Algorithms and Applications in Complex Network Representation, Classification and Manipulation

Anjan Chowdhury
CSCR, ISI Kolkata

Indian Statistical Institute

July 2024

INDIAN STATISTICAL INSTITUTE

DOCTORAL THESIS

Algorithms and Applications in Complex Network Representation, Classification and Manipulation

Author:
Anjan Chowdhury
CSCR, ISI Kolkata

Supervisors:
Dr. Kuntal Ghosh
Associate Professor
MIU, ISI-Kolkata, India
Dr. Sanjukta Bhowmick
Associate Professor
CSE, UNT, USA

A thesis submitted to the Indian Statistical Institute
in partial fulfilment of the requirements for
the degree of
Doctor of Philosophy (in Computer Science)

Center for Soft Computing Research

Indian Statistical Institute, Kolkata

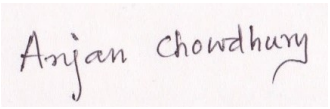
July 2024

Dedicated to my father

Acknowledgements

My deepest thanks and respect for my research advisors, Drs. Kuntal Ghosh and Sanjukta Bhowmick, cannot be expressed in words. They gave me the motivation to work independently as a researcher and showed me the value of critical thinking. I am extremely appreciative of my collaborators, colleagues, friends and well wishers: Prof. Animesh Mukherjee, Dr. Swarup Chattopadhyay, Dr. Sriram Srinivasan, Dr. Rajdeep Chatterjee, Dr. Jayanta Kumar Pal, Dr. Abhinav Chakraborty, Dr. Rahul Dasgupta, Suman Midya, Subhendu Chakraborty, Geetanjali Aich, Adarsh Raj, Snigdha Agarwal, Jogindar Singh, Susanta Samanta, Bibhuti Das, Anupam Sarkar, Sukriti Roy, Keerthi S Chandran, Rajdeep Das, Roshni Mondal, Srutiparna Neogi, Sandipa Roy, Shibsankar Roy, Barnini Bhattacharya and Mr. Pawan Kumar for sharing their knowledge and intuition with me in order to solve my technical and scientific problems.

Anjan Chowdhury
July 2024



Digitally signed by Anjan Chowdhury
DN: cn=Anjan Chowdhury, o=Indian Statistical Institute, ou=Center for Soft Computing and Research (CSCR), email=anjanchowdhury88@gmail.com, c=IN
Date: 2024.08.21 14:39:16 +05'30'

Abstract

A complex network is a useful model for many real-world systems. Recently, much effort has been put into studying the insights of the complex network. This thesis is all about the study of complex networks. Based on the study, this thesis can be broadly divided into three parts: The first one involves analysing a complex network to find a crucial network structure called constant community by extracting and applying some features called graph representations. The second part involves the study of the quality of the graph representations on a downstream task, i.e., the node classification task. In the third part, we tried to apply the handcrafted and automatically learned graph features to some real-world scenarios, i.e., in brain networks.

While detecting the constant community, we developed two strategies to construct and use the graph representations: semi-supervised and unsupervised. In the semi-supervised approach, we converted the original graph to its corresponding line graph, where a node in the line graph represents an edge in the original graph. We then applied a graph neural network (GNN) as a graph representation learning tool to classify the nodes in the line graph, which in turn was used to capture the constant communities in the original graph. In the unsupervised approach, using some hand-crafted features for each edge in the original network, we developed some novel algorithms inspired by image thresholding algorithms to filter out the non-constant community edges and hence find the constant communities.

In the semi-supervised approach, we noticed that when we reduced the number of training nodes, the representational capability of GNN decreased, and as a result, the classification accuracy of GNN drastically dropped. This phenomenon led us to develop input and output intervention methods to improve the accuracy of the GNN. In the input intervention, we extend the training nodes' set using random walk and some machine learning methods to agnostically capture similar nodes from various non-contiguous sub-networks in a whole network. In the output intervention, we used random walk methods to correctly relabel the possibly misclassified nodes by the GNN as its output.

The last part of the thesis deals with applications of network representation, classification, and finally manipulation in dealing with complex human brain networks. The brain regions and their interrelationships can be modelled using complex network. Utilising the complex network and its representation, in this part we contributed to neuroscience in two ways: first, we devised a methodology to diagnose a neurodevelopmental disease called Attention Deficit Hyperactivity Disorder (ADHD) using some extracted network features and applied them to various deep learning-based models. Then in the second work, we built a probabilistic model using anatomical and topological similarities to generate synthetic brain networks and track down the progression of a neurodegenerative disease called Alzheimer's disease (AD) in human brains. The results are promising enough to establish the use of complex network analysis in computational neurology.

Contents

Chapter 1

Introduction

Network or Graph[1]**.** During the COVID-19 pandemic, how did the Corona virus spread? How many friends are there between the two randomly chosen individuals? How does a rumour spread in society? How is it possible for a human brain to function so effectively when each brain cell is so basic? - Networks play an important role in a variety of such problems that may appear unrelated. Historically, graph theory, a subfield of discrete mathematics, has dominated the study of graphs or networks [1] and was first developed by a Swiss mathematician, Leonhard Euler, in 1735 with his solution to the popular Königsberg bridge problem. Recently, networks have received plenty of interest from a variety of sectors and can be used as a foundation for the mathematical modelling of many different complex systems. A graph or a network can be easily understood as a set of objects (referred to as nodes or vertices) and a set of connections (referred to as links or edges) between these objects. Based on the underlying topology, a graph can be classified into various classes, such as regular graph [2], random graph [3], complex graph [4] etc.

A **regular graph** is one in which every vertex has the same number of neighbours, or the same degree or valency. Starting with a collection of isolated vertices, a random set of edges connecting them results in a **random graph**. When a network's collective behaviour cannot be predicted from its constituent parts, it is said to be a **complex network**.

[1] In this thesis we synonymously use these two terms

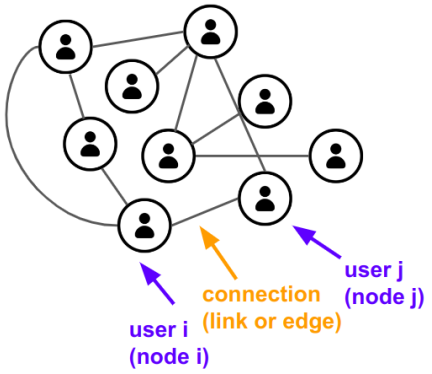


FIGURE 1.1: Social network representation with graph

Complex Network. The idea of complex networks was first found in two seminal papers: one by Watts and Strogatz [5] and others by Barabasi and Albert [6]. A complex network has some topological properties that cannot be seen in a regular or random network. These properties include a high clustering coefficient, a heavy tail in the degree distribution, assortativity (or disassortativity) among vertices, hierarchical structure, and community structure. Even though the nodes and linkages may have diverse interpretations, complex networks exhibit significant statistical and topological similarities. Many real-world systems can be effectively modelled using complex networks. Examples include: the social network [7] where the nodes stand in for the individuals and the edges between the nodes (individuals) signify that they are friends (FIGURE. 1.1), Neuroscience [8] where the brain regions can be represented as nodes and the correlations between them can be represented by edges, NLP [9] where words are represented as nodes and an edge exists between two words if they are synonyms. Other domains include computer graphics [10], recommendation systems [11], transportation networks [12] etc. In addition to these, there are some well-known examples of real-world networks created by linking a few nodes, such as the football network [13], karate club network [14], Les Miserable [15], Dolphin social network [16] etc. Studying the topologies of complex networks, however, has proven to be a difficult challenge. There is no one location where one can get a complete image of the topology because a large-scale network is typically made up of dozens or even millions of nodes. Furthermore, networks are rapidly evolving and changing over time. Various random graph theory-based models had been put forth in the early stages of complex network research to represent the topology of a complex network. Erdos and Renyi [17] developed one of the earliest theoretical models of a complex network where

the nodes are connected at random. Unfortunately, when modelling real-world complex networks using a random graph, the structures of these graphs fail to capture the structural characteristics because most of the vertices have nearly the same degree. Due to these difficulties, a number of new network models [18, 19] have been put forth that can characterise the similarities and differences in the structures and functions of actual complex systems and networks across a range of domains.

Concept of Communities in a network. The structural properties of the complex network provide insights into the characteristics of the underlying complex system. Most of the research in network analysis focuses on identifying network properties that map to interesting features of complex networks and developing efficient algorithms to compute these properties. As mentioned earlier, one of the important properties of a complex network is the presence of community within it. A community is a group of nodes where, within this group, nodes are more densely connected with the edges than outside the group. A wide range of fields, including criminology [20], public health [21, 22], politics [23, 24], customer segmentation [25], Recommendation Systems [26, 27], Social Network Analysis [28], etc., can benefit from community detection.

There are several algorithms, such as **Louvain** [29], **Infomap** [30], **label propagation** [31] etc., to find such communities. Despite its importance, there is yet no mathematical definition of what constitutes a community in a network. Instead, community detection algorithms are defined by how well a certain parameter, such as modularity [32], or entropy [33] is optimized. All these optimization problems are NP-hard, and approximations to the algorithms render the process stochastic. Even when the optimum values of the parameters are obtained, community detection algorithms can produce multiple solutions or exhibit a resolution limit. Thus, due to the nature of the problem, compounded with the algorithm approximations, community detection results can differ based on the methods used, changes in parameters, and even the order of the vertices in the input.

Therefore, one of the most challenging tasks these days is to study and analyze the community structure in the complex network in order to understand both the microscopic and macroscopic insights of the system. These studies involve finding and

analysing the community structure of a complex network by identifying several non-trivial characteristics or features in the network.

Feature engineering in the network The conventional feature engineering of network data typically focuses on obtaining a number of predefined simple features at the node, edge and subgraph levels (frequent subgraphs [34] and graph motifs [35]). Common features are: network diameter, centrality measures, average path length, clustering coefficient, etc. Although they describe a number of fundamental aspects of the graphs, those few hand-crafted, clearly defined features ignore patterns that they are unable to cover. Furthermore, since real-world network phenomena are extremely complex, they either require highly complex, unproven combinations of those predefined features or cannot be described by any of them. Furthermore, these handcrafted feature generations suffer from computational overhead, especially in large-scale networks.

Graph Representation Learning (GRL) [36] on the other hand, has received plenty of attention in recent years. While maintaining the network topology, vertex/edge content, and other side information, GRL aims to learn latent, low-dimensional representations of network vertices/edges. Network analytic tasks can be completed quickly and effectively by applying traditional vector-based machine learning algorithms to the new representation space after new vertex/edge representations have been learned. Traditional graph embedding includes K-nearest neighbours (KNN) [37] based techniques in association with the multidimensional scaling (MDS) method [38], locally linear embedding (LLE) [39], laplacian eigenmaps (LE) [40] etc. Modern graph embedding techniques such as random walk [41], matrix factorization [42], Graph Neural Networks (GNN) [43, 44], etc. are considered the generalization of these traditional embedding techniques.

In this thesis, we begin with the study and analysis of community structure using various handcrafted and automated features. Following this, an investigation of modern automated feature generation tools, the Graph Neural Network (GNN), is performed here. In addition, we also take care of some of the applied parts, i.e., how we can apply the learned features or representations in the various real-life applications

such as neuroscience, recommendation systems, etc.

1.1 Motivation and Objectives

Recently, complex networks have been used to mimic a broad spectrum of real-world systems like social networks, biological networks, ecological networks, recommendation systems, etc. Modelling these real-world systems results in huge success in finding several hidden but important pieces of information, such as groups in a social network involved in criminal activities, functional groups that cause cell proliferation or cell death, etc. One such crucial structural characteristic, community structure, has started to emerge in the continuous evolution of real-world complex networks and helps us get such important information. But the investigation of communities is a challenging task due to the stochastic nature of the community detection algorithms. Recent advancements in feature engineering techniques open up a doorway to studying several important characteristics of a network. This motivates us to begin with a deeper investigation of the communities inside the complex network by developing several algorithms using some hand-crafted as well as automated feature engineering techniques. Now, developing algorithms using such feature engineering techniques comes up with the following important questions: how good is the constructed graph representation (especially using the automated feature engineering technique)? Does it capture enough information about a particular network? And how can the application of graph representations help us in real-life situations?

The objective of this thesis is to develop and apply algorithms to study the changing nature of the community structures in a complex network using hand-made and automated feature engineering techniques, analyse the graph representation learning technique itself to study how well it represents a network, and apply them in some real-life scenarios.

The thesis can be broadly categorized into three parts: (i) developing algorithms to study the invariant nodes inside the community using the graph representation technique; (ii) studying the goodness of representations of a network; and (iii) applying the

graph representations in neuroscience. Next, we'll discuss the inspiration for each of the three components before moving on to the thesis' chapter-by-chapter contributions.

1.1.1 Part I: Developing algorithms to study the invariant nodes inside the community using Graph Representation technique

From social networks to neuroscience to politics, community structure in a network yields significant information. While there are a few well-known techniques for the detection of communities in a complex network, they tend to be stochastic in nature, which leads to changing the community structure in a network during each execution of the community detection algorithms. Since communities are important because of their diverse applicability in real-world systems, changing the community structure or moving the nodes from one group to another in different executions creates severe problems, like the accurate determination of criminal groups in a social network or harmful protein groups in a biological network, etc. Thus, the determination of the nodes that never leave their corresponding group is important. One simple approach is to execute a particular community detection algorithm several times and track the nodes' belongingness. But this approach is computationally expensive, especially for large networks. This problem makes us start thinking about the following: can we quickly identify these invariant sets of nodes? One possible solution can be found in the properties of the network itself, and therefore, this motivates us to use the network features or representations along with developing algorithms.

1.1.2 Part II: Study the goodness of representations of a network

In Part I, we identified the invariant nodes within the community using graph representations, and we employed Graph Neural Network (GNN) as one of the Graph Representation Learning (GRL) techniques. Variants of Graph Neural Networks (GNNs) have recently been utilised as a key tool in a variety of these downstream tasks, such as node categorization [43], link prediction [45] etc. The effectiveness of these activities depends on how well the network's node- or edge-level information is captured. Although they sometimes learn the graph representation very well, they occasionally

miss the mark. This observation inspires us to investigate the GNN's learning in this section.

1.1.3 Part III: Application of the complex network and its representations in neuroscience

Complex network models are becoming applicable to a wide range of real-world applications, including social networks (such as Facebook and Instagram), biological networks (such as gene regulatory networks), recommendation systems, and many others. In case of neuroscience, the human brain can be modelled as complex network [46] where each region of the brain can be considered as node and the functional connection between two regions can be treated as edges in the network. The human brain is susceptible to a wide variety of neurodevelopmental and neurodegenerative disorders, including attention deficit hyperactivity disorder (ADHD) [47], Alzheimer's disease [48], Parkinson's disease [49], and many others. In case of neurodevelopmental disease (e.g., ADHD) if patients do not receive an early diagnosis, their ability to participate in social activities becomes severely limited. In case of neurodegenerative disease (e.g., Alzheimer's disease), it is very important to understand the present status of the patient's brain - i.e., whether a person has any tendency that in future he will be suffering from Alzheimer's disease or not. In other words, it is extremely important to track the progression towards the Alzheimer's disease from a healthy brain. For this reason, it is of the utmost importance to have an understanding of two things: (i) whether or not brain disorders can be detected at an early stage, and (ii) whether or not their progression can be monitored. Given that the human brain can be described as a complex network, the question becomes whether or not it is possible to use the properties of the network to solve these two challenges. This motivates us to employ the graph representations once more to address these problems because we already use the manually produced and automated network attributes to tackle the preceding concerns.

1.2 Organization of the thesis

The background and original contributions made in the various chapters of this thesis will be discussed in this section. There are seven chapters in this thesis, of which four

(Chapters $3-6$) contribute. In Chapter 2, we discuss all the preliminary and related works necessary for the rest of the chapters.

Part I includes only Chapter 3. This is the first contributing chapter, it deals with the problem of finding invariant nodes inside the community in a complex network with two methods - semi-supervised and unsupervised.

Part II includes only Chapter 4. In Chapter 3, we used an automated feature engineering technique (the Graph Representation Learning (GRL) technique) in the semi-supervised method. We have seen that the accuracy drops when the size of the training nodes used in the semi-supervised method is very small. This is due to the fact that training a Graph Neural Network (as a GRL tool) with such a small number of training nodes results in poor learning in network representation. In this chapter, we try to resolve this problem.

Part III includes Chapters 5 and 6. Since the human brain can be represented by a complex network, we make use of the complex network and its properties or network representations to deal with the problems associated with neuroscience. Chapter 5 deals with diagnosing of a neurodevelopmental disease called Attention Deficit Hyperactivity Disorder (ADHD). In Chapter 6, we investigate the progression towards a neurodegenerative disease called Alzheimer's disease (AD) from a healthy brain.

Finally, there is a conclusion chapter that summarizes the basic findings and contributions of this thesis and indicates the future directions of research.

1.2.1 Chapter 3: Detection of constant communities in a complex network

Communities in a network are important since they can be used in diverse areas. But due to the stochastic nature of the community detection algorithms, it is very difficult to have the same community structures in a network for different executions of a particular algorithm. Since the community structure changes, few nodes change the community in which they were in the previous execution; as a result, it may hurt the applied part. For example, in a social network, we cannot properly track people who are involved in a particular activity, or in a brain network, we cannot properly determine the belongingness of a brain region (node) in a particular functional group. But there are a few nodes that never change their community, and in a real-life scenario,

it is very important to identify these invariant nodes. These sets of nodes are called constant community nodes, and the community they form is called constant community. Thus, in this chapter, our objective is to determine the set of constant communities in a complex network. For that, we have developed two novel approaches—semi-supervised and unsupervised—based on graph representation techniques. In the semi-supervised technique, we use Graph Convolutional Neural Network (GCN) [43] and line graphs [50] to determine the constant communities, and in the unsupervised technique, we have used several image thresholding approaches to do the same. Finally, we performed a case study where we employed the neighbourhood-based recommender system and showed the effectiveness of our proposed algorithms.

1.2.2 Chapter 4: Improving the accuracy of Graph Neural Network

While using a semi-supervised approach in our previous work, we have observed that if the size of the training nodes (seeds) is very small (two per class), then the classification accuracy of GCN drastically drops. Not only GCN, but we have also observed that the same problem occurs in several variants of Graph Neural Networks (GAT, GraphSAGE, etc.). Now, keeping the size of the training set small is important since a large training set defeats the purpose of a semi-supervised approach, and in some real-life situations, due to some unavoidable circumstances, it is sometimes very difficult to obtain the training seeds. Therefore, it is necessary to develop some techniques that, if applied, would keep the GNN's accuracy high even if the training set size is very small; in other words, the GNN could well capture the topological information (learn the graph representations). For that, we have discussed two such techniques: input-level intervention and output-level intervention, to achieve the solution to this problem. In the input-level intervention technique, we develop a method that agnostically increases the training set and captures nodes from different parts of the network with the same class labels. The output-level intervention is like a post-processing technique where, after a successful execution of the GNN, we tired to find a few miss-classified nodes and tried to relabel them with their correct labels.

1.2.3 Chapter 5: Diagnosing ADHD disease using complex network and its representation

Chapters 5 and 6 deal with the importance of graph representation in human brain especially in computational neurology. In this chapter, we discuss the diagnosis of a neurodevelopmental disease, Attention Deficit Hyperactivity Disorder (ADHD) using fMRI data. ADHD is one of the most common neurodevelopmental disorders in childhood. People with ADHD may struggle to focus, manage impulsive behaviours (doing things without considering the consequences), or be extremely active. Despite the fact that ADHD cannot be cured, if it is diagnosed early, it can be effectively controlled, and some symptoms may improve as the kid gets older. In order to classify the ADHD brain from a non-ADHD brain, we use complex network and its representation in our study. The relationships among the Blood Oxygen Level Dependent (BOLD) [51] signals extracted from each of the brain regions can be modelled by a connectivity matrix. Since the connectivity matrix is symmetric, it can be considered as a weighted adjacency matrix of anetwork where each region is represented as a node and their relationship can be represented as an edge. Finally, after building a handcrafted network representation (feature vectors) of the network, we feed it into some one-dimensional neural networks to diagnose the ADHD brain[2]. We also discuss how accuracy varies when we vary the brain atlas or connectivity measures.

1.2.4 Chapter 6: Investigating the progression of the brain network: from a healthy brain to Alzheimer's disease

In chapter 5, we have discussed the diagnosis of a neurodevelopmental disease called ADHD and shown how the complex network and its representation can be applied to model the fMRI data while diagnosing the disease. In this chapter, we again apply the complex network and its representation to study the progression towards Alzheimer's disease in the healthy brain using a graph manipulation approach. The majority of neurodegenerative diseases like Alzheimer's disease, Parkinson's disease, etc. are incurable and often worsen with time. It is very difficult for most people suffering from these diseases to go for a regular brain scan to track the progression of their disease. In

[2]Additionally, we visualised the connectivity matrix as an image and fed the image into some two-dimensional neural networks to distinguish the ADHD brain from the non-ADHD brain. We discussed this part in the appendix.

this chapter, we discuss the methods that keep track of the progression of Alzheimer's disease. We proposed a novel model that takes the real healthy brain network as an input and produces synthetic brain networks similar to those in Alzheimer's disease by doing a series of manipulations in the form of edge addition and deletion processes. We use both topological and anatomical similarity to develop the model. While using the topological similarities, we have used several classic methods as well as GNN models to generate the node embedding, and while using the anatomical similarities, we have used the centroid of the regions represented by 3D coordinates.

1.3 Conclusion

This thesis deals with the study of complex networks and its contributions in three broad areas: communities in complex networks, graph-based deep learning, and neuroscience. First, it tries to address an interesting problem of graph algorithms, which is finding the constant community in a complex network. Finding the communities using an optimization technique in a network itself is an NP-Hard problem. Thus the approximation of this optimization technique makes the community detection a stochastic process which results in different community structures in different runs. In this regard, the detection of the invariant nodes within communities becomes a challenging task. We have addressed this challenge using edge filtering technique by employing two novel approaches (semi-supervised and unsupervised) with the help of complex network representations. The semi-supervised approach tries to distinguish the invariant set of nodes with the help of line graph conversion of the original graph and applying Graph Convolutional Neural Network (GCN), a variant of Graph Neural Network. The unsupervised approach uses several classic image thresholding methods to predict the set of invariant nodes.

The second part contributes to the field of graph based deep learning. In this part, we continue with an observation of the first part. We have seen that the classification accuracy of a Graph Neural Network (GNN) drastically dropped when we decreased the number of training nodes. This problem occurs due to the poor representation capacity of GNN when the number of training nodes is low. In this part, we address the issue in two ways: first by implementing input-level intervention, where beginning with a very small number of nodes for each class, we agnostically extend the number

of training nodes and try to capture the similar nodes from every corner of the graph by applying random walks and some machine learning algorithms. In the output-level intervention, we again use the random walk to relabel the nodes that were wrongly predicted by GNN.

The final part of this thesis deals with specific challenges in an important applied area of complex networks, which is neuroscience. In this part, we study two challenging tasks: one is to diagnose a particular neurodevelopmental disease called ADHD, where the fMRI data are first converted into a connectivity matrix and then into a complex network, and some handcrafted network properties are collected to build a feature vector for that network to feed into some neural network for the classification task. The second task is to build a novel model to study the progression of a healthy brain network towards a brain network of a neurodegenerative disease called Alzheimer's disease (AD). Once again, we start by turning the brain's fMRI data into a complex network, and then we use the network's anatomical and topological characteristics to construct the model. The constructed model is then used in the network manipulation task to generate synthetic AD brain network from the real healthy control's (HC) brain network.

Chapter 2

Literature Survey

2.1 Concepts

2.1.1 Graph or network

Technically, a graph can be described as a collection of nodes (V) and their connecting edges (E). We can say that $(u, v) \in E$ if and only if node u and node v are connected, i.e., there is an edge between node u and node v. In this study, we shall consider a simple graph where $(u, u) \notin E, \forall u \in V$ and there exists at most one edge between u and v and all edges are undirected.

Graphs can be conveniently represented using an adjacency matrix A. We arrange the nodes in the graph so that each node indexes a specific row and column in the adjacency matrix in order to describe a graph with an adjacency matrix. The existence of an edge in a graph can then be represented by entries in the adjacency matrix: $A[u, v] = 1$ if $(u, v) \in E$ and $A[u, v] = 0$ otherwise for $u, v \in V$. As the graph in our study is undirected, the adjacency matrix is symmetric. Moreover, some graphs feature weighted edges, where the entries in the adjacency matrix take the form of arbitrary real values rather than the numbers 0 and 1. For instance, a weighted edge in a brain network illustrates the type of correlation between two brain regions.

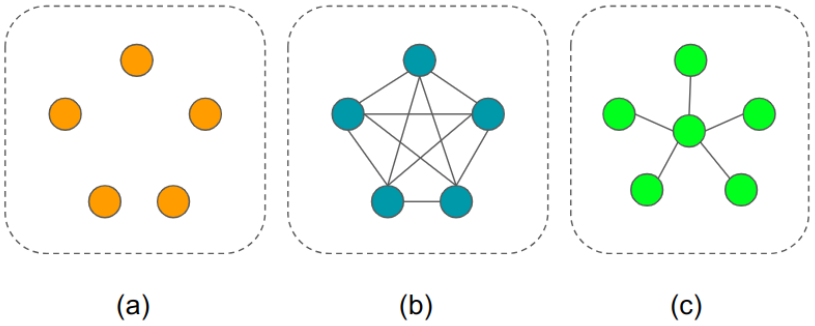


FIGURE 2.1: Examples of regular graphs (a) 0-regular graph, (b) 5-regular graph (c) Not a regular graph

2.1.2 Types of the graph or network

2.1.2.1 Regular graph

A regular network is a graph where each vertex has the same number of neighbours. A regular graph is said to be $k-$regular if all the nodes have degrees equal to k. A regular graph also has an even number of vertices with an odd degree, according to the handshaking lemma [52]. Fig 2.1 includes a few examples of regular graphs.

2.1.2.2 Random graph

A random graph is obtained by starting with a set of n isolated vertices and adding successive edges between them at random. Various random graph models exist, and they produce different results, viz., uniform random graph [53], binomial random graph [54] etc. A survey regarding the random graph model can be found in [55]. Applications of random graphs can be found in various places. A study related to the random graph model on social networks was proposed by [56]. Application of random graphs in data analysis can be found in [57]. Other applications include protein-protein interactions [58], food webs [59] etc.

2.1.2.3 Complex graph

A complex network is characterised by the inability to draw definitive conclusions about the entire network by analysing its sub-networks individually. We can define

a complex network in two ways, one is as done by Barabási and Albert [60] and another is that by Kim and Wilhelm [61]:

i. Complex networks are systems composed of interconnected elements, where the arrangement of connections displays non-trivial structural properties such as small-world phenomenon, power-law degree distributions, and community structure [60].

ii. A complex network is a network consisting of different subgraphs, where two subgraphs are different if they are non-isomorphic. The more different subgraphs a graph contains, the more complex it is [61].

Thus, a complex network is a network having non-trivial topological features that are not present in lattice or random network but often occurs in networks representing real systems, such as long tail in the distribution of degrees, high clustering coefficient, either assortative or dis-assortative connections between vertices, the presence of community structure, and a hierarchical organisation [62]. [61] proposed different measures to quantify the complexity of a complex network such as the relative number of non-isomorphic one-edge-deleted subgraphs, entropy measures to quantify the diversity of different topological features. Other measures of complexity of complex network includes Medium Articulation [63] and Offdiagonal complexity [64].
Complex networks can be found in most social, biological, and technological networks since their connection patterns are neither strictly regular nor strictly random. Scale-free [6] networks and small-world networks [65] are two well-known and representative class of complex networks. Structural characteristics of both topologies include power-law degree distributions for the scale-free case, short path lengths in both cases, and high clustering for the small-world case.

2.1.2.4 Line Graph

We can define a line graph $L(V_l, E_l)$ [50] of a graph $G(V, E)$ as:

- Each node u_l in the line graph L maps to an edge e in G.
- An edge $(u_l, v_l) \in E_l$ exists in line graph *iff* two edges e_u and e_v in G mapped to u_l and v_l respectively are incident in the graph G.

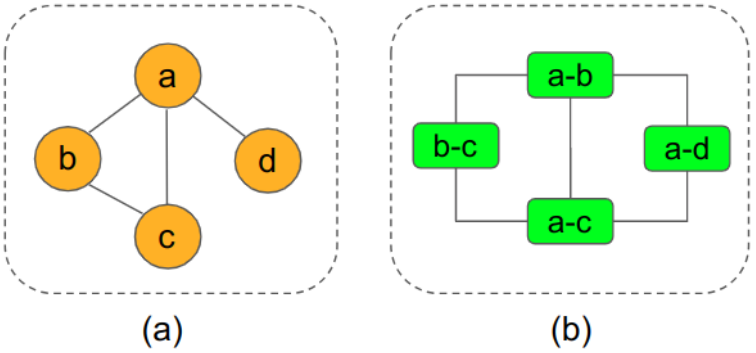


FIGURE 2.2: (a) Graph G with four nodes and four edges, (b) Line graph (L) of graph G.

An example of a line graph is shown in Fig. 2.2. The number of nodes in a line graph L is $|V_l| = |E|$ the number of edges in the graph G and the number of edges in L can be calculated as $|E_l| = \frac{1}{2}\sum_{i=1}^{|V|} deg_i^2 - |E|$, where deg_i is the degree of node $v_i \in V$. The linear time conversion of a graph G to its line graph L can be found in [66, 67].

2.1.3 Various graph properties

2.1.3.1 Clustering coefficient

How tight a community is can be found by a metric called clustering coefficient [68]. The clustering coefficient C_i for a vertex v_i is then given by the proportion of links between the vertices within its neighbourhood divided by the number of links that could possibly exist between them. Formally:

$$C_i = \frac{2\left|\left\{e_{jk} : v_j, v_k \in Nbd(i), e_{jk} \in E\right\}\right|}{k_i(k_i - 1)}$$

where k_i is the degree of node v_i, E is the set of edges, $Nbd(i)$ is the set of neighbourhood nodes of node v_i. Clustering coefficient is important due to several reasons: (i) It is a measure of the degree to which nodes in the graphs tend to cluster together. (ii) Empirically nodes with higher degree having a lower local clustering coefficient on average. (iii) It measures how influential a node is in a network.

