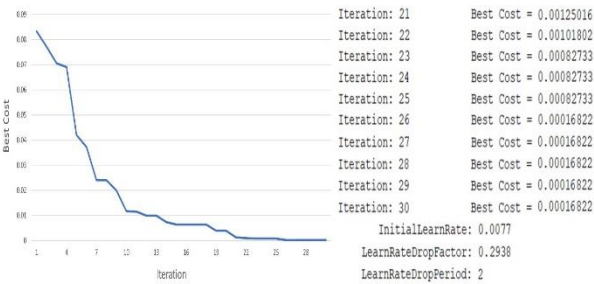


Figure 4. Values proposed by the GOA.

The normalized confusion matrix was calculated, as shown in Figure 5. Two classes were employed in network training; one label to highlight the COVID-19 damaged areas and one label to represent the background.

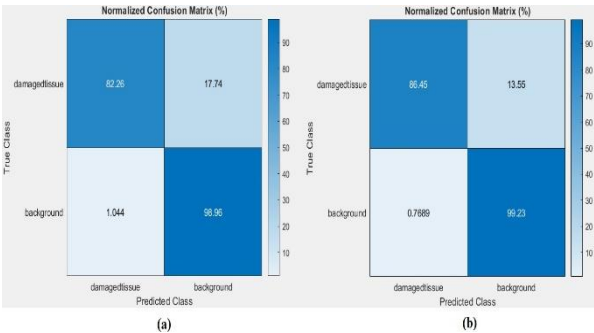


Figure 5. a) U-Net confusion matrix, b) Optimized U-Net confusion matrix.

Table1 . Base and optimized CT image segmentation

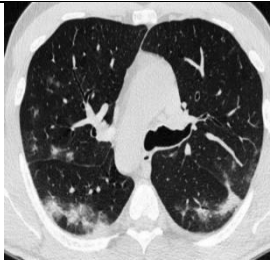
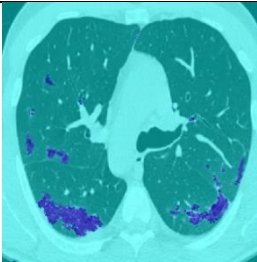
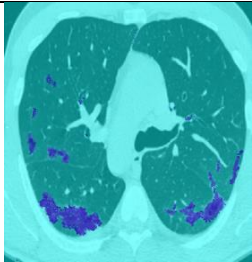

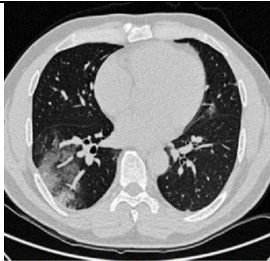

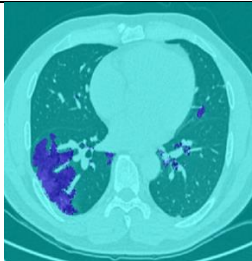

Input Image	U-Net	Optimized U-Net	Ground Truth
			
			

Table 1 provides several tested samples and the results of base and optimized segmentation.

6.1. Evaluation Results

Performance was evaluated using the true positive, true negative, false positive, and false negative parameters:

TP: Correct classification of pixels corresponding to damaged areas of the lung,

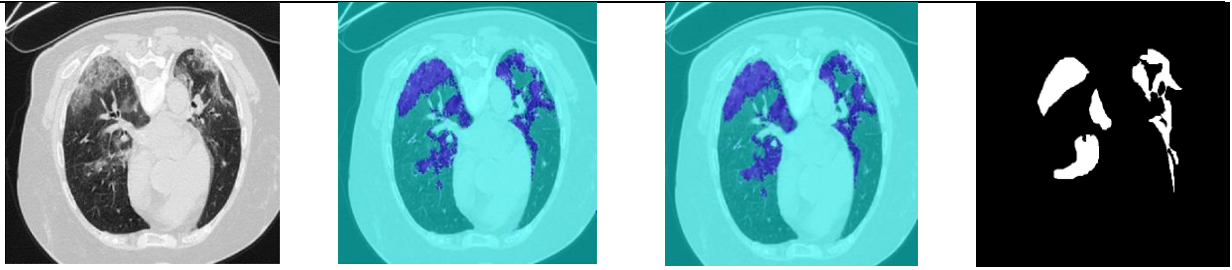
TN: Correct classification of pixels in background regions,

FP: Pixels corresponding to background regions incorrectly classified as damaged regions,

FN: Pixels corresponding to damaged areas of the lung that are incorrectly classified as background regions.

In a CT image of the chest, the positive class refers to the damaged lung areas, while the negative class corresponds to the background.





Equations (4-7) represent the evaluation criteria [20- 23].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = Sensitivity = \frac{TP}{TP + FN} \quad (6)$$

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (7)$$

Table 2 compares the base and optimized U-Net models based on the confusion matrix and evaluation criteria.