2.1.3.2 Degree Centrality

The degree centrality [69] of a node v in a graph G is the fraction of nodes it is connected to, as given by the formula

$$centrality_{deg}(v) = \frac{deg_v}{(|V| - 1)}$$

where $centrality_{deg}(v)$ is the degree centrality of a vertex or node v and deg_v are the number of edges adjacent to the node v, with V as the set of all nodes in graph G. If a node has higher degree centrality, then the node is more central in the network. This aids in identifying the node that distributes information to a large number of nodes. Degree centrality is useful in many contexts. For instance, the degree of a node in a gene regulatory network makes it possible to evaluate the node's regulatory relevance right away. Highly interacting proteins with the majority of signalling proteins in signalling networks imply a central regulatory function, i.e., the regulatory hubs [70]. The degree of a node may also indicate a crucial involvement in transcription factors, or gene expression, depending on the type of protein [71]. While having a high degree is vital, there are situations in which low degree nodes end up being more significant than high degree nodes. Take into consideration, for instance, the case when a node of degree two links two sections of the network together.

2.1.3.3 Closeness Centrality

Closeness centrality [72] of a node v is the reciprocal of the average shortest path distance to v across all n-1 reachable nodes, given by the formula

$$centrality_{close}(v) = \frac{n-1}{\sum_{u=1}^{n-1} d(u,v)}$$

where $d(v,u)$ is the shortest-path distance between v and u, and $n-1$ is the number of nodes reachable from v. A node in a network has lower closeness centrality if it is closer to the center. The closeness centrality is commonly employed to assess the efficiency of information flow among nodes in a network, as well as to choose the most optimal starting point (node) within the network. Closeness is employed in several situations. In the study of bibliometrics, the concept of closeness has been employed

to examine how academics select journals and bibliographies in various disciplines, as well as to assess an author's influence on a particular subject and their social standing [73]. The notion of closeness centrality has been used to discover the metabolites that are particularly significant in a genome-based, large-scale metabolic network [74], contrast unicellular and multicellular animals [75] and obtain a deeper understanding of the development of metabolic organisation [76]. Closeness centrality suffers from the disconnected graph problem. In a disconnected graph, the distance between two vertices belonging to different components is usually set to infinity [77] and for such a graph, closeness centrality does not provide any information, as each vertex is assigned to $\frac{1}{\infty}$.

2.1.3.4 Betweenness Centrality

Betweenness Centrality [78] of node v is the sum of the fraction of all-pairs shortest paths that pass through v, given by the formula

$$centrality_{bet}(v) = \sum_{s,t \epsilon V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

where V is the set of nodes, $\sigma(s,t)$ is the number of shortest paths between nodes s and t and $\sigma(s,t|v)$ is the number of shortest paths between s and t passing through v. This centrality measures the extents to which a certain vertex lies on the shortest path of other vertex. Sometimes betweenness centrality is used to find the network hubs that can enhance the transmission efficiency of the data [79]. Betweenness centrality was first proposed in [77] to circumvent the problem of closeness centrality problem in the disconnected graphs as any pair of vertices s and t that does not have a shortest path connecting them will simply add 0 to the betweenness centrality of every other vertex. Betweenness centrality has several applications. In protein networks, the betweenness of a protein reveals its potential to facilitate communication among a variety of proteins [80]. In a telecommunications network, a node with higher betweenness centrality would exert greater influence over the network as it would serve as a critical pathway for the transmission of information [81].

2.1.3.5 Eigenvector Centrality

Eigenvector centrality [82] calculates a node's centrality based on the centrality of its neighbours. The eigenvector centrality for node v is the v^{th} element of the vector x, which is defined by the equation

$$Ax = \lambda x$$

where A is the adjacency matrix of graph G with eigenvalue λ. It is the extension of degree centrality and the measure of the transitive influence of a node in a graph. This centrality measure is based on a idea that a node is important if it is connected to other important nodes. In recent times, scholars from several disciplines have examined the uses, expressions, and expansions of eigenvector centrality in diverse areas: the eigenvector centrality of a neuron in a model neural network has been discovered to have a correlation with its relative firing rate in the field of neuroscience. [83]. A study conducted in the Philippines utilised data to demonstrate the disproportionate prevalence of political candidates' families with high eigenvector centrality in local intermarriage networks [84].

2.1.3.6 Page Rank

Page Rank [85] ranks nodes in graph G based on the structure of the incoming links. It assigns an importance score to each node. Important nodes have a high number of in-links from important pages. Page Rank is a link analysis algorithm and it assigns a numerical weighing to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. To define the Page Rank formally, we need to define its rank r_j of a node as:

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

where, d_i is the out degree of node i. As the matrix formulation, let M is a column stochastic matrix (column sum equals to 1) defined as $M_{ij} = \frac{1}{d_j}$; then the flow equation corresponding to the rank vector can be written as:

$$r = Mr$$

2.1.3.7 Global Efficiency

The efficiency [86] between two nodes u and v in a graph can be defined as

$$Eff(u,v) = \frac{1}{d(u,v)}$$

The global efficiency [87] is the average efficiency over all $u \neq v$, where, $u, v \in V$. Mathematically,

$$GE = \frac{1}{|E|} \sum_{u,v} Eff(u,v)$$

Global efficiency has become increasingly important in several practical applications, such as optimising transportation systems [88], brain connectivity [89] etc.

2.1.3.8 Local efficiency

The multiplicative inverse of the shortest path [90] between any two nodes in a network is the local efficiency [86] of those nodes. It is the global efficiency computed on the neighborhood of the node. Formally, the local efficiency of node $v \in V$ can be defined as:

$$LE(v) = \frac{1}{|\{e|e \in G_{Nbd(v)}\}|} \sum_{u,v \in Nbd(v)} d(u,v)$$

Where $Nbd(v)$ set of neighbourhoods of node v. is the The local efficiency measures the ability of a network to withstand failures at a small scale. The local efficiency of a node refers to its ability to facilitate information flow among its neighbours in the event of its removal.

2.1.3.9 Community

Finding community structure inside a network means characterizing the network structure at the mesoscopic level[91] - which means neither the nodes (microscopic level) nor the whole network (macroscopic level) but rather an intermediate level is considered. The problem of finding a community is difficult to formalize since the task can be done in several ways. So, an intuitive definition of a community can be defined as: A network is said to have community structure if the nodes in it can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected

internally. One can relate the community detection task to cluster analysis, which partitions a set of objects to identify homogeneous groups. In cluster analysis, each object is represented by a vector of attributes, and the grouping is done based on the comparison of those vectors. However, in community detection, the grouping is done based on the relationships (links) between the objects (nodes).Formally, a community in a graph $G(V, E)$ is a partition $P = \{V_1, V_2, ...V_p\}$ of the set of nodes V on which we get the optimum value of a particular objective function (e.g., modularity [32], conductance [92]). But finding such an optimum partition is an NP-hard problem. Several fast approximations are used to find such partitions. Popular community detection tools like Louvain[29], Infomap[30], Permanence[93] etc. are available.

Evaluating community structure How good the community detection algorithm is can be found by several metric calculations. Modularity [32] is one of those metrics that can be used to measure the strength of the network community structure. Formally, it can be defined as:

$$Q = \sum_{i=1}^{k} (e_{ii} - a_i^2)$$

where e_{ii} is the % of edges in community i and a_i is the % of edges with at least one end in community i. Another definition of modularity is:

$$Q = \frac{1}{4m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m})$$

where m is the number of edges in the graph, k_i is the degree of node i and A_{ij} is 1 when there exists an edge between node i and j else 0.

Comparing communities Comparison between communities can be done by several metrics, viz., Normalized Mutual Information (NMI) [94], Adjusted Rand Index (ARI) [95], Purity etc. In ARI, each object is considered individually. In Purity [96], a pair of objects are considered, and in NMI, information theory approach is considered.

2.1.3.10 Constant community

When the optimization problem (e.g., modularity [32], conductance [92]) is NP-hard, the order in which vertices are processed as well as the heuristics can change the results. But there are few vertices who never change their communities, irrespective of the changes in the orderings or heuristics. This invariant group of vertices forms a constant community [97]. The vertices that belong to the constant community are called constant vertices. Note that constant community is not a partition of nodes (as in the case of community), since a few non-constant nodes are not included in this structure.

2.1.4 Graph representations

The representation of the graph data is difficult and distinct from the representation of image and text data. Words in textual data are connected to one another in the form of a phrase, and within that sentence, they occupy predetermined positions. Pixels in image data are typically laid out on a grid space that is ordered, and this grid can be represented by a matrix. Graphs, on the other hand, have non-ordered nodes and edges that each have their own characteristics. A simple graph representation is the **adjacency matrix** A, where a cell $A[i][j]$ in the matrix denotes whether there is any edge exists between node i and node j or not. Formally,

$$A[i][j] = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

The problem of this representation is that for the large graph, its space complexity is high. Moreover, this simple representation can not capture the hidden but important features of the graph entities. Another way of representing the graph is mapping graph entities into vector space. A typical example is node embedding of a graph, where each node is mapped into a d-dimensional vector space. An extension of the node embedding could be edge embedding or the whole graph embedding.

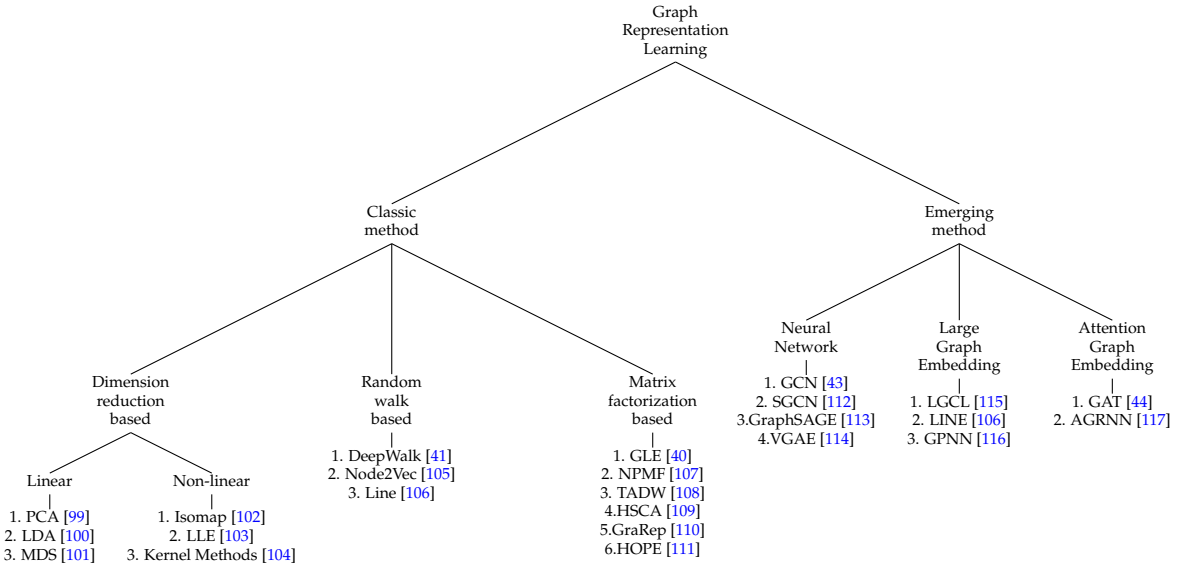


FIGURE 2.3: Various Graph Representation Learning methods

2.1.5 Graph Representation Learning

Graph representation allows the topological information of the interacting entities to be stored and accessed efficiently. A number of graph embedding techniques are adopted to convert the raw graph data into a vector representation while keeping its intrinsic graph properties. This process is called graph representation learning. After the graph representation is learned, it can be used with machine learning tools for many downstream tasks. The challenges of learning graph representation include: (i) A higher-dimensional representation tends to preserve more information than a lower-dimensional representation, but it is very costly in terms of storage and computation. (ii) As a graph contains a large number of attributes, picking the right one to embed can be problematic. According to [98], graph representation learning can be categorized into classic methods and emerging methods. Figure 2.3 shows the full categorization of graph representation learning.

2.1.6 Random Walks on network

On a graph, a random walk is a process that starts at one vertex and proceeds to a different vertex after every time step. When the graph is unweighted, the walk advances to a new vertex that is uniformly selected at random from its neighbours. If a graph

has weighted edges, it will migrate to a neighbour with a probability proportional to that edge's weight.

Formal overview. Let X be a random variable. $\{X_n\}$ denotes a sequence of random variables, where $n \geq 0$ is called the time step. This sequence of random variables is also called "discrete-time stochastic process" when these random variables take values from a set S, where S is called the state space and S could be $\mathbb{N}$, $\mathbb{Z}$ or in our case, the vertex set V. As $S = V$ is discrete, thus it is also called "a discrete-time discrete-space stochastic process. With this definition, a Markov process can be defined as a stochastic process X_n having a property:

$$P[X_{n+1} \in \mathcal{A} | X_n = x_k, k \leq n] = P[X_{n+1} \in \mathcal{A} | X_n = x_n]$$

, where $\mathcal{A}$ denotes the set of all events and $n \geq 0$. This Markov process can be analyzed by its one-step-transition probabilities:

$$p_{ij} = P[X_n = j | X_{n-1} = i]$$

Thus, a random walk in a network can be defined as a discrete-time stochastic process where the state space is V whose one-step-transition probability can be defined as:

$$p_{ij} = \begin{cases} \frac{1}{d_i}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

, where d_i is the degree of node i, which means that the walk jumps from a vertex to its adjacent vertex with equal probability. When the network is connected, one can represent the all-pair Markovian process in a matrix form:

$$P = D^{-1}A$$

where D is the degree matrix, which is defined as,

$$D_{ij} = \begin{cases} d_i, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

where d_i is the degree of a node i, and A is the adjacency matrix representing the network.

2.1.7 Graph Neural networks

GNNs are a new type of deep learning architecture that is built exclusively for graph-structured data. Unlike classic deep learning algorithms, which were designed primarily for text and image processing, GNNs are deliberately designed to process and analyze graph information. Figure 2.4 shows various graph based tasks performed by GNN. GNNs have become a potent tool for graph learning and have produced outstanding outcomes across a range of applications and sectors. The discovery of a novel antibiotic by a GNN model is among the most striking examples [118]. The model was evaluated on a library of 6,000 chemicals after being trained on 2,500 molecules. It stated that a substance known as Halicin [119] should be able to destroy a large number of germs that are resistant to antibiotics while posing little risk to human cells. This prediction led the researchers to treat mice afflicted with microorganisms resistant to antibiotics with halicin. They established its efficacy and held the model's potential for usage in the creation of fresh pharmaceuticals.

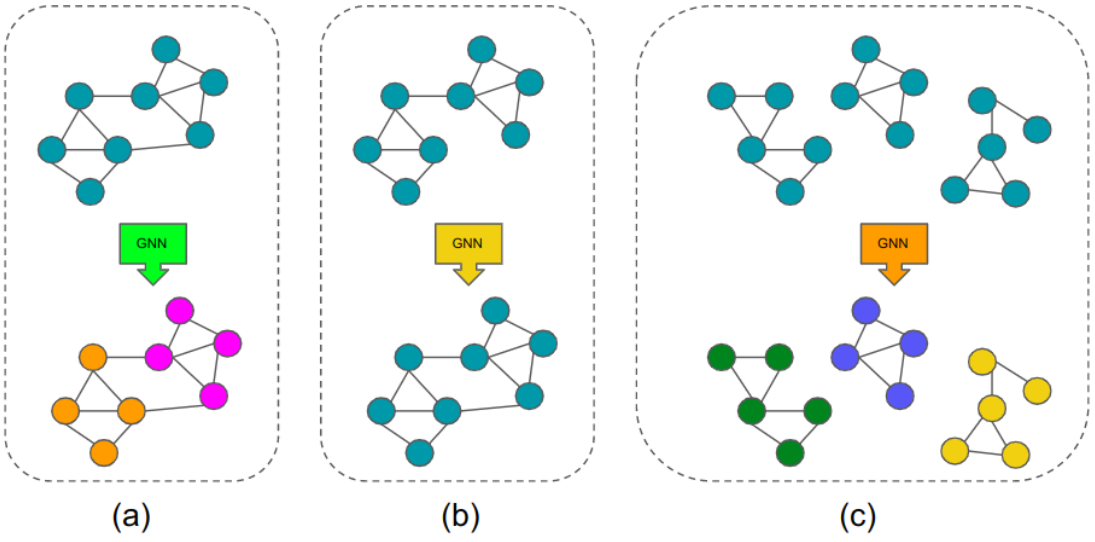


FIGURE 2.4: Application of GNNs. (a) GNN is used for the node classification task. (b) GNN is used for link prediction tasks. (c) GNN is used for graph classification tasks.

2.1.7.1 Types of GNN

In the literature, graph neural networks mostly come in three flavours: Recurrent Graph Neural Network [120], Spectral Convolutional Network [121] and Spatial Convolutional Network [122].

Recurrent Graph Neural Network. Most of these are early GNN works. RecGNNs use recurrent neural architectures to learn node representations. They presume that each node in a graph communicates with each of its neighbours continuously until a stable equilibrium is attained. Conceptually significant RecGNNs influenced later research on ConvGNNs. Let, x_u, x_v, x_{uv} denote the feature vector corresponding to nodes u, v and edge (u, v). Then, the node's current state is recurrently updated as follows:

$$h_u^t = \sum_{v \in Nbd(u)} f(x_u, x_{uv}, x_v, h_u^{t-1})$$

where f is a parametric function. Examples include GraphESN [123], GGNN [124], SSE [125] etc.

Spectral Convolutional Network. Graph signal processing uses spectral-based approaches because of their strong mathematical foundation. Here, the vector $x_i \in \mathbb{R}^n$ corresponding to a node i is considered a signal. The fourier transformation of the graph signal can be defined as:

$$\mathcal{F}(x) = U^T x$$

Where $U = [u_0, u_1, ..., u_{n-1}] \in \mathbb{R}^{n \times n}$ is the matrix of eigenvectors ordered by eigenvalues λ_i (spectrum). This matrix U can be found by factorizing the normalized laplacian matrix $L = U \Lambda U^T$, where Λ is the diagonal matrix of eigenvalues. Let g be a filter, thus, the graph convolution operation can be defined as:

$$x * g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g)) = U(U^T x \odot U^T g)$$

where $\odot$ is the element wise multiplication. The above equation can be simplified as:

$$x * g = U g_\theta U^T x$$

taking g_θ = diag($U^T g$), diag() is the diagonal matrix. Now if we consider g_θ as a set of learnable parameters $\Theta_{i,j}$ then the k^{th} hidden layer can be defined as:

$$H^k_{:,j} = \sigma(\sum_{i=1}^{f_{k-1}} U\Theta^k_{i,j} U^T H^{k-1}_{:,i}), \quad \text{j} = 1, 2, ..., f_k$$

where $H^0 = x$. Examples of Spectral Convolutional Network include GCN [43], ChebNet [126] etc.

Spatial Convolutional Network. Spatial-based approaches define graph convolutions based on a node's spatial relations, similar to the convolutional operation of a traditional CNN on an image. Images can be thought of as a particular type of graph, with each pixel acting as a node. Each pixel has a direct connection to the pixels around it. To create the updated representation for the central node, the spatial-based graph convolutions convolve the representation of the central node with the representations of its neighbours. Spatial-based ConvGNNs and RecGNNs both adhere to the same principles of information propagation. One example of Spatial Convolutional Network is NN4G [122], where the equation of the next layer node state can be given as:

$$H^k = f(XW^k + \sum_{i=1}^{k-1} AH^{k-1}\Theta^k)$$

Where f is an activation function, $X \in \mathbb{R}^{n \times d}$ is the node feature matrix, W and Θ are the learnable model parameters, and A is the adjacency matrix representing the graph G. Other examples include Diffusion CNN (DCNN) [127], Message Passing Neural Network (MPNN) [128] etc.

2.1.8 Thresholding algorithms used for edge classification in a network

In Chapter 3, we used an unsupervised algorithm where we classify the constant community edges (edges whose two terminals always belong to the same community irrespective of multiple runs) with the non-constant community edges using an image-based thresholding algorithm. We have used a variety of image-based thresholding approaches. A short description of each of the methods is provided below:

2.1.8.1 Bi-Level

The bi-level [129] method is a simple and fast image thresholding approach. If the background and foreground portions of a histogram are distributed equally, the histogram is completely balanced. In this method, the best threshold is determined by attempting to balance an uneven histogram. The background and the foreground are the two main classes that the input image is supposed to be separated into.

2.1.8.2 Otsu

Otsu's binary classification algorithm [130] is a form of finding a globally optimal k-means. This algorithm is used to segment/binarize grayscale images such that the object in the foreground can be distinguished from the background. The output of the algorithm is a single intensity threshold that separates pixels into two classes, foreground and background. This binary level thresholding algorithm [130] can be easily extended to multi-level thresholding [131]. As shown in Fig. 2.5, the single threshold based binarization may not always yield the best results. In this case, the flower (object) cannot be demarcated from the background by the conventional binary Otsu.

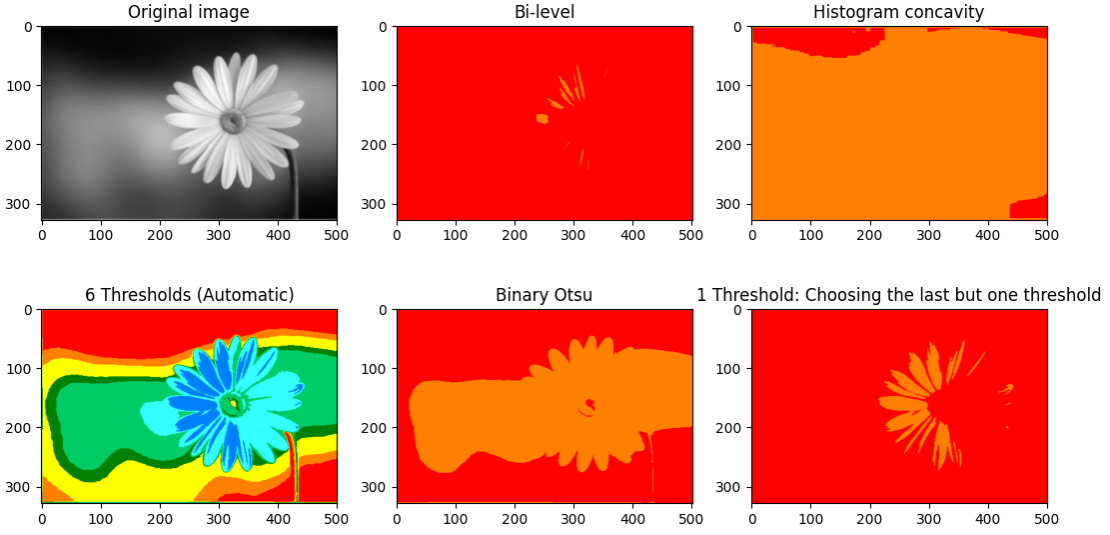


FIGURE 2.5: Several thresholding approaches for images that separate an image's foreground from background.

2.1.8.3 Histogram concavity

If an image can be easily divided into object and background in a histogram, the ranges can be easily distinguished. In this case, the threshold is the point at the valley's bottom, which is clearly visible. However, the object becomes difficult to distinguish from the background when the ranges in the histogram overlap. The best threshold would then be at the root of the shoulder (where the background pick and object peak overlap). It is not easy to locate the valleys and the shoulders. However, both of them are part of the histogram's concavities. The ideal threshold can be discovered by examining the concavity structure.

2.1.9 Fundamentals of brain networks: a brief anatomy of the human brain

The brain is a sophisticated organ that manages every bodily function as well as thought, emotion, memory, motor skills, touch, temperature, vision, respiration, and hunger. The central nervous system, or CNS, is made up of the spinal cord that emerges from the brain. The cerebrum, brainstem, and cerebellum can be thought of as the three main parts of the brain.

2.1.9.1 Cerebrum

It is the largest part of the human brain. It can be divided into two parts: the left and right hemispheres. Each hemisphere can be divided into four lobes: the frontal, parietal, temporal, and occipital lobes.

Frontal lobe. Of the four primary lobes, the frontal lobe is the biggest. It is covered by the frontal cortex. The frontal lobe is mainly responsible for decision-making, numeracy skills, and language.

Parietal lobe. The parietal lobe is located behind the frontal lobe and above the temporal lobe. The parietal lobe is crucial for manipulating things, understanding numbers and their relationships, and integrating sensory data from different sections of the body.

Temporal lobe. The second-largest lobe is the temporal lobe, which is located behind the ears. They are frequently linked to memory encoding and the processing of auditory information.

Occipital lobe. The majority of the anatomical region of the visual cortex is located in the occipital lobe, which is the brain's processing hub for visual information.

2.1.9.2 Cerebellum

Cerebellum is the small part of the human brain. Cerebellum contains more neurons than the rest of the brain because of the large number of granule cells.Although it takes up only 10% of the total brain volume. It involves motor movement regulation and balance control. It is attached to the underneath of the cerebral hemispheres. The flocculonodular lobe, the posterior lobe, and the anterior lobe make up the cerebellum. The vermis connects the anterior and posterior lobes in the centre. The anterior and posterior lobes of the cerebellum play a role in the coordination and smoothing of complicated motor motions, whereas the flocculonodular lobe is responsible for maintaining balance.

2.1.9.3 Brainstem

Brain stem is the stalk-like structure of the brain that connects the cerebrum to the spinal cord and cerebellum. The Brain stem is responsible for breathing, blood pressure, sleep, heart rate etc. The mid-brain, pons, and medulla make up the brainstem, which is located beneath the cerebrum.

Mid-brain (also known as the mesencephalon): This is the topmost and the smallest portion of the brain stem. It can be further subdivided into three parts: Tectum, Tegmentum and Ventrul tegmental area. Tectum forms the ceiling, tegmentum forms the floor of the mid-brain. and ventrul tegmental area is composed of paired cerebral penduncles. Mid-brain is responsible for eye movement, auditory and visual processing.

Pons: Pons lies in between mid brain and medula oblongata. It coordinates the activities within the cerebral hemisphere. It influences the sleep cycle, manages breathing, pain signals etc.

Medulla oblongata: It is located on the lower half of the brain stem. It deals with heart rate, blood presure and breathing.

2.1.10 Neurological applications of brain network analysis: Attention Deficit Hyperactivity Disorder and Alzheimer's disease

The human brain is a complex biological system. It is responsible for everything that we do, feel, and perceive. Thus, it is very important to know the brain diseases that affect regular life. Brain diseases can be of many types, ranging from injuries and infections to brain tumours and dementia. In this section, we shall discuss two such diseases: Alzheimer's Disease (AD) and Attention Deficit Hyperactivity Disorder(ADHD). AD is a neurodegenerative disorder [132] whereas ADHD is a neurodevelopmental disorder [47].

Neurodegenerative disorder is a condition in which the central nervous system's cells degenerate or stop functioning. In most cases, there is no therapy for neurodegenerative diseases, which often worsen with time. They could develop due to a tumour or stroke, or they might be inherited. A significant alcohol intake, exposure to some viruses, or ingestion of certain poisons are all risk factors for neurodegenerative diseases. Other examples include: Parkinson's disease [133], Huntington's disease [134].

Disorders that influence how the brain develops are known as neurodevelopmental disorders. The severity of the illnesses might range from minor ones that don't interfere with daily life to severe ones that require lifetime care. Other neurodevelopmental disorders include Autism [135], Schizophrenia [136] etc.

2.1.10.1 Alzheimer's disease

Alzheimer's disease (AD) [137] is a neurological condition that typically develops gradually and gets worse over time. 60–70% of cases of dementia progress to Alzheimer's disease. The disease is gradual, starting with mild memory loss and potentially progressing to the loss of communication and environmental awareness. The brain regions that are responsible for thought, memory, and language are affected by Alzheimer's disease. It can significantly impair a person's capacity to carry out daily tasks. It is unclear what causes Alzheimer's disease. There are numerous genetic and environmental risk factors connected to its development. The most potent genetic risk factor originates from an APOE [138] allele. A history of clinical depression, head trauma,

and high blood pressure [139] are additional risk factors. As per the report in [140], in 2020, around 50 million people worldwide suffered from Alzheimer's disease.

2.1.10.2 Attention Deficit Hyperactivity Disorder disease

One of the most prevalent neurodevelopmental diseases in children is Attention Deficit Hyperactivity Disorder (ADHD) [141]. It frequently persists into maturity and is typically first diagnosed in infancy. Children with ADHD may struggle to focus, manage impulsive behaviours (doing things without considering the consequences), or be extremely active. Many adults with ADHD experience antisocial, depressive, and anxiety disorders, just like their younger counterparts. Additionally, they have clinically significant impairments, such as histories of academic failure, work-related issues, and traffic accidents [142]. ADHD is not completely curable. If it is diagnosed early, then with some proper techniques such as medications [143, 144], meditation [145], virtual reality [146] etc., the inattentiveness, anxiety, and hyper-activity can be controlled, which helps them lead a smooth social life.

2.1.11 Brain Imaging Techniques

The fast development of brain imaging techniques over the past few decades, such as magnetic resonance imaging (MRI) [147] , electroencephalography (ECG) [148], magnetoencephalography (MEG) [149], electrocorticography (ECoG) [150], etc., has benefited the advancement of cognitive neuroscience. Out of these techniques, three widely used methods of magnetic resonance imaging include diffusion tensor imaging (DTI) [151], magnetic resonance imaging (MRI) [147] and functional MRI (fMRI) [152]. They have been used largely as a diagnostic tool as well as for research purposes.

2.1.11.1 MRI

The term "magnetic resonance imaging" [147] (MRI) refers to a method of medical imaging that creates detailed 3D-images of the organs and tissues within the body by employing a magnetic field and computer generated radio waves. In 1979, MRI was first used to image the human brain. In contrast to CT and PET scans, magnetic resonance imaging (MRI) does not require the use of X-rays or any other form of ionising

radiation and provides better contrast in images of soft tissues. Brain MRI scans can be used to segment the grey and white matter volumes. In clinical studies of brain illnesses, alterations in grey matter found in the MRI images could show morphological traits, such as the location and size of a specific lesion in the brain region. MRI has been extensively employed in the study of numerous neuropsychiatric illnesses, including Parkinson's disease [49], alzheimer's disease [153], schizophrenia [154] etc.

2.1.11.2 DTI

Diffusion tensor imaging [151], often known as DTI, is a type of magnetic resonance imaging (MRI) that estimates the axonal (white matter) architecture of the brain by using anisotropic diffusion [155]. Peter Basser was the one who initially presented the DTI method in 1994. It is an upgraded version of the traditional MRI, in which the signals are generated entirely from the motion of water molecules. The unpredictable and thermal movement of water molecules is denoted by the term "diffusion." In other words, diffusion tractography imaging (DTI) makes use of the diffusion of water as a tool to assess the anatomy of a brain network. This offers information on the static anatomy of the brain, which is an anatomy that is unaffected by the functioning of the brain.

2.1.11.3 fMRI

Functional magnetic resonance imaging [152], often known as functional MRI or fMRI, is a method for measuring brain activity that focuses on identifying changes linked with blood flow. The notion that cerebral blood flow and neuronal activation are connected is the foundation of this method. Blood flow to a certain region of the brain increases when that area of the brain is being actively used. The blood-oxygen-level dependent (BOLD) contrast was found by Japanese biophysicist and neuroscientist Seiji Ogawa [156] in 1990 and is used in the principal form of functional magnetic resonance imaging (fMRI). This is a specific form of brain and body scan that can be used to map neuronal activity in the brain of humans. This is accomplished by imaging the alteration in blood flow (hemodynamic response [157]) that is associated with the use of energy by brain cells.

2.1.12 Functional connectivity matrix generation using fMRI data

Generating time-series signal: fMRI is 4-D data. More specifically, it is a collection of 3-D images taken over a particular interval of time. If we consider a particular 3-D image, then each voxel has some particular intensity value representing the oxygen level at that point in the brain. Since the amount of oxygen changes over time, the intensity of each pixel representing it changes, and therefore, if we plot the changes in intensity value of a voxel with respect to time, we get a time-series signal. A time-series signal corresponding to a particular brain region is the mean signal of all the voxels that constitute the particular brain region. Fig 2.6 shows the generation of a single time-series from a particular brain region from a fMRI data.

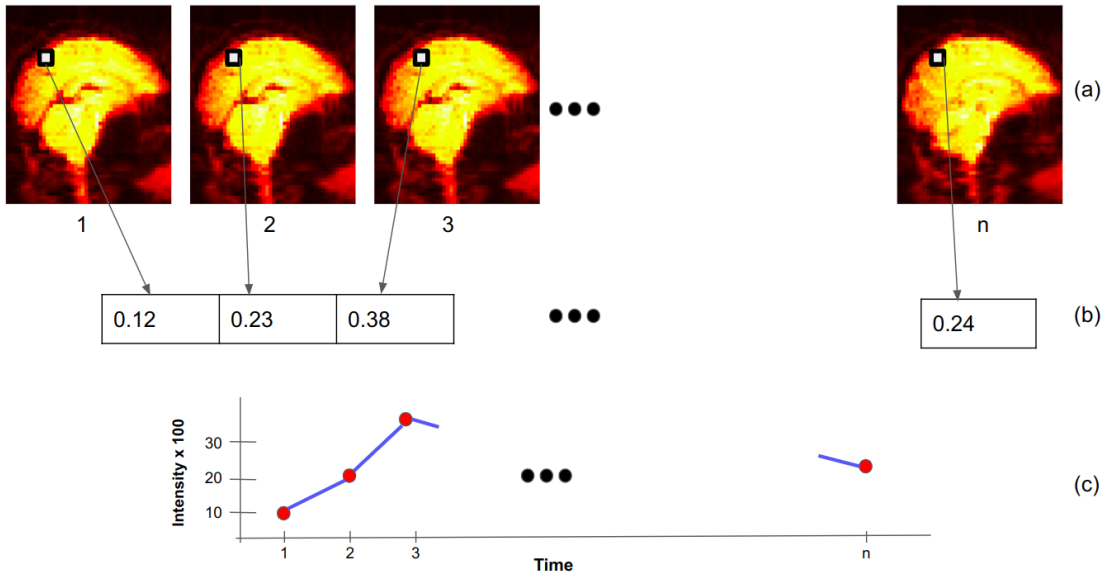


FIGURE 2.6: Steps for time-series generation: (a) 3-D images of fMRI data, in each time stamp, a region (a small square box) is selected to get the variation in mean intensity of the voxels in that region. (b) An array representing the mean intensity value of that region. (c) Plotting the array with respect to time (index starting from 1) indicates the time-series signal of that particular region.

Generating functional connectivity matrix: To obtain time-series signals from specific parts of the brain, the fMRI data needs to be masked with a standard brain atlas. Several brain atlases exist such as Harvard atlas [158], AAL atlas [159], MSDL atlas [160] etc. After obtaining time-series signals from various regions, a standard functional relationship (like Pearson correlation, partial correlation, covariance, etc.) is measured between every pair of regions, and a matrix representing their association is formed. This matrix is named the functional connectivity matrix. Later on, we will see how

this functional connectivity matrix can be used in various tasks such as the diagnosis of brain disease, the progression of brain disease, etc.

2.1.13 fMRI preprocessing

Since the downloaded fMRI data is in raw form, it needs to be treated in advance to remove various artefacts. Before using the fMRI data, it must go through some preprocessing steps, such as: (i) Remove the first few brighter images [161] (ii) head motion correction [162, 163], (iii) slice time correction [164, 165], (iv) distortion correction [166, 167], (v) registration [168], and (vi) spatial smoothing [169].

2.1.13.1 Removal of the first few brighter images

As a first step, the first few images are removed. Due to the magnetization effect, the first few images appear to be extremely brighter than the subsequent images. Thus, the volumes that are acquired for the first few seconds (typically 10s) are removed.

2.1.13.2 Head motion correction

Head movement has two main consequences. The location of succeeding images in the time series is first misaligned; this is referred to as **bulk motion** [170] since it entails a large-scale head movement. By realigning the images in the time series to a single reference image, typical motion correction techniques are intended to rectify this type of motion. Due to the fact that huge changes in picture intensity can occur when a voxel that at one point had no brain tissue suddenly contains tissue due to motion, bulk motion can have significant effects on activation maps, which typically occur near edges in the image. Second, head movement may cause the MRI signal to be corrupted. The protons from a nearby slice that migrate into a voxel as the head moves have an excitation that is different from what the scanner anticipates, and as a result, the reconstructed signal will not accurately reflect the tissue in the voxel. This phenomenon is referred to as the **spin history effect** [171]. If interleaved acquisition is employed, these effects can cause significant fluctuations in the intensity of a single slice or group of slices, which can be seen as stripes of alternating bright and dark slices. Standard motion correction

techniques cannot be used to correct this type of motion, but experimental techniques like ICA [172] or spin-history corrections [173] are used nowadays.

2.1.13.3 Slice timing correction

The fMRI data are gathered using two-dimensional MRI acquisition, which collects data one slice at a time. The slices are acquired in either ascending or descending order. Another technique, interleaved acquisition, involves sequentially acquiring every other slice, so that half of the slices (for example, the odd slices) are acquired before the other half (e.g., the even slices). Data in different areas of the image are regularly acquired at different times when using two-dimensional acquisition, and these discrepancies might be as much as several seconds (based on the pulse sequence's repetition time, or TR). It is difficult to analyse fMRI data when various voxels have varied acquisition times. A statistical model that simulates the anticipated signal that the task will evoke is built using the times of events (such as trials in a task). The data from each time point are then compared to this model, but because this analysis assumes that all of the data in the image were recorded simultaneously, there is a discrepancy between the model and the data, which varies across the brain.

2.1.13.4 Distortion correction

Gradient-echo echoplanar imaging (EPI), the most popular technique for fMRI acquisition, exhibits artefacts close to areas where air and tissue converge, such as the sinuses or ear canals. Dropout and geometric distortion are two of these effects, both of which are brought on by the air-tissue interfaces' inhomogeneity of the main magnetic field (also known as B0). Dropout is characterised by diminished signals in the orbitofrontal cortex and the lateral temporal lobe, two brain regions close to these air-tissue interfaces. It is best to use MRI acquisition techniques that minimise dropout since, after the data have been recorded, there is no way to retrieve data from a region with considerable dropout. fMRI scans can exhibit spatial distortion in the same locations in addition to signal loss. These inhomogeneities in the magnetic field lead to inaccuracies in the positioning of structures when gradients are used to encode spatial information in the MRI picture. With the aid of a field map [166], which describes the B0 field, the consequences of magnetic field inhomogeneity can be substantially mitigated.

2.1.13.5 Registration

Compared to MRI data, fMRI volumes have lower resolution and lower inter-tissue contrast. Thus, registration of fMRI data is difficult compared to MRI data. The registration process of fMRI data is a three-step process: first, it is registered into high-resolution MRI data (T1-weighted) of the same subject by rigid body transformation [174]. Then, the constructed T1-weighted MRI data is registered onto a standard space by affine transformation [175]. These two transformation matrices are then combined and applied to the fMRI data to register it in standard space.

2.1.13.6 Spatial smoothing

By applying a filter to the image, spatial smoothing [176] eliminates high-frequency information. Researchers apply spatial smoothing to the fMRI data for a variety of reasons. First, smoothing improves the signal-to-noise ratio for signals with greater spatial dimensions by removing high-frequency information or small-scale changes in the image. Second, it is recognised that there is spatial variability in the placement of functional zones when data from different people is pooled, and this spatial variability is not fixed by spatial normalisation. At the price of spatial resolution, spatial smoothing can lessen the disparity between individuals by fuzzing the data across space.

2.1.14 Preliminaries of Recommender system for network based study

Recommender systems (RS) are data-driven tools that use information processing to predict, prioritise, and identify users' preferences from a wide range of options. The main objective of recommender systems is to minimise the user's effort and time needed to get pertinent information on the internet. The ability of RSs to forecast a user's preferences and interests by examining their behaviour in order to generate specialised suggestions is its most important feature. The recommendation system was first applied to e-commerce sites to suggest products to customers and increase merchant revenue through greater sales [177]. The fundamental ideas of friendship and similarity served as the foundation for recommender system development. RSs typically have three components: user information, item information, and filtering techniques that use the user and item information to find products that match the users' interests. User information

includes things like what the user likes, what he or she has purchased, user ratings, and reviews. Item information includes things like how the items look and function. It takes both user and item information to create user-item interactions, which are the decisions the user makes when interacting with the system for a certain item. Participation could take the form of looking for a certain item or offering feedback on one. These activities are all recorded in the recommendation system's activity log. Finally, the RS filtering algorithms take into account user-item interactions in addition to user information and item information as input and offer a suggestion for a new item for the users to consider. The procedure is shown in Figure 2.7.

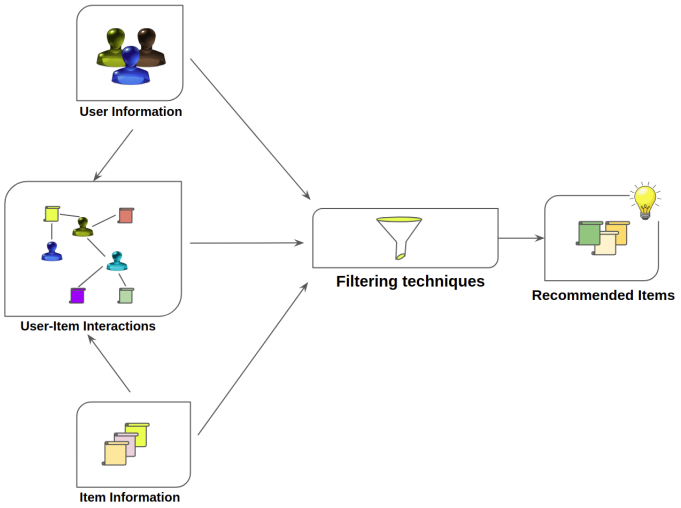


FIGURE 2.7: Basic Architecture of a Recommender System

2.1.14.1 Types of recommender system

The four main RSs approaches (Figure 2.8) are content-based filtering (CBF), collaborative filtering (CF), hybrid filtering (HF), and knowledge-based filtering (KBF).

Collaborative filtering (CF) The most popular method for creating recommender systems [178] is called collaborative filtering (CF). The typical CF recommendation approach analyzes past user-item rating data to locate new like-minded users to provide recommendations and suggestions. It predicts a user's behaviour based on their past behaviours, or, to put it another way, it establishes a connection between two or more comparable users based on their past behaviours in order to provide a recommendation [179]. "Neighbours" refers to these users who are similar to one another. The CF technique is used in almost every data field in recommender systems, despite the fact

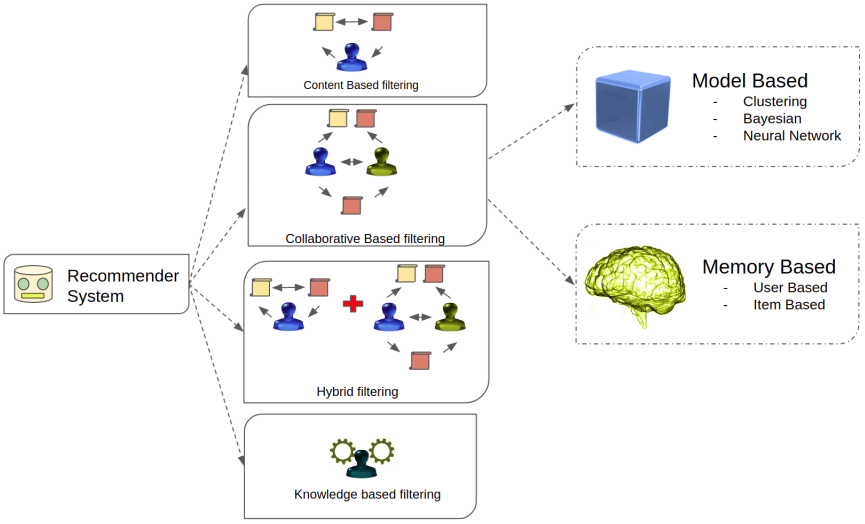


FIGURE 2.8: Major types of Recommender System

that it has a cold start issue [180, 181]. Collaborative filtering can be further divided into memory-based and model-based filtering.

Content-based filtering (CBF) The content-based filtering (CBF) recommender system is developed using a user's past behaviour. It makes the assumption that if someone previously preferred something, they will do so going forward. By comparing each of the unique features of the two items, one can determine how similar they are. The CBF method focuses mostly on these extracted features of the various items in an RS and delivers recommendations for an item to a user inside that RS based on these extracted features. When meaningful information cannot be extracted from data objects, CBF occasionally experiences limited content analysis [182] problems.

Hybrid filtering (HF) Collaborative filtering (CF) suffers from a cold start problem, whereas content-based filtering (CBF) suffers from a limited content analysis problem. Researchers have created hybrid systems [183] that blend CF and CBF techniques in order to overcome these problems. Combining CF and CBF in various ways can lead to the creation of a number of hybrid strategies [184]. The performance of recommender systems has improved thanks to the growing use of hybrid techniques. As a result, hybrid methods have been used in practically every recommendation-related industry, from e-commerce, hotels, and tourism to movies and novels [185, 186].

Knowledge-Based filtering: Knowledge-based (KB) filtering [187, 188] eliminates extraneous information by using background knowledge or data about users, objects, and

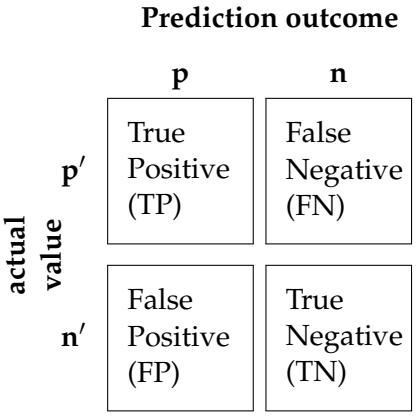


FIGURE 2.9: Confusion matrix. **p**: actual positive value, **n**: actual negative value, **p**′: predicted positive value, **n**′: predicted negative value,

their relationships. Some e-learning recommendation systems base their recommendations on their expertise in the subject. Knowledge-based filtering methods show how a certain product meets the requirements of a particular customer. To use the KB approach effectively, one must gain domain-specific knowledge about people and things. Relational information is used by KB systems in the context of e-learning to find learning resources that are pertinent to the users of the systems [189].

2.1.15 Some Evaluation techniques

Confusion matrix. The confusion matrix is a matrix that is used to assess the performance of classification models for a given set of data. It can be determined only if the true values of the data are known. The confusion matrix is used to evaluate the performance of a model. A visual representation of the confusion matrix is shown in FIGURE 2.9.

Precision. It can be described as the number of correct outcomes produced by the model, or how many of the positive outcomes predicted correctly by the model were actually true. It can be defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall. It is defined as the percentage of positive classes predicted correctly by a model out of a total of positive classes. It can be defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 score. It is the harmonic mean of precision and recall. It is difficult to compare two models that have low precision but high recall, or vice versa. So, we can use the F1 score for this purpose. This score allows us to assess both recall and precision simultaneously. If the recall equals the precision, the F1 score is maximised. It can be defined as:

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy. It is one of the crucial factors in figuring out how accurate a classification problem is. It specifies how frequently the model predicts the right result. The number of accurate predictions made by the classifier divided by the total number of predictions made by the classifiers can be used to compute it. It can be defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

Mean absolute error (MAE). It is broadly used in recommender systems. Let's say there are N items. If $\hat{y}_i$ denotes the predicted rating for an item i and y_i denotes the actual rating the item has, then MAE can be defined as:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}$$

Precision@k. In the domain of recommendation systems, the term "precision" can be defined as:

$$\text{Precision} = \frac{\text{Number of relevant items in the recommended items}}{\text{Total Number of recommended items}}$$

Recommended items can be divided into relevant recommended items and irrelevant recommended items. Now, in a recommender system, the rank of the items matters. Defining precision in such a way misses the consideration of rank. Thus, to include the

rank of the items, “precision@k” is considered a metric. Formally,

$$\text{Precision@k} = \frac{\text{Number of relevant items in the top k recommendation}}{\text{Number of items in top k recommendation}}$$

Average precision@k. Based on the definition of precision@k, the average precision@k can be defined as:

$$\text{Average precision@k (AP@k)} = \frac{1}{\text{Total number of relevant items}} \sum_{k=1}^{K} \text{Precision@k} * \text{Relevent(k)}$$

$$\text{Relevent(k)} = \begin{cases} 1, & \text{if item at } k^{th} \text{ rank is relevant} \\ 0, & \text{otherwise} \end{cases}$$

Mean average precision@k (MAP@k). Average precision@k is based on a single user only. When we have more than one user, the mean average precision@k is taken into account. Formally,

$$\text{MAP@k} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \text{AP@k}_i$$

Where, N is the total number of users.

2.2 Related works

2.2.1 Graph representation Learning

Presently, an enormous number of graph representation learning techniques are produced, and they are applied in different domains such as bio-informatics [190], chemistry [191], criminology [192], neuroscience [193] etc. Recent works on graph representational learning are mainly based on deep neural networks. Popular models include ChebyNet [126], GCN [43], GAT [44], APPNP [194], GPRGNN [195], etc. Zheng et al. [196] proposed “MathNet," which uses wavelet multiresolution analysis techniques for graph representation learning and takes different graph structures as input and assembles several consistent graph embeddings. Zhong et al. [197] uses temporal graph transformer as a model for graph representation learning in dynamic graphs. In [198],

the authors combined deep generative models with a graph auto encoder to exploit the uncertainty of hidden variables to produce better graph representations. A work based on unified representation learning considering both social networks and knowledge graphs is done in [199]. Most of the emerging GNN based graph representation learning techniques are based on semi-supervised methods. Some recent graph representation learning methods based on unsupervised methods include [200, 201, 202] etc. Some survey papers based on graph representation learning techniques can be found in [203, 204, 205].

2.2.2 Classification using Image Thresholding

The image thresholding algorithms are very commonly used for classification of images, such as in Mahdy et al.[206] where they use the image thresholding algorithms to classify the detection of COVID-19 patients. Iqbal et al. [207] presented a survey for the detection and classification of citrus plant diseases, and Amin et al. [208] presented a method to detect and classify tumours. Survey papers by Lamont et al. [209], Asokan et al. [210], and Chouhan et al. [211] present an overview of all the works related to image thresholding algorithms for classifying images. This paper is the first attempt to use this approach for graph related classification problems. We believe this approach will be extended to other classification problems in the future.

2.2.3 Graph Neural Networks

Graph Convolution Neural Network is categorised as a sub-class of techniques under the broader domain of Graph Neural Networks (GNNs). There are various types of GNNs based on their application domains. A graph neural network can be used in many fields, such as classifying the nodes [43, 212], link prediction [213, 214], graph classification [215, 216, 217], graph generation [218, 219, 220], community detection [221, 222] etc. A comprehensive survey on GNNs can be found here [223, 224]. Souravlas et al. [225] gave a brief overview of current state & advances in deep learning techniques for community detection. We have also seen many attempts to detect communities using GNN such as Bruna et al. [226], Shchur et al. [227], and Moradan et al. [228]. We didn't find any work that has attempted to use GNN for detecting constant communities.

In recent studies, GNNs have also been used with line graphs. For example, a supervised community detection task using a GNN model called a line graph neural network (LGNN) is proposed in [229]. LGNN uses both the graph G and its corresponding line graph $L(G)$ to find the communities in G. Using the LGNN, the authors in [230] study the link prediction task in the graph.

2.2.4 Constant or Consensus Community

Community detection is a well-studied problem, and numerous algorithms exist (see survey [231]). However, finding non-stochastic communities is a much less studied and more challenging problem.

Community detection algorithms are primarily based on optimizing objective functions. Due to underlying stochasticity, the resulting structures show considerable variations. Newman [232] suggests that stochasticity can be reduced by identifying "building blocks", i.e., groups of network nodes that are usually found together in the same community. In [233], a precursor to this work, the authors investigated the properties of constant communities with respect to within communities and across community edges.

A popular approach to finding stable communities is via consensus clustering, as introduced in [234]. Variations include multi-resolution consensus clustering [235], ensemble clustering [236, 237] and fast consensus [238] clustering and CHAMP [239]. Nevertheless, as seen here, these are not yet fast enough for large networks.

2.2.5 Random walks in a network

A random walk is the process of randomly going from one node to the next and forming a path. A random walk on a directed graph is equivalent to a Markov chain, while a random walk on an undirected graph is equivalent to a time-reversible Markov chain [240]. [241] studied large networks with random walks and came up with the precise expression for the mean first-passage time between two nodes. Variants of random walks include random walk with restart [242], PaRWalk [243], personalized PageRank [244], lazy random walk [245], etc. In [246], they used random walk to extend the original node2vec node-neighbourhood sampling method and produce a

second-order random walk sampling for heterogeneous multi-graphs. Multiple applications are built using random walk based algorithms including recommendation system [247, 248], link prediction [249], computer vision [245], network embedding [250, 105], semi-supervised learning [251, 252], complex social network analysis [253] etc. Several surveys based on random walk include [254, 255, 253].

2.2.6 Random walks based node embedding techniques

Node embedding technique is a popular graph representation learning (GRL) method [256]. Few popular random walk based node embedding techniques include DeepWalk [250], Large-scale Information Network Embedding (LINE) [106], Node2Vec [105], Asymmetric Proximity Preserving (APP) [257] etc. DeepWalk applies standard random-walk methods to generate node embeddings. Node2Vec uses biased random walks for better embeddings. APP uses rooted PageRank. By maximising an objective function, LINE maintains the integrity of both the global and local network architecture. *Huang et. al.* [258] broadly categorizes the random walk based node embedding process into two categories: Pointwise Mutual Information (PMI) and Auto-covariance. DeepWalk [250], LINE [106], NetMF [259], Node2Vec [105], WalkLets [260], and NetSMF [261] fall into the PMI group whereas, Multiscale [262] and the proposed method of (*Huang et. al.*) [258] falls in the auto-covariance group. Few surveys on node embedding techniques based on random walk method includes [263, 264, 265, 266].

2.2.7 Graph-based semi-supervised learning

Semi-supervised learning (SSL) [267, 268] and graph-based SSL (GSSL) [269, 270, 271] have become popular over past few years. Spreading labels from a small number of labelled data points to a larger number of unlabeled data is the topic of this study. Some GSSL [272] learned from both labelled and unlabeled data by taking advantage of pairwise correlations between the nodes in the min-cut algorithm. Some methods take into account the cluster assumption [273], which stipulates that the decision boundary shouldn't cross densely inhabited areas. The spectrum approach was employed by the [274, 275] for the semi-supervised learning challenge. [276] talked on the consistency of optimization-based methods for the GSSL under the presumption that the unlabeled data and labels with low noise levels are well clustered. Other works include

label propagation techniques using harmonic and Gaussian fields [270], transductive SVM [277], random walks [278, 279], combination of label propagation and bipartite graph construction [280] etc. *Fergus et al.* [281] provided a method for creating numerical approximations to the eigenvectors of normalised graph Laplacian by taking advantage of the convergence of the eigenvectors of the normalised graph Laplacian to the eigen-functions of weighted Laplace-Beltrami operators. Few GSSL approaches take into account both the graph structure, which includes regularisation tasks, and the feature vectors related to the data. *Zhang et. al.* [282], for instance, proposed using prototype vectors under the low-rank approximation and minimal information loss assumption to approximate the graph-based regularizer in GSSL. *Belkin et. al.* [283] suggested a few graph regularisation algorithms. The proposed techniques resolve a single sparse system of linear equations and are rather simple. *Zhou et. al.* [284] introduced GSSL by higher order regularisation, which is similar to a higher order Sobolev seminorm, where they employed Iterated Laplacian regularisation. *Li et.al.* [285] improved the performance of Laplacian Regularised Least Squares by using Nystrom subsampling and preconditioned conjugate gradient descent. *Belkin et. al.* [286] demonstrated a technique for data-dependent regularisation that makes use of the geometry of the probability distribution. Recent GSSL surveys include [287, 288].

2.2.8 Graph neural networks for node classification

Graph Neural Network (GNN) [223] is a tool for various graph-based semi-supervised downstream tasks such as node classification [289], edge prediction [290], network classification [215, 291], network embedding [292], and so on. One of the most important tasks in the network domain is node classification because it has many applications in various fields such as text classification [293], molecular fingerprints learning [294] neural machine translation [295], etc. Over-fitting and over-smoothing hinder the development of GNN for node categorization. Thus, the majority of current GNN research efforts are directed on solving these problems. For instance, in [296], the authors extended the training set using PaRWalk [243], which increased the GCN's accuracy. Another option is Jumping Knowledge Networks (JKNet) [297], where the neighbourhood features are aggregated differently for each node. Label smoothing [298] and knowledge distillation [299] are the sources of inspiration for Mutual Teaching for Graph Convolutional Networks (MT-GCN) [300]. The model trains label expansion

and teaches the peer network using both the temperature of the softmax layer and the ground truth labels. However, knowledge distillation systems frequently experience poor distillation due to insufficient utilisation of unlabeled data or too confident and biassed pseudo-labels. *Luo et. al.* [301] provides DualGraph, a framework for using unlabeled graphs for semi-supervised graph classification more effectively, as a solution to this issue. It was inspired by recent advancements in contrastive learning [302] and dual learning [303]. PageRank [85] and GCN [43] are used by Approximate Personalised Propagation of Neural Predictions (APPNP) [194] to create a better message passing scheme in the graph. The APPNP model, which suggested a generalised PageRank GNN, was the source of inspiration for the generalised PageRank Graph Neural Network (GPRGNN) [195], which addresses the issues of over-fitting and over-smoothing. DropEdge [304] randomly eliminates a predetermined number of edges from the input graph during each training epoch, acting as both a data augmentation and a message carrying reduction. The accuracy is increased by iterative deep graph learning (IDGL) [305], which simultaneously and incrementally learns graph structure and graph embedding. Deep Graph Infomax (DGI) [306] is a method that focuses on maximising the mutual information between patch representations and the associated high-level graph summaries, both of which are produced using a well-known architecture for a graph convolutional network. The Graph Harmonic Neural Network (GHNN) [307] is made up of two modules that look at graph topology data from different perspectives: a graph kernel network (GKN) [308] module and a graph convolutional network (GCN) module. In order to harmonise the training of the two modules and reconcile the consistency of their predictions, they created a novel harmonic contrastive loss and a harmonic consistency loss by emphasising high-quality unlabeled data during the training of the two modules.

2.2.9 Graph Theoretical Analysis in brain imaging

Graph theory offers many network modeling techniques to simulate the evolutionary processes of real-world complex networks [309, 310, 311]. Network modeling can infer the reasons governing interconnections and explain the mechanisms underlying the creation of a network [312, 19]. Previous research has shown that network

modeling can be effectively applied to study the network growth and dynamics observed in real-world complex networks [313, 314, 315]. With the help of suitable network models, one can quickly generate or simulate the desired network, which contains structural features similar to those explored in the real-world complex network [315, 316, 317]. Graph theory also helps us in identifying many important topological properties that can evaluate and analyze the performance of the complex human brain networks [318, 319, 320, 321]. A node, here, represents a brain region in such networks, and an edge corresponds to the relationship between two brain regions. Recent research also focuses on applying various deep learning approaches to study the cognitive aspects of the brain network [322, 323, 324]. People have explored the alterations of important topological properties of a network (transitivity, efficiency, degree distribution, modularity, etc.) during the formation of brain networks generated from different data sets such as functional MRI, structural MRI, Electroencephalography (EEG), etc [8, 325]. Thus, network modeling is considered a valuable technique that becomes important for understanding the mechanism of topological inter-connectivity inside a network. In particular, several node embedding techniques, in the deep learning framework, are becoming increasingly popular in representing the nodes in a brain network in a vector space for extracting important features or properties, e.g. in the present case for diagnosing AD patients [326, 327, 328].