**Table 2. Evaluation results(%)**

Evaluation Criteria	U-Net	Optimized U-Net
Accuracy	90.61	<b>92.84</b>
Precision	98.74	<b>99.11</b>
Recall	82.26	<b>86.45</b>
F1-Score	89.74	<b>92.34</b>

## 7. Discussion

As a result of the optimization conducted in this research, the accuracy of the entire network for CT scan image segmentation has observed an increase of 2.23%. Furthermore, it is worth noting that the accuracy has significantly improved by 4.19% in detecting the damaged area. It is noteworthy that the optimization method discussed here can be applied to all deep learning models during the learning process. The utilization of meta-heuristic algorithms for solving non-linear problems is a widely accepted practice. In this research, the GOA was employed to optimize the learning rate of the network. In a study closely related to the present research, Mahesh Kumar et al. were able to enhance the accuracy of segmentation in MRI images of brain tumors through the utilization of a metaheuristic algorithm known as the Adaptive Search Coyote Optimization Algorithm (AS-COA) [24]. Another study carried out by Popat et al. demonstrated that the accuracy of a neural network for the

segmentation of Retina Blood Vessels can be improved by optimizing its parameters using the genetic algorithm [25]. Nevertheless, the challenge of increasing network learning time cannot be overlooked. Also, evaluating the impact of optimizing other critical hyperparameters during the network training process, such as batch size, can be investigated.

## 8. Conclusion

This paper implemented GOA to optimize the U-Net learning rate and enhance the segmentation accuracy of CT images of the chest to enable accurate detection of COVID-19 infected areas within the lungs. Although iterative metaheuristic algorithms lengthen training, they are very useful in network optimization when large databases are not available for network training and improved accuracy is preferred over time savings. To optimize hyperparameters in such models, it is important to assume a larger grasshopper population and a larger number of iterations to obtain a more optimal solution. In other words, this optimization approach is more useful in network training with smaller databases and low-resolution images.

## Acknowledgement

This article is derived from the master's thesis of Alireza Mehravin. University of Damghan, Damghan, Iran.

## References

- [1] Ronneberger, O., Fischer, P., & Brox, T. (2015). "U-NET: Convolutional Networks for Biomedical Image Segmentation". *arXiv (Cornell University)*.  
<https://doi.org/10.48550/arxiv.1505.04597>.
- [2] Shamim, S., Awan, M. J., Zain, A. M., Naseem, U., Mohammed, M. A., & Garcia-Zapirain, B. (2022). "Automatic COVID-19 Lung Infection Segmentation through Modified Unet Model". *Journal of Healthcare Engineering*, 2022, 1–13.  
<https://doi.org/10.1155/2022/6566982>.
- [3] Saood, A., & Hatem, I. (2021). "COVID-19 lung CT image segmentation using deep learning methods: U-Net versus SegNet". *BMC Medical*