2.2.10 Machine Learning Techniques in computational neurology

Recently, machine learning approaches are increasingly being used in differentiating between healthy and diseased situations by analyzing physiological patterns (biomarkers) [329, 330]. Recent studies by Lo et al.[331] have used diffusion tensor image tractography to construct connection matrices and graph matrices from DTI data of Alzheimer's patients. In [332], people have used fractional anisotropy (FA) values as input features in a machine learning approach to discriminate between Alzheimer's disease and healthy controls. Research in [333] suggests a probabilistic neural network model for classifying brain images using wavelet transformation. Convolution neural networks (CNN) is one of the modern deep learning techniques [326, 327] that has been successfully applied to classifying brain networks from magnetoencephalography (MEG) data [334] and attention deficit hyperactivity disorder ($ADHD$) data [335]. Recent research has developed several neural network-based graph embedding techniques

[41, 105, 336] that have been successfully applied in brain research. To better learn deep representations of graph-structured data, graph convolutional neural networks (GCNs) have recently evolved [337] and have been demonstrated to outperform other conventional relational learning approaches. An end-to-end graph similarity learning framework is proposed by Ma et al. [338, 339] to learn the brain network representations from multi-subject fMRI data using a supervised technique. To characterise and understand the community structure in brain networks, the proposed framework executes higher-order convolutions by inserting higher-order proximity into GCN. Morris et al. [340] proposed a sparse graph embedding technique for diagnosing Autism Spectrum Disorder (ASD). Hence, machine learning techniques have become popular in diagnosing several neurological disorders such as $ADHD$, AD, and ASD in the human brain.

2.2.11 Machine learning frameworks on diagnosing ADHD:

In general, the three main steps of a machine learning-based classification process are feature selection, feature extraction, and label selection via classifiers. The feature selection part includes LASSO [341], SVM-REF [342], FADR [343] etc. To achieve high discriminating accuracy in the diagnosis of ADHD, a Support Vector Machine-Recursive Feature Elimination (SVM-RFE) classifier was used with a different number of feature sets. In LASSO, the functional data were first converted into wavelet data, changing the problem of data fitting into a problem of variable selection. A discriminative subset of functional-anatomical brain regions was chosen for Functional-Anatomical Discriminative Regions (FADR), which aims to identify anomalies in functional connectivity in mental diseases. In feature extraction a set of features are extracted from the data. LDA [100], ICA [343], fusion fMRI method [344], subspace projection algorithm [345], some graph based methods [346, 347] etc are used as the feature extraction methods. Lastly, several classifiers such as SVM [346, 348], random forest, decision tree [349], adaptive boosting decision trees (AdaDT) [350], KNN [351] are used as the classification step. In [352], the authors fused imaging data with non-imaging data and apply SVM to diagnose ADHD. They did not use any atlas, rather used Affinity Propagation clustering algorithm [353] to find various brain regions. *Kautzky et. al.* [354] used PET and MRI scans data along with genotypic information and apply SVM for the classification of ADHD with the healthy control.

In [346], Based on their inter-graph distance measurements, all the subjects are projected from an unknown graph-space to a low dimensional space using the Multi-Dimensional Scaling (MDS) technique. The primary focus in [348] is to used machine learning to assess the relative predictive value of teacher and parent ratings, behavioural measures of executive function (EF), and brain measures of EF in predicting ADHD in a sample. In [351], Electroencephalogram (EEG) data were divided using empirical mode decomposition (EMD) and discrete wavelet transform (DWT) techniques.

2.2.12 Deep learning frameworks on diagnosing ADHD:

The deep learning based framework removes the boundary among the above three steps of the machine learning based framework. Several works include fully convolutional neural network (FCNet) [355] - where the model extracts functional connectivity directly from raw fMRI time-series signals, Separated channel attention convolutional neural network (SC-CNN-attention) [356] - where they presented a new two-stage network structure by combining a separated channel convolutional neural network (SC-CNN) with an attention-based network (SC-CNN-attention) in order to distinguish between ADHD and healthy controls on a large-scale multi-site database, 3-D-CNN [357] - where through the use of MRI scans and 3-D convolutional neural networks (CNNs), they created a deep learning-based ADHD classification system. They [357] also created a method that extracts useful 3-D low-level characteristics from functional MRI (fMRI) and structural MRI (sMRI) data in order to decrease the vast number of parameters used by neural networks. In [358], the author created a 4-D CNN that uses granular computing and can calculate granularity at a coarse level by stacking layers. The CNN was trained using derivative changes in entropy. In [359], using the Event-Related Spectral EEG and CNN, the authors discriminated adult ADHD from healthy individuals. The combination of EEG and CNN is also used in [360] for the diagnosis of ADHD. In [361], spectral features of EEG signals is used with the LSTM for the ADHD classification tasks. In [362], the authors built 4-D CNN based on granular computing that was trained on derivative changes in entropy and can calculate granularity at a coarse level by stacking layers. In [363], a short-time diagnostic approach was proposed which can swiftly evaluate the patients' fMRI data and help clinicians in remote diagnosis. In [364], the ADHD-200 dataset has been used to generate 2-dimensional

images. A convolutional neural network (CNN) algorithm along with long short-term memory (LSTM) have been employed. They have used these image datasets to classify typically developing controls. Other works include [335, 365, 366] etc.

2.2.13 Machine learning frameworks on diagnosing Alzheimer's disease

A degenerative condition called Alzheimer's disease (AD) causes progressive, irreversible cognitive decline. Numerous methods based on machine learning with neuroimaging images have been suggested to obtain a precise and timely diagnosis and identify AD at early stages. In [367], Franciotti et al. applied three machine learning techniques: random forest, gradient boosting, and extreme gradient Boosting algorithms for diagnosing the Alzheimer's disease. They also combine clinical and biological measures to improve the diagnosing accuracy. Kumar et al. [368] proposed MRI based hybrid machine learning technique (SVM and CNN) to diagnose Alzheimer's disease. In [369], the authors proposed a multi-class diagnosis of Alzheimer's disease using several machine learning techniques such as: SVM, random forest, LDA, KNN etc. Gao et al. [370] suggested a combination of SHAP (SHapley Additive exPlanations) [371] and XGBoost to diagnose Alzheimer's disease and to explain the progression of this disease. In [372], the authors integrate CNN with KNN with baysian optimization to diagnose Alzheimer's disease. Neffati et al. [373] uses Downsized Kernel Principal Component Analysis (DKPCA) and multiclass Support Vector Machine (SVM) as their ML tools for AD classification.

2.2.14 Deep learning frameworks on diagnosing Alzheimer's disease

In particular in the field of computer vision, deep learning, a cutting-edge machine learning approach, has demonstrated outstanding results over conventional machine learning methods for recognizing complicated patterns in complex high-dimensional data. Recent years have seen a significant increase in interest in the use of deep learning for automated classification and early detection of Alzheimer's disease (AD), thanks to the quick development of neuroimaging methods and the resulting production of vast amounts of multimodal neuroimaging data. Several deep learning models include RBM [374], DBM [375], DBN [376], CNN [377] etc are used to diagnose Alzheimer's

disease. In [378], the author uses various CNN based models: DenseNet121, VGG 16, ResNet50, EfficientNetB7, and InceptionV3 for diagnosing Alzheimer's disease. Chiyu et al. [322] used FSBi-LSTM for the Alzheimer's classification. Islam et al. [379] proposed a deep CNN-based pipeline to identify Alzheimer's disease along with its various stages. Sarraf et al. [380] proposed Optimized Vision Transformer (OViTAD) to Predict Various Stages of Alzheimer's Disease. A recent survey on deep learning based diagnosis and prognosis of Alzheimer's disease can be found in [381].

2.2.15 Network modeling in brain network simulation

It is important to note that network modelling has recently undergone a number of noteworthy advancements, most notably in the field of research pertaining to the simulation of brain networks. These advancements have been made possible by the intersection of graph theory and network neuroscience [382, 383, 384, 385]. In [382], the authors proposed Economical Clustering Model (ECM). The authors attempted to discover the connection mechanism behind the construction of brain networks while ECM was being used to build the brain network by adopting the local topologies of common neighbours (CN) between two areas. In a similar manner, [383] put out a number of generative models of human brain networks that were organised according to various network architectures. They tried to gain a better understanding of the wiring rules that determine the topologies of the brain network. In [384], they outlined the core aims of building network models; then they went over the most popular types of network models, which can be broadly classified along the following three main lines: from data representations to first-principles theory; from biophysical realism to functional phenomenology; and from elementary descriptions to coarse-grained approximations, and at last, they established validation rules for these models using concepts from biology, philosophy, and other fields. In [385], they presented an overview of the main results of resting-state activity across a variety of neuroimaging modalities, including fMRI, EEG, and MEG. The authors outlined the best ways to categorise and examine anatomical and functional brain networks and how disrupting the balance of these networks may result in mental health issues.

Part - I

Developing algorithms to study the invariant nodes inside the community using Graph Representations technique

Publications:

- Anjan Chowdhury, Sriram Srinivasan, Sanjukta Bhowmick, Animesh Mukherjee, and Kuntal Ghosh. "Constant community identification in million scale networks using image thresholding algorithms." In Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'21), pp. 116-120. https://doi.org/10.1007/s13278-022-00895-8, 2021.
- Anjan Chowdhury, Sriram Srinivasan, Sanjukta Bhowmick, Animesh Mukherjee, and Kuntal Ghosh. "Constant community identification in million-scale networks." Social Network Analysis and Mining 12, no. 1: 70, https://doi.org/10.1007/s13278-022-00895-8, 2022
- Swarup Chattopadhyay, Anjan Chowdhury, and Kuntal Ghosh. "Application of Machine-Learning Techniques in the Development of Neighbourhood-Based Robust Recommender Systems." In Recommender Systems, pp. 203-233. CRC Press, https://doi.org/10.1201/9781003319122-13, 2023.

Chapter 3

Detection of Constant Communities in a Complex Network

Chapter summary: The detection of community in a network is a challenging task due to its "NP-hard" nature. Hence, most of the community detection algorithms are based on optimizing some objective function, and therefore their results are stochastic in nature. Since community detection is extremely important in a network due to its diverse applications, changes to the community structure on every execution creates obstacle to obtain accurate results. In this chapter, we are going to discuss how we solve the problem by developing some algorithms using some hand-made and automated feature engineering (graph representations) on the network.

3.1 Introduction

One of the most important core operations in large-scale real-world networks is the detection of communities. However, this operation is likewise stochastic by nature. The algorithm, the settings, and even the order in which the vertices are processed can all affect the communities that are found. Finding continuous communities is one way to lessen the impact of these algorithmic artefacts. Constant communities are a collection of vertices that consistently belong to the same community and display stable partitioning (FIGURE 3.1).

The current methods for finding constant communities entail executing a community identification algorithm numerous times or a variety of them, merging the results, and then identifying the set of vertices that are consistently clustered together. These methods do not scale to large networks, however, and are quite expensive in terms of time and memory.

In this chapter, we provide binary edge classification as a technique to recognise constant communities. A network's edges are categorised according to whether or not they belong to a stable community. Our approaches are substantially faster than the currently available methods and may be used with million-scale networks because the categorization just depends on easily calculable edge attributes rather than on locating communities.

In this research, the **semi-supervised** and **unsupervised** methods for obtaining constant community are both taken into consideration. In the unsupervised approach, we manually find the features of the edges, build histograms based on these features, and apply some thresholding algorithms to classify the constant community edges. We have used various histogram-based image thresholding algorithms like Bi-Level [129], Histogram Concavity [386], Otsu's [130] algorithm and its variants in our study. In the semi-supervised approach, we automated the process of finding edge features using GCN [43] and Line Graph [50].

Although the semi-supervised approach performs better than the baselines on real-world networks, obtaining a set of known labelled nodes for training purposes can be challenging at times, and for large networks, it takes more time to execute than the unsupervised approach. Our unsupervised algorithms, on the other hand, can handle networks with millions of vertices with good accuracy.

The study on noisy environments has also been incorporated into the current work, and practical evidence has shown that our strategy still outperforms the baselines.

Lastly, we have done a case study where we have shown that application of community and constant community gives better outcomes in the neighbourhood based recommendation system [387]. The constant community we have used in this recommendation system are found using both of our unsupervised and semi-supervised methods.

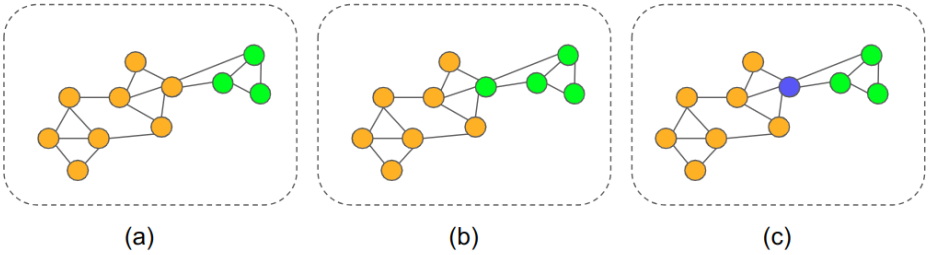


FIGURE 3.1: Example of formation of constant communities. (a) Output of first algorithm (b) Output of second algorithm (c) Group of nodes always staying together are called constant community. Different colors (orange and green) represent different communities to which vertices are assigned for each algorithm. blue node is not the part of constant community.

3.2 Proposed strategies

3.2.1 Semi-supervised approach to detecting constant communities

In a graph-based semi-supervised learning scenario, the aim is to predict the labels of other nodes using the label information of a small number of available nodes. GCN is a well-liked deep learning method that employs graph-based semi-supervised learning methods. The main function of GCN is to categorise the nodes in a graph. In our research, we use the GCN to perform edge classification and identify constant communities. We transform the input graph into its equivalent line graph so that the GCN can classify the edges. This method allows for the representation of every edge in the input graph G by a distinct node in the associated line graph L. The categorization of edges in the input graph follows logically from the classification of nodes in the line graph. The key steps of the semi-supervised approach are given below:

Step1: Generation of the line graph L. The GCN is a node classifier by default, as was already mentioned. As the node of the line graph represents an edge of the input graph, we must transform the input graph G to its corresponding line graph L in order for it to function as an edge classifier. Thus, classifying the nodes in L is equivalent to classifying the edges in G.

Step2: Generating features from graph G. GCN requires that each node be associated with a feature vector so that it can apply the smoothing operation to the feature vectors for the node classification task. A feature vector is composed of a number of features.

In this step, we have discussed how we construct the feature vector for each node in the line graph.

Before constructing the features, we numbered all the edges in G with a unique integer in $\{0, 1, ..., |E|\}$ and all the nodes in G with a unique integer in $\{0, 1, ..., |V|\}$. We then calculate four feature vectors $\mathcal{F}^e_{EI}$, $\mathcal{F}^e_{EU}$, $\mathcal{F}^e_{NI}$ and $\mathcal{F}^e_{NU}$ for each edge e in G.

(i) $\mathcal{F}^e_{EI}$: For each edge e in G, we take a vector of size $|E|$ and initialize all its cells to zero. Let EI_e represent the set of all edges collected by taking the intersection of the neighbourhood edges of the two terminals of e. We then assign 1 to a cell i if the i^{th} edge belongs to EI_e. Formally,

$$\mathcal{F}^e_{EI}(i) = \begin{cases} 1 & \text{if } i \in EI_e \\ 0 & \text{Otherwise} \end{cases} \tag{3.2.1}$$

(ii) $\mathcal{F}^e_{EU}$: Same as before, we again consider the edges in the neighbourhoods of an edge e but this time taking the union of the neighbourhood edges of the two terminals of e. Let EU_e represent this set of neighbourhood edges of e. Now $\mathcal{F}^e_{EU}$ can be constructed by first taking a vector of size $|E|$ and initialize all its cells to zero. Then, assigning 1 to a cell i if the i^{th} edge belongs to EU_e. Formally,

$$\mathcal{F}^e_{EU}(i) = \begin{cases} 1 & \text{if } i \in EU_e \\ 0 & \text{Otherwise} \end{cases} \tag{3.2.2}$$

(iii) $\mathcal{F}^e_{NI}$: For edge $e \in G$, $\mathcal{F}^e_{NI}$ be a vector of size $|V|$ and initialize all its cells to zero. Let $NI_e = \{v \in V | v \in Nbd(e_0) \cap Nbd(e_1)\}$, e_0 and e_1 are two terminals of edge e, $Nbd(x)$ is the set of neighbours of node x. Then we assign 1 and 0 to $\mathcal{F}^e_{NI}$ based on the following rules:

$$\mathcal{F}^e_{NI}(i) = \begin{cases} 1 & \text{if } i \in NI_e \\ 0 & \text{Otherwise} \end{cases} \tag{3.2.3}$$

(iv) $\mathcal{F}^e_{NU}$: For each edge e in G, we take a vector of size $|V|$ and initialize all its cells to zero. Let NU_e represent the set of all nodes collected by taking the union of the

neighbourhood nodes of the two terminals of e. We then assign 1 to a cell i if the i^{th} edge belongs to NU_e. Formally,

$$\mathcal{F}^e_{NU}(i) = \begin{cases} 1 & \text{if } i \in NU_e \\ 0 & \text{Otherwise} \end{cases} \tag{3.2.4}$$

Step3: Concatenation of the feature vectors. After obtaining the feature vectors for each edge e in G, we horizontally concatenated them and made a single vector $F(e)$ of size $2|E| + 2|V|$.

Step4: Reducing the dimension of $F(e)$. The size of the concatenated feature vector $F(e)$ is $2|E| + 2|V|$, and thus it takes a huge amount of memory for the large graphs. It also takes enormous time to train a GCN with such large feature vectors. Therefore, we reduce the dimension of the feature vector to 20 using the Principle Component Analysis (PCA) [99] tool. Let $F_r(e)$ denote the final reduced feature vector of an edge e in G. Since an edge e in G is converted to a node u^l_e in a line graph L, we can write the feature vector $F_r(e)$ as $F_r(u^l_e)$ denoting the feature vector corresponding to the node u^l_e in L.

Step5: Training node selection. To train a GCN, a set of nodes with known labels is required. Since we are applying GCN to the line graph L, we need to select the seed nodes from L. To do this, we again take help from the input graph G. First, we applied a particular community detection algorithm a few times $(2-3)$ over the input graph G, then from there selected two sets of edges: (i) E_C: set of edges that never change their community; (ii) E_{NC}: set of edges that change their community. We then create the node sets $V_{LC} \subset V_L$ and $V_{LNC} \subset V_L$ of known labels in L. V_{LC} is constructed by labelling the nodes in L that correspond to the edges in E_C to 1 and V_{LNC} is constructed by labelling the nodes in L that correspond to the edges in E_{NC} to 0. Thus, the set of training nodes in L will be: $V_{LT} = V_{LC} \cup V_{LNC}$, where, $V_{LC} \cap V_{LNC} = \Phi$.

Step6: Applying GCN. Finally, the GCN is applied to the line graph with the set of labelled training nodes V_{LT} and the set of generated feature vectors F_r. The node set V_L is divided into two classes by GCN: V_{L1} and V_{L0}. The set of constant community

edges is represented by the edges in G representing the nodes in V_{L1}, while the set of non-constant community edges is represented by the edges in G representing the nodes in V_{L0}.

The workflow diagram regarding the steps of the semi-supervised approach is given in Fig. 3.2:

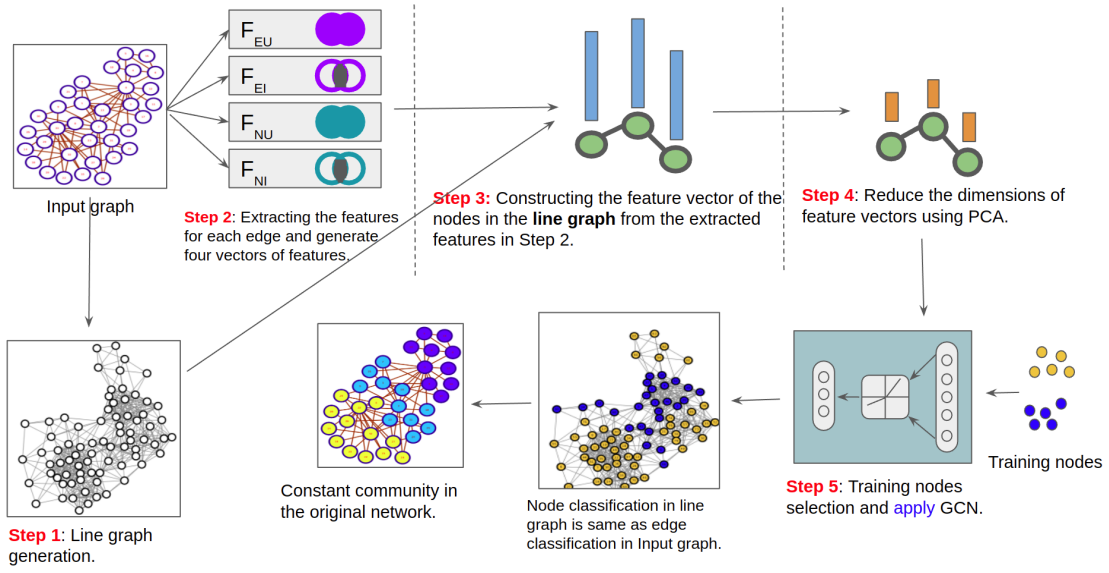


FIGURE 3.2: **Steps to detect the constant community using semi-supervised workflow.** In Step 1, the line graph is generated from the input graph. In Step 2, features are extracted from the original graph. In Step 3, the feature vector for each node in the line graph is constructed using the features from Step 2. In Step 4, the dimensions of the feature vectors are reduced using PCA. In Step 5, with the help of the selected training nodes, GCN is applied to the line graph to classify the nodes. Finally, the edges of the input graph are classified using the classified nodes in the line graph to find the constant community.

3.2.2 Un-supervised approach to detecting constant communities

The semi-supervised technique has significant shortcomings, even though it works well:

- It requires a collection of training nodes (nodes whose labels are known) for its training. But finding a sufficient number of training nodes can be challenging.
- It also takes time and memory to generate a line graph.
- Training a large line graph is also time-consuming.

We experimented with a novel, unsupervised method to address these issues. With the help of the neighbours of each edge's two terminals, we identify four local properties of each edge in this method, and for each of these properties, a histogram is produced. Then, we identify the ideal thresholds for each histogram using a well-known image thresholding algorithm. Finally, we use these thresholds into our novel algorithms to filter out the non-constant community edges in the graph and produce the constant communities (FIGURE 3.4). The key steps of our unsupervised approach are given below:

Step1: Extracting the edge features. Let $e = (u, v) \in E$ be an edge and u and v be its terminal nodes. Also, let $N(u)$ signify the collection of neighbours for a vertex u and $\Delta(u)$ denote the set of triangles that contain vertex u. Also, for a vertex set Y, let $\mathcal{D}(Y)$ signify the density of the subgraph that is generated by Y, which is the proportion of the number of edges in the subgraph to its entire number of possible edges. The computed four features (FIGURE 3.3) can be described as:

- Density induced by *the common neighbors of the edge terminals* (D_{both}).
  Formally, $D_{both}(u, v) = \mathcal{D}(N(v) \cap N(u))$.
- Density induced by *all the neighbors of the edge terminals including the terminals* (D_{any}).
  Formally, $D_{any}(u, v) = \mathcal{D}(N(v) \cup N(u))$.
- The ratio of the number of triangles containing both u and v to the number of triangles containing at least one of u or v (D_{tri}).
  Formally, $D_{tri}(u, v) = \frac{|\Delta(u) \cap \Delta(v)|}{|\Delta(u) \cup \Delta(v)|}$.
- The proportion of the number of common neighbors and the number of all neighbors of the terminals (termed as the Jaccard Index (JI)).
  Formally, $JI(u, v) = \frac{|N(v) \cap N(u)|}{|N(v) \cup N(u)|}$.

Step2: Generating histograms. For each feature, a histogram is produced. The x-axis of the histogram corresponds to the feature values, and the y-axis denotes their frequencies. The histogram is named as: $\mathcal{H}_{both}, \mathcal{H}_{any}, \mathcal{H}_{tri}, \mathcal{H}_{JI}$ corresponding to $D_{both}, D_{any}, D_{tri}$,

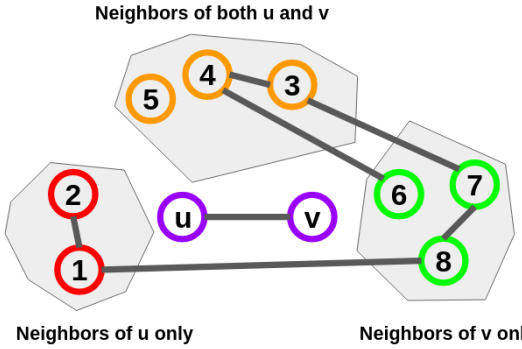


FIGURE 3.3: Features used for edge classification. For an edge (u, v); the red vertices are connected to u only, the green vertices are connected to v only, and the orange vertices are connected to u and v both. For ease of visualization, the connections to u and v are not shown.

$$D_{both}(u,v) = \text{density of } (3,4,5) = \tfrac{1}{3}$$
$$D_{any}(u,v) = \text{density of } (1,2,3,4,5,6,7,8,u,v) = \tfrac{18}{45}$$
$$D_{tri}(u,v) = |\{(u,v,3),(u,v,4),(u,v,5)\}|div(|)\{(u,v,3),(u,v,4),(u,v,5),(u,1,2),(u,3,4),(v,7,8),(v,4,6),(v,3,7),(v,3,4)\}| = \tfrac{1}{3}$$
$$JI(u,v) = |\{3,4,5\}|div(|)\{1,2,3,4,5,6,7,8,u,v\}| = \tfrac{3}{10}.$$

.

and JI respectively.

Step3: Obtaining a threshold. In this step, we have applied an image thresholding algorithm to the histograms to obtain the thresholds from the histograms. We named the thresholds $\mathcal{T}_{any}, \mathcal{T}_{both}, \mathcal{T}_{tri}$ and $\mathcal{T}_{JI}$ correspond to $\mathcal{H}_{any}, \mathcal{H}_{both}, \mathcal{H}_{tri}$ and $\mathcal{H}_{JI}$ respectively.

Step4: Edge classification. In this step, we filtered out the edges into two sets: **B100** and **B0**. **B100** is the set of edges that are predicted to belong to a constant community, and **B0** is the set of edges that are predicted to belong to a non-constant community. To classify an edge as **B100**, the following conditions should be satisfied:
(a) Both D_{both} and JI should be high. i.e., the edge should have a high percentage of common neighbours, and the subgraph that is induced by both terminals of the edge should be high.

or,

(b) D_{any} is high. i.e., the density induced by all the neighbours of the edge terminals, including the terminals, is high.

or,

(c) D_{tri} is high. i.e., the ratio of the number of triangles containing both u and v to the number of triangles containing at least one of u or v is high.

The workflow diagram regarding the steps of the unsupervised approach is given in Fig. 3.4:

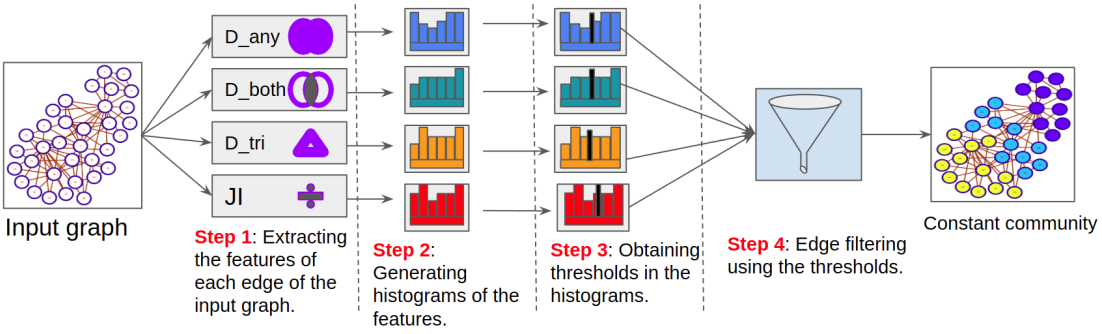


FIGURE 3.4: **Steps to detect the constant community using unsupervised workflow.** In Step 1, features are extracted using the neighbors of each edge. In Step 2, histograms are generated for each feature. In Step 3, by applying an image thresholding algorithm, an optimum threshold is obtained for each histogram. In Step 5, edges are filtered out based on the thresholds, and finally we get the constant communities.

3.2.3 Application of the variants of the thresholding algorithms

The image thresholding methods that we used in the unsupervised approach are now the subject of substantial discussion. We categorise the strategies into two main groups: **binary-thresholding** approach and **multi-thresholding** approach.

Applying binary-thresholding algorithm. When using the binary classification technique, an optimal threshold is independently determined for each histogram, and the results are combined to classify the edges in accordance with the guidelines in Step 4 above. A high value of the characteristics is one that exceeds the threshold set by the relevant thresholding technique, according to our definition. Algorithm 1 describes the implementation of the binary-thresholding technique. In this investigation, we employed three different binary thresholding methods: Bi-level, Histogram concavity, and Binary Otsu.

Applying multi-thresholding algorithm. If an image is divided into many (more than two) divisions, it may sometimes be possible to distinguish between the object and

Algorithm 1: Binary-thresholding method for classifying the edges.

Input : The graph $G(V, E)$. Sets of features D_{any}, D_{both}, D_{tri} and JI for all the edges of G.
Output: Two sets of edges **B100** (for constant community) or **B0** (for non-constant community).

```
/* Finding the thresholds using binary-thresholding method */
[T_Dboth] ← binThresholdMethod(D_both)
[T_Dany] ← binThresholdMethod(D_any)
[T_Dtri] ← binThresholdMethod(D_tri)
[T_JI] ← binThresholdMethod(JI)
/* Filter the edges using the thresholds */
B0 ← Φ; B100 ← Φ;
for each edge e ∈ E do
    if ((D_both(e) > T_Dboth/2 AND JI(e) > T_JI/2) OR (D_any(e) > T_Dany OR D_tri(e) > T_Dtri/2)) then
        B100 ← B100 ∪ e
    else
        B0 ← B0 ∪ e
```

background more easily. To divide an image into more than two divisions, we need to have multiple thresholds on its histogram. The multi-thresholding approach we utilised to identify the optimal thresholds was TSMO (henceforth referred to as Multi-Otsu). The difficulty in using Multi-Otsu is testing every combination of thresholds and features that might possibly exist in order to identify the best one. For instance, if the Multi-Otsu method returns t various thresholds, then with four features, $\mathcal{O}(t^4)$ combinations should be verified in order to find the best solution, which is computationally expensive.

We take into account the thresholds for each feature individually rather than collectively in order to make classification computationally feasible. We consider the threshold that produces an equal number of edges on the right and left sides of it. The intuition behind this is that the number of edges within communities should be at least equal to the number of edges among communities, if not more, given a strong community structure. The initial set of edges that will always be within communities is provided by this optimal threshold. Algorithm 2 describes the implementation of the multi-thresholding technique.

Further improvements. To make further enhancement on the multi-thresholding approach, we have modified Algorithm 2.

M1: Applying iterative multi-thresholding approach. By repeatedly applying Multi-Otsu to the subgraph created by the edges in **B0**, which has the potential to contain additional within-community edges, we increase the accuracy of the algorithm. 2.

Algorithm 2: Edge classification using the multi-thresholding approach

Input : The graph $G(V, E)$. Sets of features D_{any}, D_{both}, D_{tri} and JI for all the edges of G.
Output: Two sets of edges **B100** (for constant community) or **B0** (for non-constant community).

/* Finding the thresholds using multi-thresholding method */
$\{\mathcal{T}_{both_1}, .., \mathcal{T}_{both_{n1}}\} \leftarrow$ `mulThresholdMethod`(D_{both})
$\{\mathcal{T}_{any_1}, .., \mathcal{T}_{any_{n2}}\} \leftarrow$ `mulThresholdMethod`(D_{any})
$\{\mathcal{T}_{tri_1}, .., \mathcal{T}_{tri_{n3}}\} \leftarrow$ `mulThresholdMethod`(D_{tri})
$\{\mathcal{T}_{JI_1}, .., \mathcal{T}_{JI_{n4}}\} \leftarrow$ `mulThresholdMethod`(JI)
/* Using cross products to produce all possible combinations */
$\mathcal{T}_{all\_comb} \leftarrow \{\mathcal{T}_{both_1}, .., \mathcal{T}_{both_{n1}}\} \times \{\mathcal{T}_{any_1}, .., \mathcal{T}_{any_{n2}}\} \times \{\mathcal{T}_{tri_1}, .., \mathcal{T}_{tri_{n3}}\} \times \{\mathcal{T}_{JI_1}, .., \mathcal{T}_{JI_{n4}}\}$
/* Optimum threshold calculation */
$RL \leftarrow \Phi$
for $(\mathcal{T}_{both}, \mathcal{T}_{JI}, \mathcal{T}_{any}, \mathcal{T}_{tri}) \in \mathcal{T}_{all\_comb}$ **do**
  $R \leftarrow max(min($`getNoOfEdgesRight`$(\mathcal{T}_{both}/2),$ `getNoOfEdgesRight`$(\mathcal{T}_{JI}/2)),$ `getNoOfEdgesRight`$(\mathcal{T}_{any}),$ `getNoOfEdgesRight`$(\mathcal{T}_{tri}/2))$
  $L \leftarrow |E| - R$
  $RL \leftarrow RL \cup (1 - \frac{R}{L})$
/* Choose a combination for which value in RL is minimum */
$(\mathcal{T}^*_{both}, \mathcal{T}^*_{JI}, \mathcal{T}^*_{any}, \mathcal{T}^*_{tri}) \leftarrow$ `getOptimumThreshold`$(\mathcal{T}_{all\_comb}, RL)$
/* Classify the edges. */
B100 $\leftarrow \Phi$; **B0** $\leftarrow \Phi$
for $e \in E$ **do**
  if $((D_{both}(e) > \frac{\mathcal{T}^*_{both}}{2}$ *AND* $JI(e) > \frac{\mathcal{T}^*_{JI}}{2})$ *OR* $(D_{any}(e) > \mathcal{T}^*_{any}$ *OR* $D_{tri}(e) > \frac{\mathcal{T}^*_{tri}}{2}))$ **then**
    B100 $\leftarrow$ **B100** $\cup\, e$
  else
    B0 $\leftarrow$ **B0** $\cup\, e$

Based solely on the feature values of the edges in **B0**, we recompute the histograms for the edges in **B0**. We determine the optimum set of thresholds on this set by applying Algorithm 2 and then split **B0** into **B0´** and **B100´** with the same assumptions as **B0** and **B100** respectively. Then, we set **B0** to **B0´** and shift the edges of **B100´** into **B100**.

This iteration is carried out until there is very little change in the threshold ($\leq \delta$, where δ is set to 0.01) and no new edges move from **B0** to **B100**. This technique is known as *Multi-Otsu iterative*.

M2: Fixing the singleton communities. The communities formed by the subgraph induced by the edges in B100 serve as the constant communities once the edges in B100 have been categorised. Nonetheless, certain communities can consist of just one vertex, or form singleton communities, as can be seen in the instance of the blue vertex in FIGURE 3.1.

The majority of community discovery algorithms incorporate this vertex into a nearby community instead of keeping singleton communities. We recognise nodes that do not belong to the constant communities. If both of the node's neighbours are members of

the same community and the node has degree 2, we include it in that community. If the node's neighbours are in separate communities, we shift the node to that community, as there is a 50% chance that at least one of the node's neighbours will be included in one of the two communities. We called this method *Multi-Otsu iterative+SC*.

Empirically, it has been demonstrated that this heuristic results in a marginally higher F1-score. As the heuristic is applied to vertices with a higher degree, the accuracy, however, declines. This is due to the fact that if the vertex were to be categorised as a singleton, it may reside in any of its surrounding communities. The more nearby communities there are, the lower the likelihood that the vertex is in any particular community.

3.2.4 Formulation of various thresholding methods

Now we discuss some mathematical formulation of some of the image-based thresholding methods that we used in our networks for edge classification. While demarcating the background from the foreground of an image, the pixel intensities of the image are used to build the histogram. In our case, the values of a property are used to build the histogram. The mathematical formulations are given below:

Formal overview for Bi-level. Let $m(g)$ be the number of entries having a feature value g, L be the number of feature values, and e_0 and e_{L-1} be the first and last feature values (along the x-axis) in the histogram. So, $e_{mid} = \frac{e_0+e_{L-1}}{2}$ is the middle of these two feature values. Now, the weight of the left side of the e_{mid} be $W_l = \sum_{i=e_0}^{e_{mid}-1} m(i)$ and the right side is $W_r = \sum_{i=e_{mid}}^{e_{L-1}} m(i)$. Based on the proposed algorithm in [129] (Algorithm 4.1), it checks which side is heavier and updates W_l, W_r, e_0 and e_{L-1} accordingly until $e_0 = e_{L-1} = T$ = optimum threshold.

Formal overview for Binary Otsu.

Formal overview. The probability distribution of g is, $p(g) = m(g)/m$; m is the total number of entries. For a threshold T, all entries with feature value below T are given by $n_B(T) = \sum_{i=e_0}^{t} p(i)$ and all entries with feature value over the threshold are given by $n_O(T) = \sum_{i=T}^{e_{L-1}} p(i)$. e_i is the i^{th} feature value, L is the number of the feature values, $t \in \{e_0, ..., e_{L-1}\}$ is the maximum value just below T.

Let $\mu_B(T)$ be the mean, and $\sigma_B^2(T)$ be the variance of all entries with a feature value less than the threshold, which in this case is analogous to the background of the image. Let $\mu_O(T)$ be the mean, and $\sigma_O^2(T)$ be the variance of all entries with feature values above the threshold, which in this case is analogous to the foreground, i.e., the object present in the image. Let σ^2 is the combined variance, and μ is the combined mean over all edges. Given these values, the within class variance can be defined as $\sigma_W^2(T) = n_B(T)\sigma_B^2(T) + n_O(T)\sigma_O^2(T)$ and between class variance can be defined as;
$\sigma_X^2(T) = \sigma^2 - \sigma_W^2(T) = n_B(T)n_O(T)[\mu_B(T) - \mu_O(T)]^2$.

We apply Otsu's method to obtain T such that $\sigma_X^2(T)$ is maximized.

Formal overview.A histogram H can be thought of as a bounded 2-D region: left-bottom $(e_0, 0)$, left-top $(e_0, m(e_0))$, right-top $(e_{L-1}, m(e_{L-1}))$ and right-bottom $(e_{L-1}, 0)$. The concavity of H can be found by constructing the histogram of H. A convex hull $\bar{H}$ of a histogram H can be defined as a smallest convex polygon that contains the histogram H. The concavity can be defined as the set difference between H and $\bar{H}$, i.e., $H - \bar{H}$. $\bar{H}$ can be constructed as follows: starting from start from $(e_0, m(e_0))$, we compute a slope (θ_i) between the left-top point $(e_0, m(e_0))$ and $(e_i, m(e_i))$ for $1 \leq i < L$. Thus, $(e_0, m(e_0))$ and $(e_{k_1}, m(e_{k_1}))$ is a side of the convex hull if θ_{k_1} is the maximum slope. This step will be repeated, i.e, we find the slope of $(e_{k_1}, m(e_{k_1}))$ and $(e_i, m(e_i))$, $k_2 + 1 \leq i < L$ and θ_{k_2} be the largest slope yielding $(e_{k_1}, m(e_{k_1}))$,$(e_{k_2}, m(e_{k_2}))$ and so on, until we reach $L - 1$.

Let $\bar{m}(e_i)$ is the height of $\bar{H}$ at e_i. A point e_i is in a concavity if $(\bar{m}(e_i)$ - $m(e_i)) > 0$. Optimum thresholds can be found at e_t for which the difference $(\bar{m}(e_t)$ - $m(e_t))$ is largest. An example of how the histogram concavity can be found is shown in Figure. 3.5.

3.3 Datasets, ground truths and baselines

In this section, we shall discuss the datasets, ground truths, baselines, and comparisons used in our experiments.

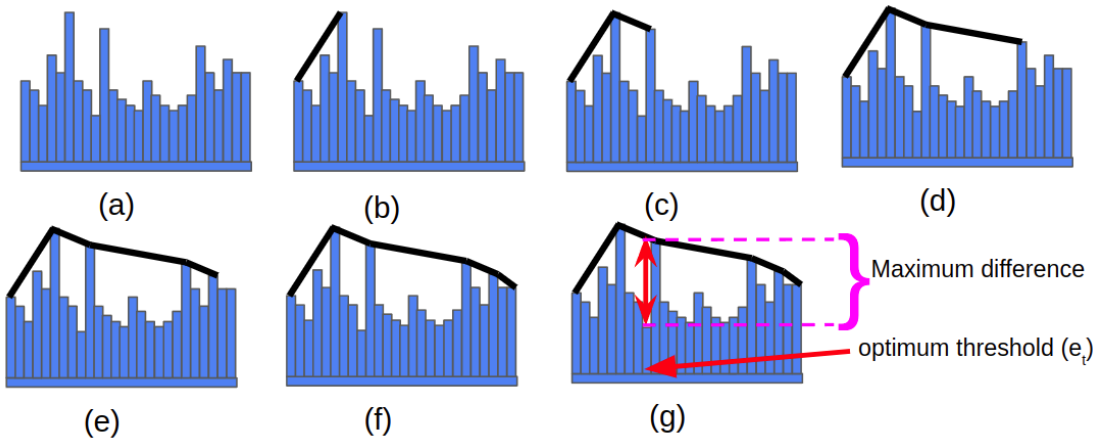


FIGURE 3.5: Using histogram concavity to find the optimum threshold. Figure. (a) denotes the histogram H constructed using the feature values and their respective frequencies. Figures. (b) to (f) represent the construction of the convex hull $\bar{H}$ over H (represented by black lines). In Fig. (g), a point (feature value) e_t can be found for which the difference between $\bar{m}(e_t)$, the frequency of e_t in $\bar{H}$ and $m(e_t)$, the frequency of e_t in H is maximum.

3.3.1 Datasets

In this thesis, we have made use of a variety of real-world networks of varying sizes. Table 3.1 describes the datasets.

3.3.2 Ground-truth generation

By repeatedly executing a community detection approach over a network, we empirically generated constant communities to verify our findings. We modify the order in which the vertices are accessed at each execution by permuting the vertex order. As demonstrated in [233], altering the processing order of the vertices can alter the outcomes of community detection. For the specific community detection technique, the communities that were present in all of these executions were chosen as the constant community ground truth. By executing a community detection algorithm 50 times, we are able to obtain the ground truth constant community. Each time the algorithm is executed, the vertices' order is permuted.

The standard community detection algorithms we have used for the ground-truth generation are: Louvain [29], Infomap [394], and Label Propagation [31]. We used these three algorithms 50 times and generate the ground truth for the small-size networks.

Network	Description	|V|	|E|
Small-size n/ws			
Football [13]	A college football network played in United States. The nodes of this network represent teams (which are denoted by their college names), and the edges reflect regular-season games that take place between the two teams that are connected by those edges.	115	613
Jazz [388]	A network of Jazz musicians. Each vertex represents a musician, and if two musicians have recorded together, they are connected.	198	2742
Dolphin [389]	A network of group of dolphins found in New Zealand. Each node is a dolphin, and each edge is a pair of dolphins who were seen together more often than would be expected by chance.	62	159
Email [390]	An email network of an organization. A node represents an email address and a link among them implies a communication between a pair of email address.	1133	5451
Karate club [13]	A social network of a karate club where a node represents a member of that club and an edge represents an interactions of two members outside the club.	34	77
Polbooks [391]	A network of books of US politics published around 2004 at the time of election. Each node is a book and there is a link between them if two books purchased together	105	441
Medium-size n/ws			
Co-authorship [392]	A network of authors and a link between them represents they share a co-authorship in a paper	103,677	352,183
Com-dblp [393]	DBLP represents a database of computer science bibliographic information. This is also a co-authorship network where the node represents an author and a link represents that two authors publish at least a paper together.	317,080	1,049,866
Com-amazon [393]	It is a network of items in Amazon website. A node represents an item and if two items brought together then there is a link between them.	334,863	925,872
Large-size n/ws			
Com-Youtube [393]	Social network of Youtube. A node represents a user and an edge represents that there is a friendship between two users.	1,134,890	2,987,624
Com-LiveJournal [393]	A blogging social network. A node represents a user and an edge represents that there is a friendship between two users.	3,997,962	34,681,189
Wiki-topcats [393]	A web graph of Wikipedia hyperlinks gathered on 2011. A node represents a web page and two page are connected if there is a hyperlink between one page to another page.	1,791,489	28,511,807

TABLE 3.1: The test suite of real-world networks.

For the medium- and large-sized networks, we have used only the Louvain algorithm for the ground truth generation, as it is the fastest of these three.

3.3.3 Specifics of the algorithms

Semi-supervised: We used GCN to categorise the nodes in the line graph for the semi-supervised approach (and hence the edges in the corresponding graph). We utilise Optuna [395] to tune the hyper-parameters for this GCN model. Table 3.2 contains the list of hyper-parameters that we used. Both classes of 15% training nodes were utilised. We used every node in the network to calculate the F1 scores.

Name	methods/values
Optimizer	Adam, RMSprop, SGD
Learning rate	10^{-5} to 10^{-1}
weight Decay	5×10^{-5} to 5×10^{-1}
Loss Function	Negative Log-likelihood
Number of epochs	200
Number of layers	2

TABLE 3.2: The list of hyper-parameters used in Optuna.

Un-supervised: In the case of the unsupervised approach, we have used different thresholding algorithms such as Bi-level, Histogram concavity, Otsu's binary thresholding, Otsu's multi-thresholding and its variants, etc.

3.3.4 Baselines methods.

When comparing our method to other methods, we employ two baseline algorithms. The first approach is a consensus community algorithm [396] This involves forming a consensus matrix called D based on multiple (we selected 100) executions of community detection. where, D_{ij} is the number of times nodes i and j were co-clustered within the same community. The weighted graphs created from the consensus matrices are iteratively subjected to a community detection method until less than a predefined convergence percent (default value of 2%) of all non-zero entries in D_i have weights of less than one.

The second approach is known as CHAMP (Convex Hull of Admissible Modularity Partitions) [239]. In order to find the subset of partitions that may be optimal while eliminating the remaining partitions, CHAMP determines the modularity optimisation for each partition given a collection of partitions.

3.4 Results

In this section, we compare both the semi-supervised and unsupervised methods with the groundtruths and baselines. Due to the stochastic nature of the baselines, the average outcomes of numerous executions are reported.

3.4.1 F1-scores comparison with the baselines

As previously stated, **B100** is the set of constant community edges. The set of edges in the constant communities that are derived from the ground truths was compared to the set **B100**.

The results (F1-score) regarding the small sized networks are shown in Figs. 3.6, 3.7, and 3.8. In the case of these small-sized networks, the ground truth community edges are obtained by executing the Louvain, Infomap, and Label propagation algorithms, respectively.

The results regarding medium- and large-size networks can be found in Fig. 3.9, where, only the Louvain algorithm is applied to generate the ground truth community edges.

The semi-supervised (Line-GCN) strategy yields the highest F1-score for all the large networks and all but three of the small networks (Polbooks, Jazz, and Email). The Bi-Level thresholding technique performs poorly for a selected number of networks, including Jazz, Email, Com-amazon, Com-Youtube, and Com-liveJ. In the majority of the networks, alternative binary and multi-thresholding approaches functioned effectively. Our methods typically produce better results when compared to the baselines, particularly Line-GCN, Histogram Concavity, Binary Otsu, Multi-Otsu, and their derivatives.

Except for the Coauthorship network (F1-scores = 70%), baseline techniques for medium- and large-scale networks were not finished within a reasonable amount of time (i.e., within 2 days).

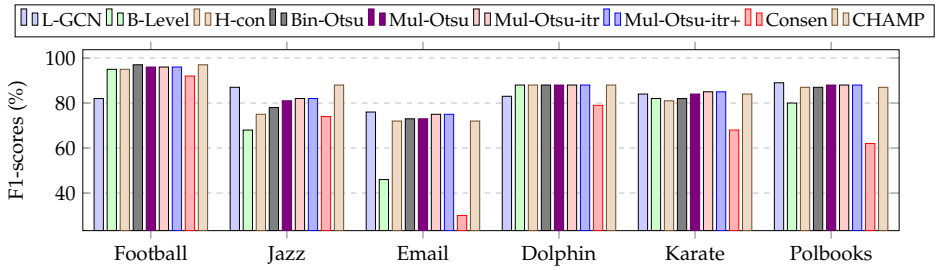


FIGURE 3.6: Performance of different methods for obtaining constant communities for small networks. The ground truth constant community edges are obtained by executing the *Louvain algorithm*. The abbreviated names are as follows; L-GCN: Line-GCN, B-Level: Bi-level, H-Con: Histogram concavity, Bin-Otsu: Binary Otsu, Mul-Otsu: Multi-Otsu, Mul-Otsu-itr: Multi Otsu iterative, Mul-Otsu-itr+: Multi Otsu iterative with singleton community, Consen: Consensus

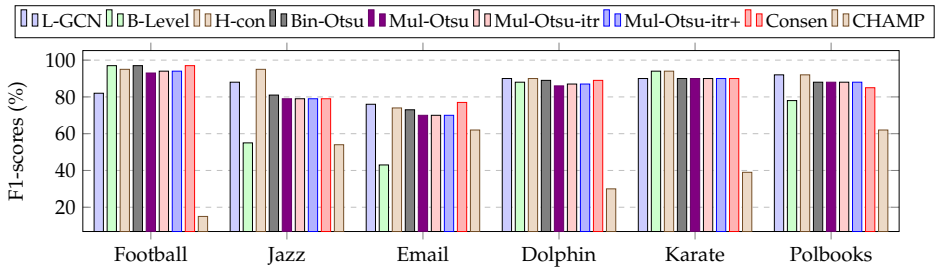


FIGURE 3.7: Performance of different methods for obtaining constant communities for small networks. The ground truth constant community edges are obtained by executing the *Infomap algorithm*. The abbreviated names are as follows; L-GCN: Line-GCN, B-Level: Bi-level, H-Con: Histogram Concavity, Bin-Otsu: Binary Otsu, Mul-Otsu: Multi-Otsu, Mul-Otsu-itr: Multi Otsu iterative, Mul-Otsu-itr+: Multi Otsu iterative with singleton community, Consen: Consensus.

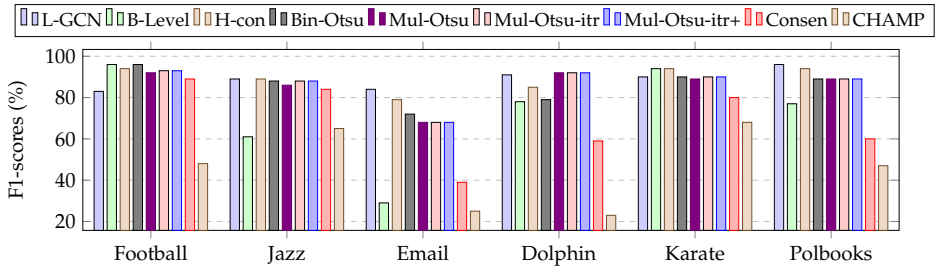


FIGURE 3.8: Performance of different methods for obtaining constant communities for small networks. The ground truth constant community edges are obtained by executing the *Label propagation algorithm*. The abbreviated names are as follows; L-GCN: Line-GCN, B-Level: Bi-level, H-Con: Histogram concavity, Bin-Otsu: Binary Otsu, Mul-Otsu: Multi-Otsu, Mul-Otsu-itr: Multi Otsu iterative, Mul-Otsu-itr+: Multi Otsu iterative with singleton community, Consen: Consensus

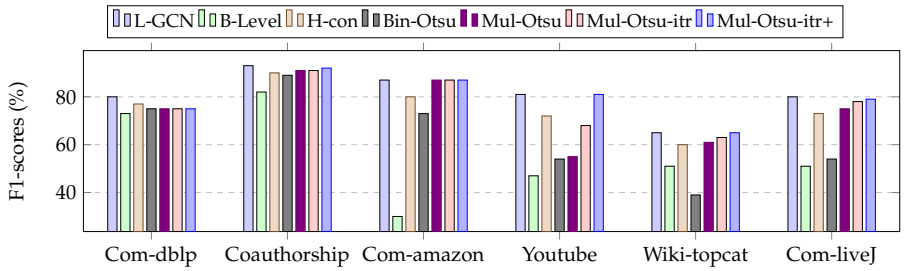


FIGURE 3.9: Performance of different methods for obtaining constant communities for medium and large networks. The ground truth constant community edges are obtained by executing the Louvain algorithm. The baselines are not shown here because they did not end within a sizable amount of time. Only for coauthorship network (not shown in the figure), the consensus method gives 70% F1-scores.

3.4.2 NMI comparison with the ground truth communities.

How closely do the ground-truth constant communities and the predicted constant communities overlap? We build the constant communities from the edges in **B100** by acquiring the connected components in order to determine that. The Normalized Mutual Information (NMI) is then calculated between the set of predicted constant communities and the set of ground truth communities. As shown in Table 3.3, our results produce NMI scores that are on par with or higher than the baselines. More specifically, Line-GCN provides comparable scores (second best or best) to the baselines for the small-sized networks in the majority of situations (except for the Football graph). In the Jazz and Email networks, the unsupervised technique performs the poorest, but in other situations, it performs best or second best. Both the semi-supervised (Line-GCN) and the unsupervised approaches—particularly the multi-Otsu iterative with singleton community—perform well for all medium- and large-sized graphs. For the Com-amazon and Wiki-topcats networks, both techniques get comparable results. Line-GCN produces the best results for Com-dblp and Co-authorship networks, while the unsupervised technique produces the second-best results. For the Com-youtube network, the unsupervised method produces the best results, while Line-GCN produces the second-best results.

Method	Networks										
	foot-ball	**jazz**	**email**	**dolp-hin**	**pol-book**	**kar-ate**	**com-amazon**	**com-dblp**	**coauth-orship**	**you-tube**	**wiki-top**
Semi-supervised (L-GCN) method with best F1	0.86	0.82	0.71	0.84	0.85	0.91	0.82	0.79	0.80	0.73	0.67
Unsupervised method with best F1	0.97	0.81	0.69	0.86	0.84	0.91	0.82	0.75	0.78	0.76	0.67
Consensus with best F1	0.96	0.82	0.77	0.86	0.80	0.90	X	X	0.63	X	X
CHAMP with best F1	0.97	0.83	0.71	0.77	0.79	0.91	X	X	X	X	X

TABLE 3.3: Comparison of NMI. Best results are highlighted in green, second best results are highlighted in blue and the worst in red. X: The process did not end within a sizable amount of time. L-GCN: Line-GCN.

3.4.3 Execution time comparisons.

According to Table 3.4, it can take up to 963 hours to compute ground truth constant communities for the largest network. These ground-truth constant communities are generated by executing louvain's community detection algorithm 50 times. Because of this, it is impractical to identify constant communities in large networks using empirical methods; instead, an effective algorithm like the one described in this paper is required.

Table 3.5 represents the execution time of our algorithm and the baselines. It is evident from the result that for all the networks, our unsupervised algorithm is faster than the baselines. Even though semi-supervised algorithms sometimes produce results on par with unsupervised ones, they need more time. The baselines either take a longer time (than our pro posed methods) to complete or cannot be completed in a reasonable amount of time.

3.4.4 Results under noisy domains

Real-world networks can have missing edges and are frequently noisy. To evaluate how effectively our strategy works in the presence of noise, we tested it using three of the following noise models.

- **Uniform**: Up until a particular number of edges are left, the edges are randomly and repeatedly eliminated.

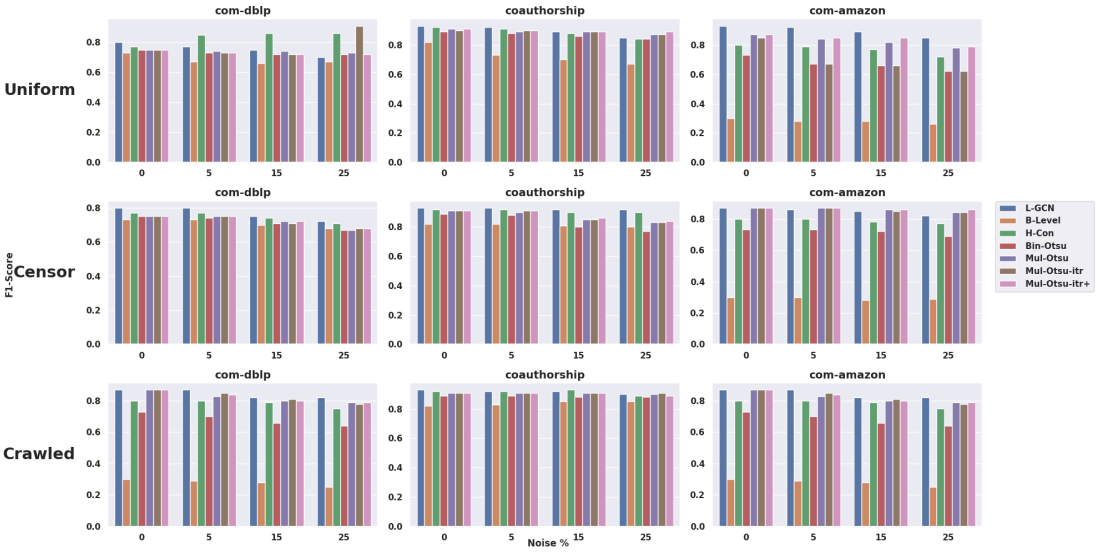


FIGURE 3.10: F1-scores under different noise scenarios.

All networks											
Foot-ball	Jazz	Email	Dol-phin	Kar-ate	Pol-books	Coauth-orship	Com-ama-zon	Com-dblp	You-tube	Com-liveJ	Wiki-topcats
1.06s	5.05s	20.09s	0.49s	0.25s	1.17s	2h26m	5h25m	34h8m	140h8m	872h10m	963h3m

TABLE 3.4: Time required for ground truth generation (#iterations = 50).

- **Crawled**: Use Breadth First Search (BFS) to crawl the network and maintain the necessary number of edges, starting with the node with the least maximum distance from all other nodes.
- **Censored**: Up until a predetermined number of edges are left in the network, at each step, randomly select one of the highest degree nodes and then randomly delete one of its edges.

Only larger networks were used for testing because adding noise to small networks causes them to disconnect. We demonstrate in Fig. 3.10 that our technique provides acceptable accuracy even when the network is noisy.

Method	All networks											
	foot-ball	**jazz**	**email**	**dol-phin**	**kar-ate**	**pol-books**	**coauth-orship**	**com-ama-zon**	**com-dblp**	**you-tube**	**com-liveJ**	**wiki-topcats**
L-GCN	2.13s	19.32s	23.56s	1.55s	1.22s	2.01s	57m21s	1h52m	4h34m	8h34m	37h13m	34h53m
B-Level	0.01s	0.21s	0.62s	0.02s	0.04s	0.04s	19.22s	6.04s	1m10s	8m11s	16h02m	10h22m
H-Con	0.01s	0.21s	0.63s	0.02s	0.04s	0.04s	23.29s	10.15s	1m14s	15m33s	16h34m	10h52m
Bin-Otsu	0.01s	0.23s	0.62s	0.02s	0.03s	0.04s	19.26s	6.17s	1m10s	8m12s	16h04m	10h23m
Mul-Otsu	0.01s	0.30s	0.65s	0.02s	0.03s	0.04s	23.29s	6.47s	1m12s	10m12s	17h03m	12h21m
Mul-Otsu-itr	0.01s	0.35s	0.65s	0.02s	0.03s	0.04s	24.25s	7.13s	1m14s	23m06s	27h03m	24h21m
Mul-Otsu-itr+	0.01s	0.36s	0.65s	0.02s	0.03s	0.04s	24.25s	7.13s	1m19s	26m41s	28h10m	25h21m
Consen (cp=2%)	1.16s	3.17s	9.68s	1.17s	1.30s	1.51s	X	141m	X	X	X	X
CHAMP	1.82s	5.05	18.60s	1.39s	1.16s	1.96s	X	X	X	X	X	X

TABLE 3.5: Time for identifying constant communities for all networks. Best timing performances for each dataset are denoted by green cells and the worst by red cells. X: The process did not end within a sizable amount of time. The abbreviated names are as follows; L-GCN: Line-GCN, B-Level: Bi-level, H-Con: Histogram concavity, Bin-Otsu: Binary Otsu, Mul-Otsu: Multi-Otsu, Mul-Otsu-itr: Multi Otsu iterative, Mul-Otsu-itr+: Multi Otsu iterative with singleton community, Consen: Consensus.