- Imaging*, 21(1). <https://doi.org/10.1186/s12880-020-00529-5>.
- [4] Kalane, P., Patil, S. D., Patil, B. A., & Sharma, D. P. (2021). "Automatic detection of COVID-19 disease using U-Net architecture based fully convolutional network". *Biomedical Signal Processing and Control*, 67, 102518. <https://doi.org/10.1016/j.bspc.2021.102518>.
- [5] Raj, A. N. J., Zhu, H., Khan, A., Zhuang, Z., Yang, Z., Mahesh, V. G. V., & Karthik, G. (2021). "ADID-UNET—a segmentation model for COVID-19 infection from lung CT scans". *PeerJ*, 7, e349. <https://doi.org/10.7717/peerj-cs.349>.
- [6] Kaur, S., Kumar, Y., Koul, A., & Kamboj, S. K. (2022). "A Systematic Review on Metaheuristic Optimization Techniques for feature selections in Disease Diagnosis": Open Issues and Challenges. *Archives of Computational Methods in Engineering*, 30(3), 1863–1895. <https://doi.org/10.1007/s11831-022-09853-1>.
- [7] Antoniou, A., & Lu, W. (2021). *Practical Optimization: Algorithms and engineering applications*. <https://doi.org/10.1007/978-1-0716-0843-2>
- [8] Wong, W., & Ming, C. I. (2019). "A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications". 2019 7th *International Conference on Smart Computing & Communications (ICSCC)*. <https://doi.org/10.1109/icscc.2019.8843624>.
- [9] Yang, D., Martinez, C., Visuña, L., Khandhar, H. M., Bhatt, C., & Carretero, J. (2021). "Detection and analysis of COVID-19 in medical images using deep learning techniques". *Scientific Reports*, 11(1). <https://doi.org/10.1038/s41598-021-99015-3>.
- [10] Elaziz, M. A., Dahou, A., Abualigah, L., Yu, L., Alshinwan, M., Khasawneh, A. M., & Lu, S. (2021). "Advanced metaheuristic optimization techniques in applications of deep neural networks: a review". *Neural Computing and Applications*, 33(21), 14079–14099. <https://doi.org/10.1007/s00521-021-05960-5>.
- [11] Takase, T., Oyama, S., & Kurihara, M. (2018). "Effective neural network training with adaptive learning rate based on training loss". *Neural Networks*, 101, 68–78. <https://doi.org/10.1016/j.neunet.2018.01.016>.
- [12] Nematzadeh, S., Kiani, F., Torkamanian-Afshar, M., & Aydin, N. (2022). "Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases". *Computational Biology and Chemistry*, 97, 107619. <https://doi.org/10.1016/j.compbiolchem.2021.107619>.
- [13] Kaveh, A., & Hamedani, K. B. (2022). "Advanced metaheuristic algorithms and their applications in structural optimization". In *Springer eBooks*. <https://doi.org/10.1007/978-3-031-13429-6>.
- [14] Li, C., Jiang, J., Zhao, Y., Li, R., Wang, E., Zhang, X., & Zhao, K. (2021). "Genetic Algorithm based hyper-parameters optimization for transfer Convolutional Neural Network". *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2103.03875>.
- [15] Thavasimani, K., & Srinath, N. K. (2022). "Hyperparameter optimization using custom genetic algorithm for classification of benign and malicious traffic on internet of things–23 dataset". *International Journal of Power Electronics and Drive Systems*, 12(4), 4031. <https://doi.org/10.11591/ijece.v12i4.pp4031-4041>.
- [16] Gaspar, A., Oliva, D., Cuevas, E., Zaldivar, D., Pérez, M. A., & Pajares, G. (2021). "Hyperparameter optimization in a convolutional neural network using metaheuristic algorithms". In *Studies in computational intelligence* (pp. 37–59). [https://doi.org/10.1007/978-3-030-705428\\_2](https://doi.org/10.1007/978-3-030-705428_2).
- [17] Okwu, M. O., & Tartibu, L. K. (2021). "Metaheuristic Optimization: Nature-Inspired algorithms swarm and computational intelligence, theory and applications". In *Studies in computational intelligence*. <https://doi.org/10.1007/978-3-030-61111-8>.
- [18] Saremi, S., Mirjalili, S., & Lewis, A. (2017). "Grasshopper Optimisation Algorithm: Theory and application". *Advances in Engineering Software*, 105, 30–47. <https://doi.org/10.1016/j.advengsoft.2017.01.004>.
- [19] El-Shorbagy, M. A., & El-Refaey, A. M. (2020). "Hybridization of Grasshopper optimization algorithm with genetic algorithm for solving system of Non-Linear equations". *IEEE Access*, 8, 220944–220961. <https://doi.org/10.1109/access.2020.3043029>.
- [20] Fazli, M., & Faraji, S. (2023). A survey of meta-heuristic methods for optimization problems. *Journal of Optimization in Soft Computing*, 1(1), Article 14020615783207. <https://doi.org/10.82553/josc.2023.14020615783207>.
- [21] Wang, R., Lei, T., Cui, R., Zhang, B., Meng, H., & Nandi, A. K. (2022). "Medical image segmentation using deep learning: A survey". *Iet Image Processing*, 16(5), 1243–1267. <https://doi.org/10.1049/ipr2.12419>.
- [22] Polat, H. (2022). "Multi-task semantic segmentation of CT images for COVID-19



infections using DeepLabV3+ based on dilated residual network”. *Physical and Engineering Sciences in Medicine*, 45(2), 443–455. <https://doi.org/10.1007/s13246-022-01110-w>.

[23] Kiyoumars, P., Kiyoumars, F., Zamani Dehkordi, B., & Karbasiyoun, M. (2024). A feature selection method on gene expression microarray data for cancer classification. *Journal of Optimization in Soft Computing*, 2(3), Article 140308101189068. <https://doi.org/10.82553/josc.2024.140308101189068>.

[24] Kumar, G. M., & Parthasarathy, E. (2023). “Development of an enhanced U-Net model for brain tumor segmentation with optimized architecture”. *Biomedical Signal Processing and Control*, 81, 104427. <https://doi.org/10.1016/j.bspc.2022.104427>.

[25] Popat, V., Mahdinejad, M., Dalmau, O., Naredo, E., & Ryan, C. (2020). “GA-based U-Net Architecture Optimization Applied to Retina Blood Vessel Segmentation”. *Proceedings of the 12th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science and Technology Publications. <https://doi.org/10.5220/001011220192>.