3.5 Case Study: Application of Constant Community Detection Algorithms in a Neighborhood-Based Recommendation System

In this section, we shall see the effect of integrating some community detection algorithms as well as our constant community detection algorithms in the neighbourhood-based recommendation (or recommender) system. Vertices in a graph can be systematically grouped using network communities. As a result, for some applications, such strongly linked clusters of individuals or products will produce better neighbourhoods than those formed in a conventional approach. One way to build a neighbourhood in RSs is to collect the common neighbours in the Recommender System. The CN [397] method generates the neighbourhood in RS by picking the n-nearest neighbours based on the weights $W(u,v)$ between two user (nodes) u and v:

$$W(u,v) = \frac{|S\_Item(u) \cap S\_Item(v)|}{|S\_Item(u)| + |S\_Item(v)|}, \tag{3.5.1}$$

Where $S\_Item(x)$ is the set of items chosen by user x. It is logical to believe that two consumers who choose many things and have many things in common are not the same as two customers who choose many things but have few things in common. We used this common neighbourhood (CN) technique to create n-nearest neighbours for our experiment. Other baseline methods include Louvain, Label Propagation Algorithm (LPA), and Infomap community detection methods for generating a user's or item's

n-nearest neighbours in a RS. As a result, identifying communities in a network of users or objects may be important for calculating n-nearest neighbours and, as a result, offering better recommendations in a RS.

3.5.1 Procedure

The proposed approach of integrating the community/constant community detection algorithms with the neighbourhood-based recommendation approach can be described as follows:

- For each pair (u, v) of users (nodes), calculate the weight $W(u, v)$ according to Equation 3.5.1.
- Build a user-user graph by applying a threshold to the weight.
- Apply a particular community or constant community detection algorithm to produce distinct communities.
- The n-nearest neighbour of a user in a community is generated by selecting the top-n neighbours based on the sorted weight values that belong to the same community.
- Apply Adsorption algorithm [398] to generate the recommendation.

The algorithm for the above steps is given in Algorithm 3

3.5.2 Description of Datasets

In our experiment, we used a variety of real-world datasets. Table 3.6 contains descriptions of the datasets. The Book Crossing dataset is a bipartite graph with nodes that are users and books. If the user reads the book, there is a relationship between the user node and the book node. The nodes in the Amazon Photo dataset represent goods, and a link between two nodes indicates that these two goods were purchased together. The nodes in the DBLP graph represent users, while a link between two nodes indicates two users who collaborated. The Movie Lens network, like the Book Crossing network, is a bipartite network in which nodes in one set represent people and nodes in the other set represent movies.

Algorithm 3: Algorithm for Community Integrated Neighbourhood-based RSs

Input : Dataset D, A threshold t, A constant k.
Output: Item recommendation vector for each user.

```
/* Construct all pair weight matrix from dataset D.                    */
W ← getAllPairWeight(D)
/* Construct user-user (item-item) graph using W and t.                */
UsrGraph ← getUsrUsrGraph(W, t)
/* apply community or constant communityy detection algorithms on
   user-user (item-item) graph                                         */
Comm ← getCommProp(UsrGraph)
/* Build top-k neighbors matrix based on the communities               */
UserNbdMatrix ← buildTopK(Comm, UsrGraph, k)
/* Using top-k neighborhoods for each users, compute preference
   vectors (item recommendations) for each user by applying
   adsorption algorithm.                                               */
UserRecomMatrix ← ApplyAdsorption(D, UsrGraph, UserNbdMatrix)
```

Network	# Vertices	# Edges
Book Crossing [399]	440k	1.1M
Amazon Photo [400]	7k	119k
DBLP [401]	1.3M	18.9M
Movie lens [402]	138k	19M

TABLE 3.6: Dataset descriptions.

3.6 Setup, experiments and results

3.6.1 Preprocessing

We filtered out the subgraph for the Book Crossing and Movie Lens datasets by removing user nodes with degrees less than 4, which means we trimmed out users who do not read more than four books (in the case of the Book Crossing dataset) or rate more than four movies (in the case of the Movie Lens dataset). In the case of the DBLP dataset, we take a subset from 1992 to 2012.

3.6.2 Dataset splits

Each dataset was divided into six sets, as indicated in Table 3.7. We obtained four subsets (s1, s2, s3, and s4) using these splits. Each subset is divided into training and testing sets. For Book Crossing datasets, we use about 20% of preferences in each split. The data from each split in the DBLP dataset is based on three years. In Movie Lens

Subset	Training data	Test Data
s1	T1 ∪ T2	T3
s2	T2 ∪ T3	T4
s3	T3 ∪ T4	T5
s4	T4 ∪ T5	T6

TABLE 3.7: Spliting the dataset into four subsets.

datasets, each split is done equitably depending on the user rating. Finally, with Amazon Photo, the splitting criterion is applied consistently based on the product category.

3.6.3 Competitive methods

We devised and tested the resilient neighbourhood-based RSs by combining the Adsorption method with five competitive network clustering approaches: Common Neighbours (CN), Modularity (Modu), Infomap (Info), Label Propagation method (LPA), our proposed unsupervised (Mul-Otsu-itr+) and semi-supervised (L-GCN) constant community detection algorithms.

3.6.4 Hyper-parameters setup

We varied the number of neighbourhoods from 5 to 10 to 15. We used four subgroups in each example. We tested the evaluation measures on various user-user graphs formed by altering the threshold value t corresponding to the NBC method from 0.0 to 0.80. We set the hyper-parameters P_{inj}, P_{term}, and P_{cont} to 0.10, 0.65, and 0.25, respectively, in the adsorption method. For the top-k suggestions, the value of k is set to 10.

3.6.5 Results

Tables 3.8, 3.9, 3.10, and 3.11 denote the performance of the competing approaches in terms of the Mean Absolute Error (MAE) and Mean Average Precision (MAP) scores corresponding to the Book Crossing, Amazon Photo, DBLP coauthor, and Movie Lens datasets, respectively. We have tuned the neighbourhood size to n and reported the corresponding scores. The **bold** font of the average values of the evaluation measures in each table indicates that the corresponding competitive technique exceeds the other.

				CA			
Nbd	Subset	**CN**	**Louvain**	**Info**	**LPA**	**L-GCN**	**Mul-Otsu-itr+**
				MAP			
5	s1	0.0072	0.0072	0.0074	0.0073	0.0074	0.0074
	s2	0.0071	0.0073	0.0073	0.0073	0.0074	0.0074
	s3	0.0074	0.0072	0.0074	0.0071	0.0074	0.0074
	s4	0.0075	0.0073	0.0072	0.0072	0.0073	0.0075
	Avg.	0.0073	0.0072	0.0073	0.0072	**0.0074**	**0.0074**
10	s1	0.0075	0.0072	0.0073	0.0074	0.0075	0.0074
	s2	0.0074	0.0072	0.0075	0.0072	0.0075	0.0074
	s3	0.0075	0.0074	0.0074	0.0073	0.0075	0.0074
	s4	0.0073	0.0074	0.0074	0.0073	0.0075	0.0074
	Avg.	0.0074	0.0073	0.0074	0.0073	**0.0075**	**0.0074**
15	s1	0.0071	0.0072	0.0074	0.0074	0.0075	0.0075
	s2	0.0074	0.0073	0.0074	0.0073	0.0075	0.0075
	s3	0.0075	0.0072	0.0075	0.0075	0.0075	0.0075
	s4	0.0076	0.0075	0.0073	0.0074	0.0075	0.0075
	Avg.	0.0074	0.0073	0.0074	0.0074	**0.0075**	**0.0075**
				MAE			
5	s1	0.643	0.596	0.641	0.644	0.652	0.653
	s2	0.653	0.614	0.642	0.652	0.653	0.652
	s3	0.637	0.612	0.646	0.636	0.653	0.652
	s4	0.646	0.588	0.646	0.646	0.654	0.652
	Avg.	0.644	0.602	0.643	0.644	**0.653**	0.652
10	s1	0.665	0.591	0.666	0.666	0.676	0.675
	s2	0.673	0.596	0.676	0.676	0.677	0.676
	s3	0.678	0.61	0.671	0.674	0.676	0.676
	s4	0.675	0.612	0.671	0.676	0.677	0.675
	Avg.	0.672	0.602	0.671	0.673	**0.677**	0.676
15	s1	0.663	0.596	0.676	0.673	0.676	0.676
	s2	0.665	0.614	0.672	0.668	0.677	0.675
	s3	0.676	0.611	0.676	0.676	0.677	0.675
	s4	0.678	0.588	0.669	0.675	0.678	0.676
	Avg.	0.670	0.602	0.673	0.673	**0.677**	0.676

TABLE 3.8: MAP and MAE scores for the Book Crossing dataset. Nbd, CA, CN, Modu, Info and LPA stands for Neighbourhood size, Clustering Approaches, Common Neighbors, Modularity, Infomap and Label Propagation Algorithm respectively. The size of the neighbourhood (n) is changed between five and fifteen, and the number of recommendations, i.e., the value of k is set at 10.

According to Table 3.10, as n increases, the MAP and MAE values of the Common Neighbourhood (CN) technique drop in comparison to the other datasets (tables). For Louvain, both the MAP and MAE scores decrease. For LPA, the MAP scores decrease a little, but the MAE score increases. For Infomap and the other two constant community detection algorithms (L-GCN and Mul-Otsu-itr+), both the MAP and MAE scores increase.

The semi-supervised constant community detection algorithm performs better in almost all the datasets compared to the competitive approaches in terms of all the evaluation metrics, as clearly seen from Tables 3.8-3.11. Again, as the value of n increases, the performance of all the algorithms increases, corresponding to all the datasets in terms of MAP and MAE values.

				CA			
Nbd	Subset	**CN**	**Louvain**	**Info**	**LPA**	**L-GCN**	**Mul-Otsu-itr+**
				MAP			
5	s1	0.0121	0.0134	0.0227	0.0193	0.0232	0.0220
	s2	0.0121	0.0133	0.0213	0.0216	0.0215	0.0216
	s3	0.0113	0.0113	0.0215	0.0237	0.0233	0.0229
	s4	0.0154	0.0182	0.0225	0.0216	0.0230	0.0231
	Avg.	0.0130	0.0140	0.0220	0.0215	**0.0228**	0.0224
10	s1	0.0141	0.0145	0.0257	0.0231	0.0248	0.0256
	s2	0.0128	0.0137	0.0225	0.0212	0.0246	0.0235
	s3	0.0145	0.0154	0.0241	0.0219	0.0241	0.0243
	s4	0.0144	0.0152	0.0235	0.0221	0.0241	0.0243
	Avg.	0.0139	0.0147	0.0239	0.0220	**0.0244**	0.0244
15	s1	0.0131	0.0141	0.0238	0.0232	0.0246	0.0244
	s2	0.0152	0.0176	0.0244	0.0231	0.0244	0.0244
	s3	0.0113	0.0178	0.0241	0.0231	0.0245	0.0241
	s4	0.0173	0.0178	0.0246	0.0228	0.0245	0.0246
	Avg.	0.0142	0.0168	0.0242	0.0230	**0.0245**	0.0244
				MAE			
5	s1	0.481	0.523	0.547	0.545	0.555	0.557
	s2	0.475	0.539	0.567	0.546	0.556	0.563
	s3	0.495	0.545	0.573	0.563	0.557	0.554
	s4	0.466	0.524	0.573	0.575	0.568	0.566
	Avg.	0.479	0.532	**0.565**	0.557	0.559	0.560
10	s1	0.463	0.535	0.583	0.575	0.582	0.576
	s2	0.494	0.536	0.583	0.575	0.582	0.575
	s3	0.496	0.577	0.576	0.566	0.586	0.574
	s4	0.496	0.567	0.578	0.567	0.587	0.577
	Avg.	0.487	0.553	0.580	0.570	**0.584**	0.576
15	s1	0.482	0.564	0.584	0.579	0.596	0.582
	s2	0.515	0.547	0.581	0.563	0.592	0.583
	s3	0.502	0.555	0.582	0.584	0.584	0.582
	s4	0.491	0.576	0.582	0.568	0.584	0.588
	Avg.	0.497	0.560	0.582	0.573	**0.589**	0.584

TABLE 3.9: MAP and MAE scores for the Amazon Photo dataset. Nbd, CA, CN, Modu, Info and LPA stands for Neighbourhood size, Clustering Approaches, Common Neighbors, Modularity, Infomap and Label Propagation Algorithm respectively. The size of the neighbourhood (n) is changed between five and fifteen, and the number of recommendations, i.e., the value of k is set at 10.

3.7 Discussion

To find stable communities in huge networks, we used both the semi-supervised and unsupervised categorization methods. The training data needed for the semi-supervised technique is not always easy to come by, despite the fact that it yields good results. We also used an unsupervised technique that was based on the idea of segmenting images. We demonstrated that the unsupervised method yields just as good, if not better, outcomes as the semi-supervised method. We have made a significant improvement in the reliability of large-scale network community detection with our study. In the unsupervised method, we manually found edge features to represent the graph, whereas in the semi-supervised approach, we used Graph Convolutional Neural Network as the

				CA			
Nbd	Subset	**CN**	**Louvain**	**Info**	**LPA**	**L-GCN**	**Mul-Otsu-itr+**
				MAP			
5	s1	0.0173	0.0172	0.0181	0.0181	0.0187	0.0179
	s2	0.0153	0.0168	0.0184	0.0183	0.0187	0.0174
	s3	0.0167	0.0169	0.0183	0.0182	0.0176	0.0182
	s4	0.0177	0.0178	0.0183	0.0181	0.0183	0.0181
	Avg.	0.0167	0.0171	0.0182	0.0181	**0.0183**	0.0179
10	s1	0.0161	0.0176	0.0182	0.0179	0.0186	0.0182
	s2	0.016	0.0163	0.0183	0.0181	0.0187	0.0181
	s3	0.0163	0.0171	0.0182	0.0181	0.0177	0.0181
	s4	0.0161	0.0177	0.0183	0.0178	0.0176	0.0181
	Avg.	0.0161	0.0171	0.0182	0.0179	**0.0183**	0.0181
15	s1	0.0158	0.0171	0.0184	0.0180	0.0189	0.0182
	s2	0.0158	0.0172	0.0184	0.0179	0.0184	0.0182
	s3	0.0156	0.0169	0.0182	0.0181	0.0187	0.0184
	s4	0.0157	0.0171	0.0184	0.0179	0.0179	0.0184
	Avg.	0.0157	0.0170	0.0183	0.0179	**0.0185**	0.0183
				MAE			
5	s1	0.529	0.523	0.567	0.569	0.562	0.562
	s2	0.522	0.531	0.585	0.565	0.583	0.563
	s3	0.513	0.542	0.596	0.575	0.584	0.565
	s4	0.512	0.534	0.567	0.556	0.585	0.584
	Avg.	0.519	0.532	0.578	0.566	**0.579**	0.569
10	s1	0.514	0.545	0.573	0.594	0.594	0.593
	s2	0.514	0.526	0.619	0.589	0.584	0.596
	s3	0.514	0.527	0.586	0.583	0.594	0.594
	s4	0.517	0.526	0.595	0.596	0.586	0.592
	Avg.	0.514	0.531	**0.593**	0.590	0.590	**0.593**
15	s1	0.515	0.527	0.592	0.579	0.593	0.587
	s2	0.504	0.537	0.602	0.588	0.596	0.596
	s3	0.514	0.531	0.596	0.586	0.597	0.593
	s4	0.516	0.524	0.589	0.614	0.593	0.599
	Avg.	0.512	0.529	**0.594**	0.591	**0.594**	0.593

TABLE 3.10: MAP and MAE scores for the DBLP dataset. Nbd, CA, CN, Modu, Info and LPA stands for Neighbourhood size, Clustering Approaches, Common Neighbors, Modularity, Infomap and Label Propagation Algorithm respectively. The size of the neighbourhood (n) is changed between five and fifteen, and the number of recommendations, i.e., the value of k is set at 10.

automated feature learner to represent the graph. Finally, using this graph/network representation, we found the constant communities in a graph by filtering the proper edges. We have also done a case study in which we showed that the constant community found in our approach can be used in a recommendation system to achieve better results.

3.8 Conclusion

Due to the "NP-hard" nature of the problem, identifying communities within networks is an extremely difficult undertaking. The majority of community detection algorithms,

				CA			
Nbd	Subset	**CN**	**Louvain**	**Info**	**LPA**	**L-GCN**	**Mul-Otsu-itr+**
				MAP			
5	s1	0.0154	0.0152	0.0151	0.0153	0.0155	0.0154
	s2	0.0152	0.015	0.0159	0.0157	0.0156	0.0155
	s3	0.0152	0.0149	0.0159	0.0155	0.0154	0.015
	s4	0.0155	0.0151	0.0152	0.0156	0.0155	0.0151
	Avg.	0.0153	0.0150	**0.0155**	**0.0155**	**0.0155**	0.0152
10	s1	0.0152	0.0153	0.0152	0.0157	0.0152	0.0153
	s2	0.0151	0.0154	0.0156	0.0155	0.0157	0.0154
	s3	0.0151	0.0152	0.0162	0.0155	0.0166	0.0151
	s4	0.0155	0.0153	0.0159	0.0157	0.0161	0.0152
	Avg.	0.0152	0.0153	0.0157	0.0156	**0.0159**	0.0152
15	s1	0.0151	0.0153	0.0159	0.0158	0.0152	0.0155
	s2	0.0154	0.0153	0.0157	0.0157	0.0156	0.0155
	s3	0.0151	0.0154	0.0158	0.0158	0.0166	0.0157
	s4	0.015	0.0156	0.0156	0.0157	0.0165	0.0152
	Avg.	0.0151	0.0154	0.0157	0.0157	**0.0160**	0.0154
				MAE			
5	s1	0.491	0.492	0.512	0.524	0.538	0.522
	s2	0.488	0.498	0.531	0.527	0.542	0.538
	s3	0.486	0.488	0.532	0.525	0.535	0.542
	s4	0.482	0.482	0.531	0.526	0.536	0.533
	Avg.	0.486	0.490	**0.526**	0.525	0.538	0.534
10	s1	0.491	0.502	0.532	0.532	0.545	0.542
	s2	0.480	0.512	0.545	0.538	0.544	0.496
	s3	0.485	0.514	0.546	0.539	0.546	0.499
	s4	0.482	0.512	0.513	0.532	0.548	0.538
	Avg.	0.484	0.510	0.534	0.532	**0.546**	0.520
15	s1	0.482	0.521	0.537	0.538	0.545	0.531
	s2	0.488	0.501	0.538	0.534	0.543	0.529
	s3	0.482	0.492	0.535	0.539	0.546	0.539
	s4	0.483	0.522	0.536	0.536	0.549	0.535
	Avg.	0.483	0.509	0.536	0.536	**0.546**	0.534

TABLE 3.11: MAP and MAE scores for the Movie Lens dataset. Nbd, CA, CN, Modu, Info and LPA stands for Neighbourhood size, Clustering Approaches, Common Neighbors, Modularity, Infomap and Label Propagation Algorithm respectively. The size of the neighbourhood (n) is changed between five and fifteen, and the number of recommendations, i.e., the value of k is set at 10.

as a consequence of this fact, are based on optimising some objective function; hence, the outcomes of these algorithms are stochastic in nature. Alterations to the community structure on each execution cause a great deal of trouble because the detection of communities is of vital significance in networks due to the wide variety of applications that they support. In this chapter, we talked about how we solved the problem by constructing some algorithms that used certain hand-made and automated feature engineering (graph representations) on the network. As a result, we ended up with two novel methods: unsupervised and semi-supervised. In the unsupervised method, our inspiration was the classical image thresholding algorithm, and in the semi-supervised method, we used a line graph and Graph Convolutional Neural Network (GCN) to find the constant communities in a complex network.

Part - II

Study the goodness of representations of a network

Publication:

- Anjan Chowdhury, Sriram Srinivasan, Sanjukta Bhowmick, Animesh Mukherjee, and Kuntal Ghosh. "Improving Node Classification Accuracy of GNN through Input and Output Intervention." The ACM Transactions on Knowledge Discovery from Data (TKDD), Volume 18, Issue 1, Article No. 17, pp. 1-31, https://doi.org/10.1145/3610535, 2023.

Chapter 4

Improving the Accuracy of Graph Neural Network

Chapter summary: In the previous chapter, our study contributed to a challenging problem of complex network field where we studied the importance of constant community in a complex network and proposed two approaches (unsupervised and semi-supervised) to discovering it. Both approaches use graph or network representation to correctly find the constant communities. In the semi-supervised approach, we introduced the Graph Convolutional Neural Network (GCN) as a graph representation learner. To train the GCN, we used 15% training nodes. But what happens if we do not have enough training nodes to train it? According to experimental findings, the accuracy of GCN drastically drops when the number of training nodes is reduced. The reason is due to the poor representation learning of the GCN on the graph, i.e., with the small number of training nodes, the GCN could not capture the features of the nodes accurately. In this chapter, we outline the problem and offer a potential resolution. In fact, we generalised the GCN and expanded the scope of the issue using Graph Neural Network (GNN). As a result, this chapter contributes to the field of graph-based deep learning.

4.1 Introduction

A crucial issue in data mining is vertex classification, which entails finding vertices in a complex network that belong to the same class. With the development of datasets with abundant information, vertex-related features now also contribute to classification. Hence, semi-supervised methods like Graph Neural Networks (GNN) [43, 195, 44, 297, 289, 194, 300, 113], which can take into account both structural and feature vector data, have gained popularity as vertex classification tools.A GNN model begins with a collection of training nodes, also known as seeds, for which the correct classes or labels are already known. It performs its operations by first gathering the features of each node's neighbours and then conducting a form of Laplacian smoothing to change the features of that node in such a way that the nodes that are assigned to the same class after smoothing have similar feature sets.

The main goal after establishing any GNN model is to adjust the hyperparameters to improve accuracy. Two simplistic methods for improving a neural network model's accuracy include (i) increasing the number of hidden layers and (ii) increasing the training nodes.

Increasing the number of hidden layers. A neural network model with poor representational capacity [403] could occasionally have trouble fitting the training set. We can boost the model's ability to represent more information by adding more layers. Yet, adding more hidden units results in higher time and memory costs. Moreover, for some GNN models, adding more hidden units frequently exacerbates the oversmoothing issue. According to the discussion in [296], the GCN (a version of the GNN) with multiple hidden layers, which can cause nodes from different classes to have nearly identical features after the final smoothing process.

Increasing the training nodes. The accuracy rises as the number of training nodes increases (see Fig. 4.1). Yet, the aim of semi-supervised learning is defeated by merely increasing the number of training nodes. A more recent approach [296] has instead concentrated on agnostically growing the number of training nodes by using random walks from a limited number of starting training nodes with known labels. Using (a)

the label data of a few seed nodes and (b) the network's topology, this technique increases the training set. The technique is similar to an unsupervised set expansion technique [404, 405] and does not require extra label information.

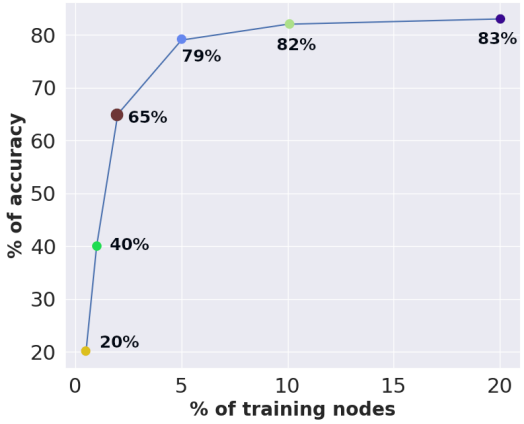


FIGURE 4.1: Accuracy dropped when the percentage of training nodes is near 0%. This experiment was done on the Cora Network [406] using the Graph Convolutional Network.

Problems with adding more training nodes. The GNN's node classification accuracy can be improved by agnostically adding more training nodes using the random walk method. But sometimes it suffers from the following problem: **(i)** This technique frequently results in training nodes that are positioned close to the seed nodes. To train a GNN, a set of seed nodes is collected from each class. However, some of these subgraphs might not be included in the training set if the same class is dispersed among non-contiguous subgraphs. If all of the nodes in the extended training set belong to the same subgraph, the nodes from the other subgraphs will be unrepresented, resulting in classification errors. Therefore, some pre-processing must be used to make sure that the training nodes are dispersed across the network and not concentrated in any particular areas. In Fig. 4.2, we provide a hypothetical scenario to clarify this concept. **(ii)** From a set of initial seeds, the extra nodes accumulated with a random walk are relabeled with the same colour as those initial seeds. In reality, though, a number of relabeled nodes may belong to various classes. When these incorrectly relabeled nodes are employed as training nodes, accuracy may suffer. We have observed in particular that the result contains localised areas of incorrectly categorised nodes. Thus, to find and change any potentially incorrectly labelled nodes, post-processing is crucial.

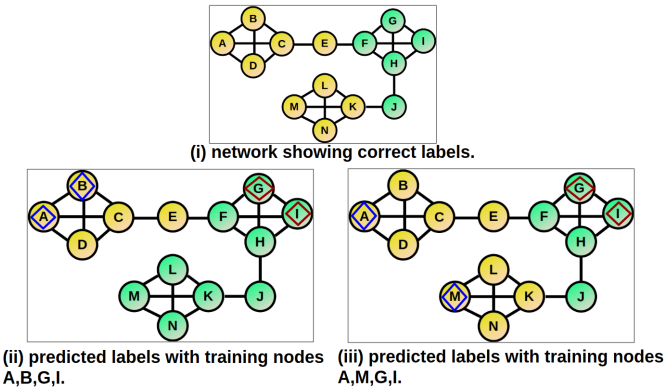


FIGURE 4.2: The collection of training nodes of the same class but non-contiguous sub-graphs improves accuracy. Top (i), the correct labels of the nodes. Bottom left (ii), training nodes are {A, B} and {I, G} from the two different classes. The subgraph {K,L,M,N} is erroneously subsumed into the green class. Bottom right (iii), training nodes are {A, M} and {G, I}. The yellow class has representative from two of its sub-graphs, and correct classification is obtained.

Main contribution. In this study, our primary contribution is to develop novel input and output intervention methods to address these issues. These methods include (i) selecting a set of training nodes that are distributed along various non-contiguous sub-graphs of the same class and (ii) identifying misclassified nodes and correcting their labels. Both of these steps are performed in order to address the two problems mentioned above.

Input intervention: We make use of the structure of the graph by combining random walks as well as the feature vector that is associated with each node, and we apply some preprocessing to the training nodes in order to improve the representational capacity of the GNN and hence the training accuracy.

Output intervention: We first identify nodes with low confidence in classification by utilising the confidence vector that is linked to each node at the output. Then, we relabel the low-confidence nodes with the labels of their neighbouring high-confidence nodes for further improvement of accuracy.

4.2 Proposed strategies

Here, we lay out our two primary contributions: (i) an intervention at the input level to increase the spatial variety of the training set; and (ii) an intervention at the output level to identify and relabel the misclassified nodes. Figure 4.3 shows all the steps.

4.2.1 Input level intervention

We take a look at two different approaches known as ParWalk and DeepWalk, both of which involve expanding the set of training nodes using a variety of distinct approaches to random walks. Nevertheless, just increasing the training nodes in a graph might not be enough to capture all of the instances of the same class if they are spread out across multiple non-contiguous subgraphs in the graph. We make use of the well-known clustering (K-means) and classification (K-NN) methods in order to locate the training nodes from several non-contiguous sub-graphs that have the same class label. The stages involved in this input intervention method can be divided into two primary categories. A set of initial training nodes (seed nodes) selection steps is required before these steps in order to proceed.

Selection of seed nodes: For each class $c \in \{1, 2, ..., k\}$, a set of nodes (I_c) is selected and accumulated in the initial training node set I. Thus, formally,

$$I = \{I_1, I_2, .., I_k\}$$

Step 1: The idea of this step is to apply the random walk or embedding algorithm to the seed nodes for each class to expand the training nodes' set. The random walk or node embedding can be obtained as follows:

- For PaRWalk (Algorithm 4), a transition probability matrix $\mathcal{A}$ is constructed (which is also termed the random walk). Given a graph Laplacian Γ, the transition probability matrix $\mathcal{A}$ can be constructed as $\mathcal{A} = (\Gamma + \alpha\Lambda)^{-1}$, where $\Gamma = D - A$, D is the degree matrix $D = diag(d_1, d_2, ..., d_{|V|})$, d_i is the degree of node v_i and A is the adjacency matrix, $\alpha > 0$ is a scalar value, $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is known as the regularizer, and $\lambda_i \geq 0$ is some arbitrary value.
- For DeepWalk (Algorithm 5), first, we constructed the model with the assistance of the algorithm that is described in [250]. The parameters that we used for this algorithm were the number of walks ($\mathcal{W}_p$) per node and the walk length ($\mathcal{W}_l$). Then we fit the graph inside the model to get the node embedding.

After constructing the transition probability matrix, or node embedding, the next task was to expand the initial seed node sets. For that, for each seed node's set I_c of class c, we use random walk (transition probability matrix) or node embedding to agnostically obtain at most t number of the nodes that are most similar to the nodes in I_c. Now t is defined as:

$$t = \frac{|I_c| \times b \times \eta}{\sum_{c=1}^{k} |I_c|}$$

b is a constant, $\eta = |V|/(d_{avg})^{\tau}$, d_{avg} is the average degree of the graph G and τ is the number of layers of the GNN. The concept of *most similar nodes* can be defined as:

- In the case of PaRWalk, a node i is more similar to node j than node k if

  $$a_{ij} > a_{ik}$$

  where a_{ij} and a_{ik} are the absorption probabilities corresponding to node pairs (i, j) and (i, k) respectively.

- In the case of DeepWalk, a node i is more similar to a node j than a node k if

  $$COS(f(i), f(j)) > COS(f(i), f(k))$$

  $COS(f(i), f(j))$ is the Cosine Similarity [407] between vector $f(i)$ and vector $f(j)$, $f(n) \in \mathbb{R}^d$ is the embedded vector representing a node n of size d.

Step 2: Using the random walk or node embedding technique, the set of extended training nodes X was constructed in Step 1. But as previously discussed, simply increasing the training node set may not include nodes from the non-contiguous subgraphs for a particular class. Therefore, to include these nodes, we employ clustering and classification techniques, which are discussed in this step.

After providing the expanded set of training nodes X, we proceed to carry out an unsupervised classification utilising the K-means algorithm in order to divide the set X into two clusters $Clus1$ and $Clus2$ according to the feature vector of the nodes. We hypothesise that the smaller cluster will aggregate nodes belonging to different classes, whereas the bigger cluster will have a greater number of nodes belonging to the same class as the seed. This hypothesis is founded on the hunch that the vast majority of the

nodes located in close proximity to the seed will belong to the same category and, as a result, will be components of the larger cluster ($C_{max} = max\{Clus1, Clus2\}$). In fact, as demonstrated by the findings, the application of this heuristic yields much better categorization results.

After that, we change out the potentially accurate locally clustered nodes from set C_{max} with new nodes that are spread out across the graph. We use the K-Nearest Neighbors (K-NN) approach to the feature vector space of the nodes in order to locate the $|C_{max}|$ nodes that are located in the closest proximity to the centroid of the nodes in set C_{max}. These new nodes were chosen from the set of nodes that are not in X. This new set of nodes comprises the set C'_{max}, and they are the set of extended nodes that were obtained from the initial seed that was used to start the random walk.

Usually, we are only able to apply the set of training nodes designated as C'_{max}. On the other hand, we intend to evaluate the advantages of utilising spatially diverse nodes in contrast to those that are generated through random walk approaches. As a result, we merge the set C'_{max} with the set C_{min} (nodes that may have been mislabeled), which results in the creation of our ultimate training set X'. Note that the nodes that could be incorrectly identified are located in relatively close proximity to one another, while the nodes that could be correctly classified are spread out across the network. We also gave the nodes in the expanded set X' a label that corresponds to the label of the starting seed nodes.

Finally, in order to create the complete training set, we combine the relabeled extended nodes that we have gotten to match all classes.

4.2.2 Output level intervention

The identification of misclassified nodes generated by a GNN's output and their agnostic relabeling to their possibly accurate labels constitute our second contribution.

The output of the GNN is passed through a softmax layer to obtain the vector of probabilities, $\mathcal{P}_u = [p_{u1}, p_{u2}, ..., p_{un}]$ of node $u \in V$, where, p_{ui} is the probability of node u belonging to class i. This vector $\mathcal{P}_u$ is what we refer to as a *confidence vector*. The

Algorithm 4: Input level intervention using PaRWalk

Input : $G(V, E), \Gamma, \Lambda, t, I, \alpha$
Output: Expanded set of training nodes $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$;
$\mathcal{A} \leftarrow (\Gamma + \alpha\Lambda)^{-1}$;
for *each class* $c \in \{1, 2, ..., k\}$ **do**
  $\mathcal{A}_S \leftarrow \sum_{i \in I_c} \mathcal{A}_{:,i}$;
  $X \leftarrow$ select the t highest valued indices from $\mathcal{A}_S$;
  $Clus1, Clus2 \leftarrow K$-means$(X, Size = 2)$;
  $C_{max} \leftarrow \max(Clus1, Clus2)$;
  $C_{min} \leftarrow \min(Clus1, Clus2)$;
  $C_{cent} \leftarrow$ centroid of C_{max};
  $C'_{max} \leftarrow$ using K-NN, select $|C_{max}|$ nearest nodes to C_{cent} from $(V - X)$;
  $X' \leftarrow C'_{max} \cup C_{min}$;
  for *each node* $x \in X'$ **do**
    $Label[x] \leftarrow l_c$;
  end
  $\mathcal{S} \leftarrow \mathcal{S} \cup X'$;
end

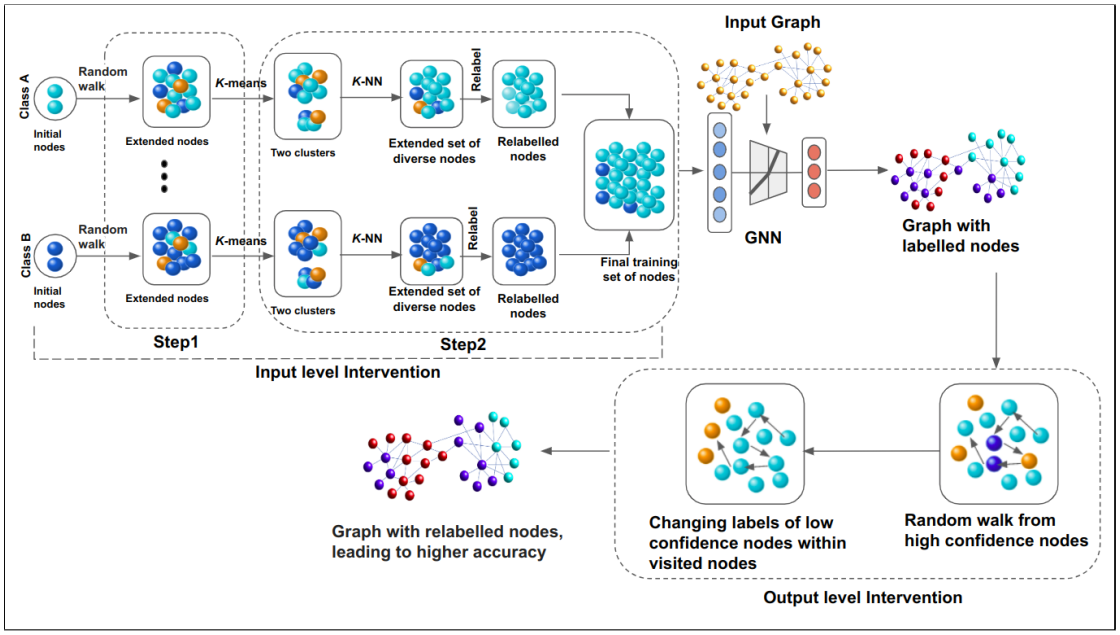


FIGURE 4.3: Our proposed input and output interventions to GNN-based node classification.

confidence of a node u can be defined as:

$$Conf(u) = max\{\mathcal{P}_u\}$$

Algorithm 5: Algorithm for input level intervention with DeepWalk

Input : $G(V, E)$, I, t, $\mathcal{W}_P$, $\mathcal{W}_L$
Output: Expanded set of training nodes $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$;
$Model \leftarrow DeepWalk(\mathcal{W}_P, \mathcal{W}_L)$;
$Emb \leftarrow$ Fit G in the Model to get the node embedding;
for *each class* $c \in \{1, 2, ..., k\}$ **do**
  $X \leftarrow \emptyset$;
  for *each node* $n \in I_c$ **do**
    $X_n \leftarrow$ get nearest $\frac{t}{|I_c|}$ nodes of n using Emb;
    $X \leftarrow X \cup X_n$;
  end
  $Clus1, Clus2 \leftarrow K$-means($X$, $Size = 2$);
  $C_{max} \leftarrow$ max($Clus1$, $Clus2$);
  $C_{min} \leftarrow$ min($Clus1$, $Clus2$);
  $C_{cent} \leftarrow$ centroid of C_{max};
  $C'_{max} \leftarrow$ using K-NN, select $|C_{max}|$ nearest nodes to C_{cent} from $(V - X)$;
  $X' \leftarrow C'_{max} \cup C_{min}$;
  for *each node* $x \in X'$ **do**
    $Label[x] \leftarrow l_c$;
  end
  $\mathcal{S} \leftarrow \mathcal{S} \cup X'$;
end

Next, we compute the mean (μ) and standard deviation (σ) of the confidence score distribution across all of the nodes. A node is said to have high confidence if

$$Conf(i) \geq \mu + \alpha \times \sigma$$

and low confidence otherwise.

We generated a number of high confidence node sets denoted by HCN for each of the classes. Where,

$$HCN = \{HCN_1, HCN_2, ..., HCN_k\}$$

HCN_c stands for the set of high confidence nodes belonging to the class c. Then from each HCN_c we use either PaRWalk or DeepWalk to retrieve a collection of nodes denoted by RWN of size atmost t. In this case, the size of t is determined by the proportion of the number of nodes in G to the total number of classes in G. If a certain node in the RWN has a low confidence score, then the label of that node will be changed according to the following rule: We start by creating a set that is the union of the nodes in HCN_c and the neighbours Nbd of the present node. Then, we select the label l_c of the node

whose confidence is the highest, and we re-label the node with the lowest confidence with label l_c.

Algorithm 6: Algorithm for output level intervention using PaRWalk

Input : The network $G(V, E)$, Predicted label of all nodes *PredLabel*, set size t, Confidence of each node $Conf$, Hyper parameter α, mean (μ) and standard deviation (σ) of the distribution of confidence of all the nodes, graph laplacian Γ, scalar value α, regularizer Λ,.

Output: Modified *PredLabel*.

$\mathcal{A} \leftarrow (\Gamma + \alpha\Lambda)^{-1}$;
$HCN \leftarrow$ select a set of high confidence nodes for every class using $Conf$, μ and $\alpha * \sigma$;
for *class* $c \in \{1, 2, ..., k\}$ **do**
  $\mathcal{A}_S \leftarrow \sum_{i \in HCN_c} \mathcal{A}_{:,i}$;
  $RWN \leftarrow$ select the t highest valued indices from $\mathcal{A}_S$;
  `/* change the label of the low confidence nodes */`
  for *node* $u \in RWN$ **do**
    $\mathcal{S} \leftarrow HCN_c \cup Nbd(u)$;
    $f \leftarrow$ select a node in $\mathcal{S}$;
    $maxConf \leftarrow Conf(f)$;
    if $Conf(u) \leq \mu$ - $\alpha * \sigma$ **then**
      for *node* $v \in \mathcal{S}$ **do**
        if $Conf(v) > maxConf$ **then**
          $maxConf \leftarrow Conf(v)$;
          $l \leftarrow \arg\max_i(CV_v)$;
        end
      end
      $PredLabel[u] \leftarrow l$
    end
  end
end

4.3 Rationale and error analysis

Rationale: We now present an explanation as to why the classification has improved as a result of our intervention strategies. The random walks in both methods have the goal of identifying the local structures based on the assumption that nodes that are structurally local belong to the same class.

By first choosing the set of nodes that are most likely to belong to the right class and then expanding this set by comparing similar feature vectors rather than locally close nodes, our input intervention seeks to reduce the discrepancies between the structure

Algorithm 7: Algorithm for output level intervention using DeepWalk

Input : The network $G(V, E)$, Predicted label of all nodes $PredLabel$, set size t, Confidence of each node $Conf$, Hyper parameter α, mean (μ) and standard deviation (σ) of the distribution of confidence of all the nodes, walks per node $\mathcal{W}_P$, walk length $\mathcal{W}_L$

Output: Modified $PredLabel$.

$Model \leftarrow DeepWalk(\mathcal{W}_P, \mathcal{W}_L)$;
$Emb \leftarrow$ Fit G in the Model to get the node embedding;
$HCN \leftarrow$ select a set of high confidence nodes for every class using $Conf$, μ and $\alpha * \sigma$;
for *class* $c \in \{1, 2, ..., k\}$ **do**
    $RWN \leftarrow \emptyset$;
    for *node* $n \in HCN$ **do**
        $X \leftarrow$ get nearest $\frac{t}{|HCN_c|}$ nodes of n using Emb;
        $RWN \leftarrow RWN \cup X$;
    end
    `/* Change the label of the low confidence nodes */`
    for *node* $u \in RWN$ **do**
        $\mathcal{S} \leftarrow HCN_c \cup Nbd(u)$;
        $f \leftarrow$ select a node in $\mathcal{S}$;
        $maxConf \leftarrow Conf(f)$;
        if $Conf(u) \leq \mu$ - $\alpha * \sigma$ **then**
            for *node* $v \in \mathcal{S}$ **do**
                if $Conf(v) > maxConf$ **then**
                    $maxConf \leftarrow Conf(v)$;
                    $l \leftarrow \arg\max_i(CV_v)$;
                end
            end
            $PredLabel[u] \leftarrow l$
        end
    end
end

of the graph and the labelling of the nodes. As a result, we blend both structural and feature properties. In the output intervention, we found the low confidence nodes that have structural similarity with the high confidence nodes and changed their labels to match the labels of the high confidence nodes. Thus, we can accurately classify the misclassified nodes in this manner.

Error while training node expansion: We have observed that the training nodes' set is increased in the input level intervention technique by applying a random walk or graph embedding technique to each set of starting seed nodes. In this section, we discuss the error that is introduced while expanding the training set. Let $\mathcal{S} = \bigcup_{c=1}^{k} \mathcal{S}_c$ be the final

set of elongated nodes in the training process, where $\mathcal{S}_c$ be the set of extended nodes for class c. Now the label of each individual node in $\mathcal{S}_c$ has been updated l_c. As a consequence of this, the labels of the nodes that belong to other classes will also be relabeled to reflect the class label l_c. Because of this, an error was produced in the extended set.

Assume that the group of nodes with the class label l_c is called V_c. Hence, it is simple to confirm that, $\bigcup_{c=1}^{k} V_c = V$ and $V_{c1} \cap V_{c2} = \Phi$ for $c1 \neq c2$ and $c1, c2 \in \{1, 2, ..k\}$. As a result, the number of nodes with the correct class label l_c in the set $\mathcal{S}_c$ will be:

$$\mathcal{N}_c = |\mathcal{S}_c \cap V_c|$$

Hence, the number of undesirable nodes (nodes from a different class) in $\mathcal{S}_c$ or the error related to $\mathcal{S}_c$

$$\mathcal{E}_c = |\mathcal{S}_c| - \mathcal{N}_c = |\mathcal{S}_c| - |\mathcal{S}_c \cap V_c|$$

As a result, the probability of error for $\mathcal{S}_c$ will be as follows:

$$\mathcal{P}_c = \frac{\mathcal{E}_c}{|\mathcal{S}_c|} = \frac{|\mathcal{S}_c| - |\mathcal{S}_c \cap V_c|}{|\mathcal{S}_c|} = \frac{t - |\mathcal{S}_c \cap V_c|}{t} = 1 - \frac{|\mathcal{S}_c \cap V_c|}{t} \quad \text{, as } |\mathcal{S}_c| = t, \forall c \in \{1, 2, ..., k\}$$

Moreover, the likelihood of error for $\mathcal{S}_c$ will be:

$$\mathcal{P} = \frac{\sum_{c=1}^{k} \mathcal{E}_c}{\sum_{c=1}^{k} |\mathcal{S}_c|} = \frac{\sum_{c=1}^{k} (|\mathcal{S}_c| - |\mathcal{S}_c \cap V_c|)}{\sum_{c=1}^{k} |\mathcal{S}_c|} = \frac{kt - \sum_{c=1}^{k} |\mathcal{S}_c \cap V_c|}{kt} = 1 - \frac{\sum_{c=1}^{k} |\mathcal{S}_c \cap V_c|}{kt}$$

$$\text{, as } |\mathcal{S}_c| = t, \forall c \in \{1, 2, ..., k\}$$

The accuracy with which the input level intervention technique gathers the nodes with the same label as the original seed nodes determines the value of the likelihood.

For instance, if we suppose that each class is equally distributed, and the input intervention method only captures the $\frac{3}{4}^{th}$ correct nodes (nodes with the same class label as c), then, $|\mathcal{S}_c \cap V_c| = \frac{3t}{4}$ implying $\mathcal{P}_c = 1 - \frac{3t}{4t} = 1 - \frac{3}{4} = \frac{1}{4}$.

In our study, we have taken t as, $t = b\frac{|I_c| \times \eta}{\sum_{x=1}^{k} |I_x|}$, where b is an integer. The set size t is changed as a result of tuning b. Too many nodes from different classes enter $\mathcal{S}_c$ when the value of t is large, increasing the error and lowering the GNN's test accuracy. On

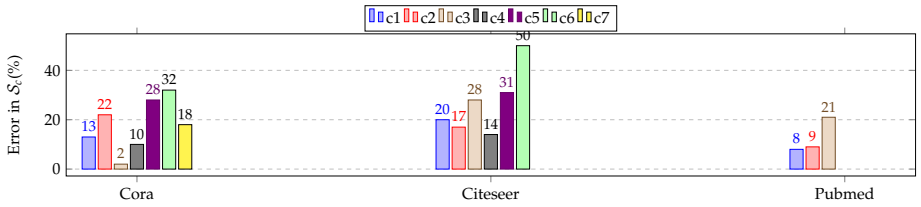
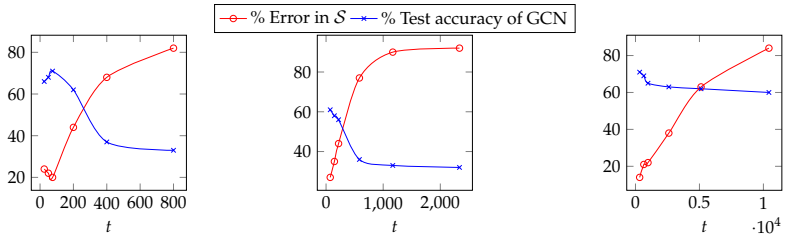


FIGURE 4.4: Percentage of error for each class while extending the training set using input level intervention technique. In the legend, c1, c2, ... c7 are the different classes ($b = 3$)

.



(a) Cora (b) Citeseer (c) Pubmed

FIGURE 4.5: Percentage of error in the extended set $\mathcal{S}$ and test accuracy of GCN when tuning the set size t

.

the other hand, when t is small, too few nodes from the actual class c enter $\mathcal{S}_c$, which lowers the GNN's test accuracy.

Figure 4.4 shows the experimental data capturing the likelihood of error for each class. As we adjust the set size t, we also show the plot of error associated with the completely expanded set $\mathcal{S}$ and the test accuracy of GCN in 4.5. In our research, we set b to 3.

4.4 Time complexity analysis

The time complexity of our algorithms will be provided in this section. We employed random walks in both the input and output intervention strategies. So, before determining the actual time complexity of both techniques, we first compute the random walk's time complexity.

Time requirement for PaRWalk: PaRWalk requires the computation of absorption probability matrix: $\mathcal{A} = (\Gamma + \alpha\Lambda)^{-1}$, where, Γ is the graph laplacian defined as $\Gamma = D - A$, $D = diag(d_1, d_2, ..., d_{|V|})$ is the degree matrix, d_i is the degree of node v_i and

A is the adjacency matrix of a network, $\alpha > 0$ is a scalar, $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is the known regularizer and $\lambda_i \geq 0$ is some arbitrary value. The computation of Γ has linear time complexity. Computation of $(\Gamma + \alpha\Lambda)$ has also linear time complexity. $\mathcal{A} = (\Gamma + \alpha\Lambda)^{-1}$ can be computed using Williams algorithm [408] in $O(|V|^{2.373})$ time.

Time requirement for DeepWalk: The time complexity of the DeepWalk [250] algorithm can be calculated as $O(\mathcal{W}_P|V|(\mathcal{W}_L + log(|V|)))$, where, $\mathcal{W}_P$ is the walk per node, $\mathcal{W}_L$ is the walk length, $|V|$ is the number of nodes. The term $log(|V|)$ comes from the time complexity of SkipGram [250] method.

4.4.1 Time complexity for input level intervention

It is evident from Algorithm 4 and 5 that the total time complexity of the input level intervention is the sum of time complexities of (i) the random walk, (ii) K-means and (iii) K-NN algorithms.

- **Random walk:** Already discussed.
- **K-means [409]:** Time complexity of K-means algorithm is: $O(t\delta mj)$, where j is the number of iterations for the convergence of the K-means algorithm, t is the number of samples, m is the number of samples, δ is the number of clusters.
- **K-NN [37]:** Time complexity of K-NN algorithm is: $O(|C_{avg}|m)$, where m is the dimension of each sample in cluster of average size C_{avg},

Time requirement of Algorithm 4: For L class labels, the time complexity would be:

$$T_{ip\_par} = O(|V|^{2.373}) + |L|(O(t\delta mj) + O(|C_{avg}|m))$$

$$= O(|V|^{2.373}) + (O(|L|t\delta mj) + O(|L||C_{avg}|m))$$

$$= O(|V|^{2.373}) + (O(|L|\frac{|I_c|b\eta}{\sum_{c=1}^{k}|I_c|}\delta mj) + O(|LC_{avg}|m)), since, t = \frac{|I_c|b\eta}{\sum_{c=1}^{k}|I_c|}$$

$$= O(|V|^{2.373}) + (O(|L|\frac{|I_c|b\eta}{2|L|}\delta mj) + O(|L||C_{avg}|m)) as, \sum_{c=1}^{k}|I_c| = 2L, \text{since we choose two nodes per class.}$$

$$= O(|V|^{2.373}) + (O(\frac{|I_c|b\eta}{2}\delta mj) + O(|L||C_{avg}|m))$$

$$= O(|V|^{2.373}) + (O(\frac{2b\eta}{2}\delta mj) + O(|L||C_{avg}|m))$$

$$= O(|V|^{2.373}) + (O(b\eta\delta mj) + O(|L||C_{avg}|m))$$

$$= O(|V|^{2.373}) + (O(\frac{|V|b\delta mj}{(d_{avg})^{\tau}}) + O(|L||C_{avg}|m)),$$

d_{avg} is the average degree of the graph G and τ is the number of layers of the GNN.

$$= O(|V|^{2.373}) + (O(\frac{|V|mj}{(d_{avg})^{\tau}}) + O(|L||C_{avg}|m)), \text{ for } b = 3\ , \delta = 2 \text{ are constant.}$$

Time requirement of Algorithm 5: Total time complexity for DeepWalk is:

$$T_{ip\_deep} = O(\mathcal{W}_P|V|(\mathcal{W}_L + log(|V|))) + (O(\frac{|V|mj}{(d_{avg})^{\tau}}) + O(|L||C_{avg}|m)).$$

4.4.2 Time complexity for Output level intervention:

For Algorithms 6 and 7, the time complexity can be calculated as the sum of the time taken by (i) the random walk, and (ii) relabel operations.

- **Random walk:** Already discussed.
- **Relabel operations:** Depends on the selected approach - PaRWalk or DeepWalk.

Time requirement of Algorithm 6: The time complexity of line 2 is O($|V|$). For each class, line 4 and 5 both require O(t) time + lines $6-19$ requires O($t(|H_{avg}| + d_{avg})$) time (considering average degree as d_{avg} and the average number of high confidence nodes as $|H_{avg}|$). So, the total time required is (using PaRWalk):

$$T_{op\_par} = O(|V|^{2.373}) + O(|V| + |L|(t + t(|H_{avg}| + d_{avg})))$$

$$= O(|V|^{2.373}) + O(|V| + |L|t(1 + |H_{avg}| + d_{avg})).$$

Since we have taken $t = \frac{|V|}{|L|}$, thus, time complexity of Algorithm 6 will be:

$$= O(|V|^{2.373}) + O(|V| + |L|\frac{|V|}{|L|}(1 + |H_{avg}| + d_{avg}))$$

$$= O(|V|^{2.373}) + O(|V| + |V|(1 + |H_{avg}| + d_{avg})).$$

Time requirement of Algorithm 7: Similarly, the total time complexity required is (using DeepWalk):

$$T_{op\_deep} = O(\mathcal{W}_P|V|(\mathcal{W}_L + log(|V|))) + O(|V| + |V|(1 + |H_{avg}| + d_{avg}))$$

Time requirement of both the input and output intervention: In this case, the random walk is computed only once, which is then used in both interventions. Therefore, the computation time in the case of PaRWalk for both interventions can be written as:

$$T_{ip\_op\_par} = O(|V|^{2.373}) + (O\left(\frac{|V|mj}{(d_{avg})^{\tau}}\right) + O(|L||C_{avg}|m)) + O(|V| + |V|(1 + |H_{avg}| + d_{avg}))$$

and in the case of DeepWalk:

$$T_{ip\_op\_deep} = O(\mathcal{W}_P|V|(\mathcal{W}_L + log(|V|))) + \left(O(\frac{|V|mj}{(d_{avg})^{\tau}}\right) + O(|L||C_{avg}|m)) + O(|V| + |V|(1 + |H_{avg}| + d_{avg}))$$

4.5 Datasets and baselines

4.5.1 Datasets

We use the datasets in Table 4.1 for our experiments. In the table, the first four datasets CiteSeer [406], Cora [406], Cora-ml [410] and PubMed [406], are citation networks. The nodes on each network represent the documents, while the edges signify the citation linkages. The node feature (a 0/1-valued vector) denotes the absence or presence of a certain word in the corresponding document. Classes, labelled starting from 0 to (number of subjects - 1) represent the subjects.

4.5.2 Baselines

We have used eight Graph Neural Networks and seven training set expansion methods as the baseline methods.

Baselines: Graph Neural Networks: Several Graph Neural Networks like GCN [43], GAT [44] etc are used here. The descriptions are given below:

Network	nodes	Edges	Features	Classes	Description
Citeseer [406]	3327	4732	3703	6	Citation networks extracted from the CiteSeer digital library. Nodes represent the documents, while the edges signify the citation linkages. The node feature (a 0/1-valued vector) denotes the absence/presence of a certain word in the corresponding document. classes, labelled starting from 0 to (number of subjects - 1) represent the subjects.
Cora [406]	2708	5429	1433	7	Cora citation networks. Description of nodes, edge, feature vector and class are same as Citeseer network.
Cora-ml [410]	2995	8416	2879	7	A sub-network of the Cora citation network containing the machine learning related papers only. Description of nodes, edge, feature vector and class are same as Citeseer network.
Pubmed [406]	19717	44338	500	3	Citation network corresponding to Pubmed dataset. Description of nodes, edge, feature vector and class are same as Citeseer network.
Amazon Photo [411]	7487	119043	745	8	Subset of the Amazon co-purchase network [412]. The goods are represented by nodes. If two goods (nodes) are purchased together then they are connected with an edge. Feature vector corresponding to the node represents absence/presence of certain words in the product review. classes repersent product categories.
Amazon Computers [411]	13381	245778	767	10	The goods are represented by nodes. If two goods (nodes) are purchased together then they are connected with an edge. Feature vector corresponding to the node represents absence/presence of certain words in the product review. Classes represent product categories.
ogbn-arxiv [413]	169,343	1,166,243	128	40	A citation network among all computer science arxiv papers. Node represents the paper and if a paper cites another paper, there is a link between them. Feature vector corresponding to the node represents absence/presence of certain words in the title and abstract. Class indicates the subject areas.
ogbn-products [413]	2,449,029	61,859,140	100	47	Amazon product co-purchasing network. Node represents products, Edge represents two products bought together. Feature vector represents the presence/absence of certain words in the product review. Class the product categories
ogbn-mag [413]	1,939,743	21,111,007	128	349	Nodes are either paper or author or institutions or fields of study. Edges between two nodes represents either "affiliations" or "citations" or "writer" or "has a topic of" relations. Class represents the venue.

TABLE 4.1: The test suite of real-world networks.

- **GCN** [43] It is a linear approximation of spectral graph convolution. For each node, the neighbours' information is aggregated and transformed to generate better node embeddings.

- **GAT** [44] Unlike GCN, where each neighbour of a node gets equal attention during aggregation, important neighbours get more weight (attention) during aggregation.
- **ChebyNet** [289] ChebyNet uses Chebyshev polynomials as its foundation to implement spectral convolution approximation. It applies Laplacian directly as a filter.
- **APPNP** [194] To create a more effective message propagation method, graph convolutional networks (GCN) and personalised PageRank are combined.
- **GPR-GNN** [195] It is an improved version of APPNP where generalized personalized page rank are used with GCN instead of personalized page rank.
- **JK-Nets** [297] Variable neighbourhood ranges for each node are aggregated to provide better structure-conscious representation.
- **GraphSAGE** [113] Instead of learning unique embeddings for every node, a function is developed that creates embeddings by selecting and combining information from a node's immediate neighbourhood.

Baselines: training set expansion: Also, we contrasted our intervention strategies with the following seven training set expansion strategies:

- **Random sampling** For each class, rather than using the random walk, the training sets are expanded using a random sampling of nodes from the set of all nodes $|V|$.
- **Co-training** [296] PaRWalk [243] is used to extend the training set. The confidence level of a node belonging to class c is calculated after the normalised absorption probability matrix A has been generated. The training set is expanded to include the top m nodes with the greatest confidence measure, labelled l_c;
- **Self-training** [296] The top m nodes with the highest confidence, as determined by the softmax scores, are added to the training nodes after a GNN has been trained with its initial training nodes for each class.
- **Co-Self-union** [296] The union of the extended training nodes returned by Co-training and Self-training.

- **Co-Self-intersection** [296] The intersection of the extended training nodes returned by Co-training and Self-training.
- **Training Node Augmentation (TNA)** [414] To increase the training set, intersections of nodes from many previously trained GNNs that have confidence above a certain threshold are gathered.
- **Hi-deg-path** A path of high-degree nodes is chosen for each class (like a meta-path in a heterogeneous graph [415]). We begin by looking for the training nodes' highest degree neighbours, selecting the highest degree neighbour, relabeling it with the starting node's label, adding it to the set, and continuing the procedure from this highest degree neighbour until the set of high-degree nodes stops growing.

4.5.3 Hyperparameter settings

We employed two training nodes from each class that were randomly chosen as part of our input level intervention. For testing purposes, each model is given a random selection of 1000 nodes (not including the training nodes). We did not use any validation set. For all the GNNs, we used two hidden layers. The learning rate and epochs are set to 0.01 and 200, respectively. For the smaller graphs (Citeseer, Cora, Cora-ml, Pubmed, Amazon Photo and Amazon Computers), we set walk length $\mathcal{W}_L$ to 50 and walks per node $\mathcal{W}_P$ to 30. For the larger graphs (ogbn-arxiv, ogbn-product and ogbn-mag), walk length $\mathcal{W}_L$ is set to 50 and walks per node $\mathcal{W}_P$ is set to 10.

4.6 Results

As previously said, our main objectives are to improve the accuracy of GNNs through the application of (a) input-level intervention and (b) output-level intervention. We employ the PaRWalk and DeepWalk algorithms for training set enlargement in the input level intervention. The output level intervention also employs these two techniques to identify the nodes that are most comparable to the high confidence nodes for each class. Thus, we can broadly categorize the results into two groups: results based on

	GNN model	Networks								
		Cora	Citeseer	Pubmed	Photo	Computer	Cora-ml	arxiv	product	mag
Baselines	GCN	0.45±0.05	0.33±0.06	0.52±0.05	0.32±0.09	0.21±0.09	0.36±0.04	0.12±0.04	0.15±0.03	0.02±0.04
	GAT	0.44±0.04	0.35±0.04	0.54±0.05	0.47±0.10	0.41±0.10	0.41±0.08	0.09±0.02	0.17±0.02	0.02±0.03
	ChebyNet	0.31±0.05	0.26±0.03	0.47±0.04	0.35±0.11	0.27±0.09	0.24±0.09	0.09±0.02	0.11±0.05	0.03±0.06
	GPR-GNN	0.48±0.06	0.38±0.05	0.62±0.08	0.54±0.13	0.44±0.17	0.52±0.09	0.14±0.02	0.21±0.02	0.05±0.03
	APPNP	0.50±0.05	0.36±0.04	0.59±0.04	0.23±0.11	0.10±0.10	0.36±0.10	0.14±0.03	0.20±0.04	0.05±0.03
	JKNet	0.35±0.04	0.27±0.05	0.57±0.04	0.17±0.10	0.32±0.10	0.23±0.11	0.08±0.03	0.12±0.02	0.02±0.03
	SAGE	0.37±0.05	0.30±0.05	0.56±0.05	0.42±0.11	0.32±0.10	0.31±0.11	0.04±0.03	0.11±0.06	0.02±0.07
Input intervention	ip+GCN	0.71±0.03	0.56±0.05	0.65±0.00	0.39±0.01	0.62±0.02	0.64±0.03	0.22±0.03	0.27±0.04	0.09±0.04
	ip+GAT	0.63±0.02	0.57±0.03	0.75±0.01	0.52±0.02	0.59±0.02	0.64±0.03	0.19±0.03	0.28±0.03	0.09±0.04
	ip+ChebyNet	0.68±0.01	0.62±0.03	0.85±0.00	0.50±0.03	0.51±0.01	0.60±0.03	0.20±0.02	0.20±0.07	0.11±0.05
	ip+GPR-GNN	0.68±0.01	0.60±0.03	0.83±0.01	0.57±0.03	0.50±0.08	0.67±0.03	0.24±0.01	0.24±0.07	0.08±0.04
	ip+APPNP	0.69±0.01	0.56±0.03	0.75±0.00	0.34±0.03	0.54±0.04	0.66±0.04	0.23±0.03	0.23±0.07	0.08±0.08
	ip+JKNet	0.52±0.01	0.43±0.03	0.75±0.01	0.27±0.01	0.48±0.03	0.45±0.02	0.19±0.07	0.18±0.08	0.04±0.07
	ip+SAGE	0.63±0.04	0.60±0.03	0.78±0.01	0.55±0.02	0.46±0.03	0.57±0.01	0.12±0.05	0.14±0.07	0.06±0.04
Output intervention	GCN+op	0.50±0.05	0.36±0.05	0.54±0.05	0.33±0.09	0.25±0.08	0.39±0.04	0.15±0.03	0.16±0.03	0.02±0.03
	GAT+op	0.46±0.04	0.38±0.05	0.53±0.05	0.48±0.07	0.49±0.08	0.45±0.07	0.12±0.04	0.17±0.04	0.03±0.03
	ChebyNet+op	0.35±0.04	0.27±0.03	0.48±0.04	0.36±0.10	0.28±0.09	0.27±0.08	0.11±0.04	0.12±0.04	0.03±0.03
	GPR-GNN+op	0.50±0.05	0.39±0.04	0.62±0.05	0.55±0.09	0.47±0.11	0.54±0.11	0.15±0.05	0.22±0.03	0.05±0.03
	APPNP+op	0.51±0.15	0.39±0.15	0.60±0.05	0.24±0.04	0.15±0.04	0.39±0.10	0.14±0.04	0.20±0.03	0.06±0.07
	JKNet+op	0.37±0.10	0.29±0.09	0.59±0.04	0.18±0.04	0.32±0.04	0.29±0.09	0.11±0.04	0.13±0.03	0.02±0.03
	SAGE+op	0.44±0.05	0.32±0.05	0.58±0.05	0.44±0.10	0.33±0.09	0.35±0.08	0.06±0.03	0.12±0.04	0.03±0.03
Both intervention	ip+GCN+op	0.74±0.01	0.63±0.02	0.67±0.01	0.40±0.04	0.64±0.05	0.68±0.02	0.24±0.04	0.28±0.03	0.09±0.04
	ip+GAT+op	0.65±0.01	0.63±0.03	0.77±0.02	0.54±0.06	0.63±0.08	0.67±0.01	0.21±0.05	0.28±0.03	0.09±0.04
	ip+ChebyNet+op	0.70±0.02	0.65±0.03	0.86±0.01	0.51±0.01	0.52±0.08	0.63±0.01	0.21±0.04	0.21±0.03	0.12±0.04
	ip+GPR-GNN+op	0.69±0.02	0.63±0.01	0.83±0.01	0.58±0.07	0.51±0.09	0.70±0.01	0.27±0.03	0.24±0.03	0.09±0.04
	ip+APPNP+op	0.71±0.01	0.59±0.01	0.78±0.01	0.35±0.06	0.53±0.08	0.69±0.01	0.24±0.04	0.24±0.03	0.09±0.04
	ip+JKNet+op	0.54±0.02	0.44±0.01	0.76±0.01	0.27±0.06	0.57±0.09	0.47±0.01	0.21±0.04	0.19±0.03	0.04±0.04
	ip+SAGE+op	0.65±0.01	0.65±0.01	0.78±0.01	0.43±0.06	0.56±0.10	0.60±0.01	0.13±0.04	0.16±0.03	0.09±0.06

TABLE 4.2: Accuracy of different baseline methods with and without applying interventions. ip: input intervention, op: output intervention. Here we use PaRWalk in the intervention technique. Photo: Amazon Photo and Computer: Amazon Computer. arxiv stands for ogbn-arxiv, product stands for ogbn-product and mag stands for ogbn-mag

PaRWalk and results based on DeepWalk. For each group, three subcategories of results are shown: (i) results based on only input interventions (ip + GNN variant), (ii) results based on only output interventions (GNN variant + op), and (iii) results based on both input and output interventions (ip + GNN variant + op).

4.6.1 Accuracies using PaRWalk

The outcomes of all three possible combinations of interventions, as well as the baselines, are presented in Table 4.2. As compared to the baselines, we can observe that the accuracy improved in every scenario, even with the only intervention being at the input level. When only output intervention was included, there was a little improvement in accuracy when compared to the baselines. When we include the input and output interventions together, as was to be expected, the combination yields the highest increases in comparison to the baselines.

	GNN model	Networks								
		Cora	Citeseer	Pubmed	Photo	Computer	Cora-ml	arxiv	product	mag
Baselines	GCN	0.45±0.05	0.33±0.06	0.52±0.05	0.32±0.09	0.21±0.09	0.36±0.04	0.12±0.04	0.15±0.03	0.02±0.04
	GAT	0.44±0.04	0.35±0.04	0.54±0.05	0.47±0.10	0.41±0.10	0.41±0.08	0.09±0.02	0.17±0.02	0.02±0.03
	ChebyNet	0.31±0.05	0.26±0.03	0.47±0.04	0.35±0.11	0.27±0.09	0.24±0.09	0.09±0.02	0.11±0.05	0.03±0.06
	GPR-GNN	0.48±0.06	0.38±0.05	0.62±0.08	0.54±0.13	0.44±0.17	0.52±0.09	0.14±0.02	0.21±0.02	0.05±0.03
	APPNP	0.50±0.05	0.36±0.04	0.59±0.04	0.23±0.11	0.10±0.10	0.36±0.10	0.14±0.03	0.20±0.04	0.05±0.03
	JKNet	0.35±0.04	0.27±0.05	0.57±0.04	0.17±0.10	0.32±0.10	0.23±0.11	0.08±0.03	0.12±0.02	0.02±0.03
	SAGE	0.37±0.05	0.30±0.05	0.56±0.05	0.42±0.11	0.32±0.10	0.31±0.11	0.04±0.03	0.11±0.06	0.02±0.07
Input intervention	ip+GCN	0.69±0.02	0.54±0.05	0.64±0.00	0.38±0.01	0.63±0.02	0.62±0.03	0.21±0.04	0.24±0.03	0.07±0.05
	ip+GAT	0.62±0.03	0.53±0.03	0.74±0.02	0.53±0.03	0.61±0.02	0.61±0.03	0.20±0.03	0.23±0.03	0.07±0.08
	ip+ChebyNet	0.67±0.01	0.61±0.03	0.85±0.00	0.50±0.03	0.51±0.01	0.58±0.03	0.19±0.03	0.18±0.05	0.09±0.04
	ip+GPR-GNN	0.68±0.01	0.60±0.03	0.81±0.01	0.60±0.03	0.50±0.08	0.66±0.03	0.23±0.03	0.24±0.05	0.07±0.04
	ip+APPNP	0.69±0.01	0.57±0.03	0.74±0.02	0.32±0.03	0.55±0.04	0.65±0.04	0.23±0.03	0.24±0.07	0.08±0.07
	ip+JKNet	0.53±0.01	0.41±0.03	0.75±0.01	0.28±0.01	0.47±0.03	0.43±0.02	0.17±0.03	0.18±0.06	0.06±0.06
	ip+SAGE	0.62±0.04	0.58±0.02	0.78±0.01	0.54±0.05	0.46±0.09	0.58±0.04	0.12±0.02	0.15±0.06	0.06±0.04
Output intervention	GCN+op	0.49±0.05	0.36±0.05	0.55±0.06	0.33±0.10	0.23±0.09	0.38±0.05	0.13±0.03	0.16±0.03	0.02±0.03
	GAT+op	0.45±0.03	0.39±0.04	0.54±0.06	0.48±0.10	0.43±0.12	0.45±0.07	0.11±0.04	0.17±0.05	0.03±0.08
	ChebyNet+op	0.35±0.04	0.27±0.03	0.48±0.04	0.38±0.10	0.28±0.09	0.27±0.08	0.10±0.06	0.13±0.03	0.04±0.05
	GPR-GNN+op	0.50±0.05	0.40±0.04	0.62±0.05	0.55±0.09	0.47±0.11	0.53±0.11	0.16±0.04	0.22±0.02	0.05±0.03
	APPNP+op	0.54±0.15	0.37±0.15	0.60±0.05	0.24±0.04	0.13±0.04	0.39±0.10	0.15±0.03	0.22±0.06	0.05±0.03
	JKNet+op	0.37±0.10	0.29±0.09	0.59±0.04	0.18±0.04	0.32±0.04	0.29±0.09	0.12±0.05	0.13±0.06	0.03±0.04
	SAGE+op	0.42±0.05	0.32±0.05	0.58±0.05	0.44±0.10	0.33±0.09	0.35±0.08	0.05±0.07	0.13±0.06	0.03±0.04
Both intervention	ip+GCN+op	0.71±0.03	0.57±0.02	0.66±0.03	0.40±0.04	0.63±0.05	0.66±0.05	0.23±0.03	0.25±0.05	0.07±0.04
	ip+GAT+op	0.64±0.01	0.56±0.03	0.77±0.03	0.54±0.07	0.62±0.10	0.64±0.02	0.22±0.05	0.23±0.04	0.08±0.03
	ip+ChebyNet+op	0.68±0.03	0.63±0.04	0.85±0.01	0.51±0.01	0.52±0.08	0.62±0.01	0.22±0.04	0.18±0.06	0.09±0.03
	ip+GPR-GNN+op	0.70±0.03	0.62±0.03	0.83±0.01	0.61±0.06	0.52±0.06	0.69±0.03	0.25±0.03	0.25±0.05	0.09±0.04
	ip+APPNP+op	0.70±0.02	0.60±0.03	0.74±0.01	0.33±0.06	0.56±0.07	0.68±0.01	0.25±0.03	0.24±0.06	0.08±0.04
	ip+JKNet+op	0.56±0.04	0.42±0.03	0.76±0.01	0.29±0.06	0.57±0.09	0.44±0.01	0.19±0.07	0.19±0.06	0.07±0.05
	ip+SAGE+op	0.64±0.02	0.62±0.02	0.78±0.01	0.55±0.07	0.47±0.09	0.61±0.02	0.14±0.03	0.16±0.03	0.08±0.05

TABLE 4.3: Accuracy of different baseline methods with and without applying interventions. ip: input intervention, op: output intervention. Here we use DeepWalk in our intervention techniques. Photo: Amazon Photo and Computer: Amazon Computer. arxiv stands for ogbn-arxiv, product stands for ogbn-product and mag stands for ogbn-mag

4.6.2 Accuracies using DeepWalk

When using DeepWalk as the training node expansion method, Table 4.3 shows the results of input-only interventions, output-only interventions, and input-and-output interventions, respectively. We can see that using the intervention methods enhances the accuracy in comparison to the baseline method, as was the case with the previous results (PaRWalk). The highest gain using the input intervention method is: (67-31)/31% = 116% (for the cora network and ChebyNet model). For output intervention, the highest gain is: (29-23)/23% = 26% (for cora-ml network and JKNet model). Finally, for both interventions, the highest gain is: (68-31)/31% = 119% (for the Cora network and ChebyNet model).

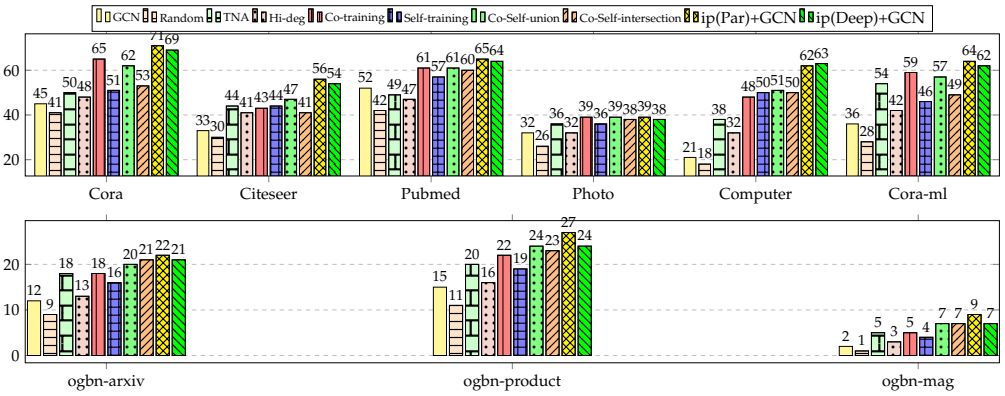


FIGURE 4.6: Comparison of % accuracy (Y-axis) for different training set expansion methods: Training Node Augmentation (TNA), random sampling (Random), High-deg-path (Hi-deg)
, Co-training, Self-training, Co-Self-union, Co-Self-intersection and our proposed input interventions. ip (Par): input intervention with PaRWalk, ip (Deep): input intervention with DeepWalk, Photo: Amazon Photo and Computer: Amazon Computer.

4.6.3 Comparison with training set expansion baselines

As already mentioned, we compared our method with seven training set expansion methods: Random sampling, Co-training, Self-training, Co-self-union, Co-self-intersection, Training Node Augmentation and Hi-deg-path. We used GCN as the Graph Neural Network tool to verify their accuracy. For each network, the accuracy for GCN is shown by the leftmost bar (base model). The following seven bars show seven baselines, and the final two bars show the input intervention techniques, which are PaRWalk and DeepWalk, respectively.

4.6.4 Significance test on output level intervention

We report the p-values for the Mann-Whitney significance test in Tables 4.4 and 4.5 in order to confirm whether the outcomes of the output level interventions are statistically significant. In every instance, $p < 0.05$ shows that the output intervention does in fact produce outcomes that differ significantly from those produced by the base GNN variant.

GNN vs GNN+op	Networks								
	cora	citeseer	pubmed	photo	computer	cora-ml	arxiv	product	mag
GCN vs GCN+op	0.002	0.001	0.002	0.011	0.012	0.004	0.002	0.01	0.02
GAT vs GAT+op	0.001	0.005	0.0001	0.001	0.011	0.004	0.004	0.02	0.04
ChebyNet vs ChebyNet+op	0.002	0.004	0.003	0.002	0.007	0.002	0.003	0.01	0.03
GPRGNN vs GPRGNN+op	0.001	0.03	0.002	0.005	0.02	0.004	0.002	0.01	0.04
APPNP vs APPNP+op	0.002	0.005	0.009	0.003	0.01	0.01	0.004	0.02	0.02
JKNet vs JKNet+op	0.01	0.002	0.0003	0.010	0.034	0.003	0.002	0.01	0.04
SAGE vs SAGE+op	0.0006	0.0007	0.001	0.0001	0.008	0.003	0.001	0.01	0.04

TABLE 4.4: Significance tests with output interventions plugged in. ParWalk is used in the intervention methods. arxiv stands for ogbn-arxiv, product stands for ogbn-product and mag stands for ogbn-mag.

GNN vs GNN+op	Networks								
	cora	citeseer	pubmed	photo	computer	cora-ml	arxiv	product	mag
GCN vs GCN+op	0.003	0.001	0.001	0.013	0.02	0.001	0.002	0.02	0.02
GAT vs GAT+op	0.003	0.002	0.001	0.002	0.01	0.005	0.004	0.01	0.02
ChebyNet vs ChebyNet+op	0.001	0.002	0.003	0.01	0.02	0.012	0.002	0.02	0.01
GPRGNN vs GPRGNN+op	0.004	0.002	0.002	0.001	0.02	0.002	0.001	0.02	0.03
APPNP vs APPNP+op	0.001	0.005	0.003	0.002	0.02	0.012	0.006	0.01	0.03
JKNet vs JKNet+op	0.001	0.004	0.004	0.002	0.03	0.001	0.002	0.02	0.03
SAGE vs SAGE+op	0.002	0.0008	0.004	0.001	0.08	0.001	0.002	0.02	0.02

TABLE 4.5: Significance tests with output interventions plugged in. DeepWalk is used in the intervention methods. arxiv stands for ogbn-arxiv, product stands for ogbn-product and mag stands for ogbn-mag.

4.6.5 Time comparisons

Tables 4.6 and 4.7 compare the execution times of all the various baseline techniques while employing PaRWalk and DeepWalk, respectively, in the intervention pipeline. Since more steps are necessary, using the intervention methods will inevitably take longer. In addition to the various GNN baselines, it is clear from the tables that our intervention method requires more time if the network is large (like a computer). The calculation of the absorption probability determines the actual time requirement for PaRWalk, whereas for DeepWalk, the actual time requirement is primarily determined by two parameters: the length of the random walk and the number of times the random walk is necessary.

4.6.6 Effects on increasing the number of hidden layers

This section explains how the GNN's accuracy in the input intervention is impacted by the number of layers. The results of the GNNs integrated with only input intervention on the Cora network are displayed in the Figs. 4.7 and 4.8 (for the other networks, the

	GNN model	Networks								
		cora	citeseer	pubmed	photo	computer	cora-ml	arxiv	product	mag
Baselines	GCN	3s	5s	24s	36s	73s	5s	88s	32015s	1608s
	GAT	35s	72s	207s	300s	601s	57s	626s	35393s	2551s
	ChebyNet	15s	44s	51s	117s	248s	38s	261s	33265s	1987s
	GPRGNN	9s	18s	38s	67s	147s	17s	166s	32965s	1843s
	APPNP	9s	17s	37s	66s	147s	16s	163s	32958s	1839s
	JKNet	8s	11s	64s	40s	85s	10s	93s	32329s	1679s
	SAGE	9s	26s	46s	98s	187s	23s	253s	32768s	1983s
	Random-sample	4s	6s	26s	38s	78s	6s	90s	32037s	1629s
	TNA	8s	14s	50s	62s	156s	11s	95s	32048s	1641s
	Hi-deg-Path	4s	8s	31s	46s	92s	9s	93s	32037s	1631s
	Co-training	10s	13s	241s	131s	756s	11s	345s	33135s	2369s
	Self-training	9s	18s	161s	86s	547s	10s	324s	33124s	2213s
	Co-Self-union	15s	27s	347s	185s	922s	18s	548s	51567s	3516s
	Co-Self-intersection	16s	27s	347s	185s	922s	18s	537s	51769s	3532s
Input intervention	ip+GCN	11s	15s	248s	160s	880s	13s	940s	47763s	37593s
	ip+GAT	44s	75s	432s	424s	1416s	71s	1788s	51194s	38566s
	ip+ChebyNet	22s	56s	278s	241s	967s	47s	1164s	49094s	37989s
	ip+GPRGNN	16s	27s	261s	193s	957s	30s	1025s	48794s	37856s
	ip+APPNP	18s	29s	259s	191s	956s	27s	1022s	48781s	37849s
	ip+JKNet	17s	18s	283s	167s	898s	19s	947s	48091s	37687s
	ip+SAGE	19s	37s	269s	225s	1003s	33s	1116s	48534s	37973s
Output intervention	GCN+op	5s	9s	240s	144s	792s	9s	731s	37266s	5127s
	GAT+op	36s	74s	422s	408s	1320s	66s	1582s	40649s	6079s
	ChebyNet+op	18s	45s	270s	227s	1060s	43s	983s	38526s	5506s
	GPRGNN+op	10s	20s	252s	188s	864s	26s	834s	38253s	5373s
	APPNP+op	10s	18s	250s	186s	863s	25s	830s	38244s	5368s
	JKNet+op	9s	12s	274s	161s	804s	15s	757s	37562s	5207s
	SAGE+op	12s	28s	263s	209s	919s	32s	996s	39037s	5723s
Both intervention	ip+GCN+op	12s	19s	468s	270s	1604s	18s	1586s	52989s	41126s
	ip+GAT+op	46s	77s	651s	534s	2135s	81s	2748s	56486s	42122s
	ip+ChebyNet+op	23s	59s	497s	351s	1785s	52s	1889s	54361s	41566s
	ip+GPRGNN+op	19s	33s	477s	314s	1676s	40s	1695s	54082s	41427s
	ip+APPNP+op	19s	30s	474s	312s	1673s	39s	1693s	54072s	41405s
	ip+JKNet+op	20s	23s	495s	288s	1620s	24s	1613s	53378s	41233s
	ip+SAGE+op	22s	40s	481s	343s	1749s	43s	1868s	54785s	41798s

TABLE 4.6: Time requirement for different methods. PaRWalk is used in the intervention methods.

trends are similar). We can see that, with the exception of GPRGNN and APPNP, the accuracy is not significantly impacted by the number of layers. When moving from layer 3 to layer 4, the accuracy of GPRGNN and APPNP suddenly decreases.

4.6.7 Sensitivity analysis

Certain parameters in both the PaRWalk and DeepWalk may have an impact on how accurate our results are. In this section, we examine the effects of changing one parameter while holding the others constant. We conducted this study for the Cora, Citeseer, and Pubmed networks (results for other networks give the same pattern but are not

	GNN model	Networks								
		cora	citeseer	pubmed	photo	computer	cora-ml	arxiv	product	mag
Baselines	GCN	3s	5s	24s	36s	73s	5s	88s	32015s	1608s
	GAT	35s	72s	207s	300s	601s	57s	626s	35393s	2551s
	ChebyNet	15s	44s	51s	117s	248s	38s	261s	33265s	1987s
	GPRGNN	9s	18s	38s	67s	147s	17s	166s	32965s	1843s
	APPNP	9s	17s	37s	66s	147s	16s	163s	32958s	1839s
	JKNet	8s	11s	64s	40s	85s	10s	93s	32329s	1679s
	SAGE	9s	26s	46s	98s	187s	23s	253s	32768s	1983s
	Random-sample	4s	6s	26s	38s	78s	6s	90s	32037s	1629s
	TNA	8s	14s	50s	62s	156s	11s	95s	32048s	1641s
	Hi-deg-Path	4s	8s	31s	46s	92s	9s	93s	32037s	1631s
	Co-training	10s	13s	241s	131s	756s	11s	345s	33135s	2369s
	Self-training	9s	18s	161s	86s	547s	10s	324s	33124s	2213s
	Co-Self-union	15s	27s	347s	185s	922s	18s	548s	51567s	3516s
	Co-Self-intersection	16s	27s	347s	185s	922s	18s	537s	51769s	3532s
Input intervention	ip+GCN	8s	11s	213s	138s	683s	11s	1123s	49567s	38251s
	ip+GAT	41s	78s	398s	401s	1201s	68s	1969s	52963s	31998s
	ip+ChebyNet	21s	51s	246s	192s	854s	42s	1347s	50823s	38639s
	ip+GPRGNN	14s	24s	229s	171s	776s	26s	1208s	50492s	38486s
	ip+APPNP	15s	23s	227s	169s	774s	26s	1205s	50483s	38482s
	ip+JKNet	13s	17s	254s	143s	702s	13s	1128s	49862s	38327s
	ip+SAGE	16s	33s	246s	206s	805s	32s	1293s	50331s	38632s
Output intervention	GCN+op	5s	8s	202s	123s	608s	8s	921s	38127s	5632s
	GAT+op	37s	74s	389s	385s	1133s	54s	1764s	41625s	6591s
	ChebyNet+op	17s	47s	237s	174s	781s	41s	1144s	39395s	6042s
	GPRGNN+op	10s	19s	217s	156s	699s	20s	1004s	39097s	5883s
	APPNP+op	10s	19s	217s	154s	699s	20s	1002s	39085s	5875s
	JKNet+op	9s	13s	245s	127s	625s	12s	922s	38484s	5021s
	SAGE+op	13s	30s	227s	193s	725s	30s	1093s	38880s	6018s
Both intervention	ip+GCN+op	12s	16s	391s	230s	1221s	15s	1956s	55682s	42283s
	ip+GAT+op	45s	80s	583s	486s	1743s	69s	3107s	59203s	36025s
	ip+ChebyNet+op	25s	54s	435s	248s	1395s	52s	2230s	56967s	42698s
	ip+GPRGNN+op	18s	28s	411s	293s	1346s	30s	2048s	56582s	42539s
	ip+APPNP+op	18s	28s	408s	291s	1341s	30s	2044s	56571s	42522s
	ip+JKNet+op	17s	22s	439s	231s	1249s	17s	1959s	56029s	41713s
	ip+SAGE+op	23s	38s	431s	307s	1361s	41s	2153s	56461s	42683s

TABLE 4.7: Time requirement for different input intervention methods. DeepWalk is used in the intervention methods.

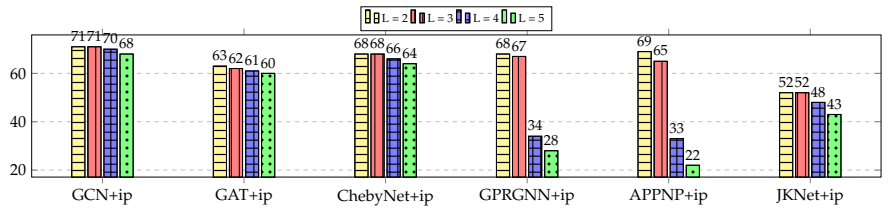


FIGURE 4.7: Effects on the % accuracy (Y-axis) for different GNN models while varying the number of layers in cora network. L: Number of hidden layers in the GNN. PaRWalk is used for input intervention.

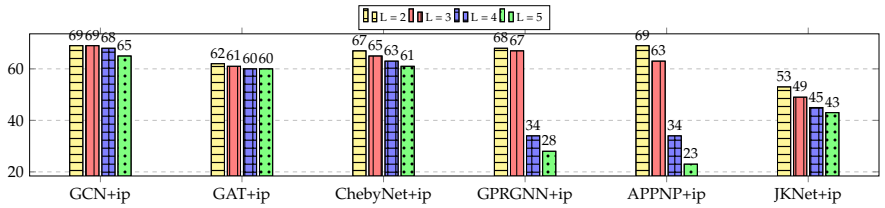


FIGURE 4.8: Effects on the % accuracy (Y-axis) for different GNN models while varying the number of layers in cora network. L: Number of hidden layers in the GNN. DeepWalk is used for input intervention.

shown for brevity). We chose the GCN model specifically because it is the fastest of all, and we only look into the input level intervention for both of these random walk methods.

Tuning PaRWalk's parameters The absorption probability α is the only parameter in ParWalk. We adjust it to 10 values ranging from 10^{-6} to 10^{-1} and plot the outcomes in Figure 4.9. As can be seen, the accuracy barely changes as α is raised from 10^{-6} to 10^{-1}. According to this finding, the GCN's accuracy is not overly sensitive to absorption probability.

Tuning DeepWalk's parameters: The DeepWalk method has a set of crucial parameters, including (i) window size ($\mathcal{W}_S$), (ii) walks per node ($\mathcal{W}_P$), and (iii) walk length ($\mathcal{W}_L$). The accuracy of the GCN is shown in Figure 4.10 as various tuning parameters are applied. Figure 4.10(a) shows that as we gradually increase the window size while keeping walks per node at 1 and walk length at 2, the accuracy gradually improves. The accuracy in Figure 4.10(b) first abruptly rises and then stabilises as the number of walks per node rises from 1, while the other two parameters are held constant at 2. The trends in Figure 4.10(c) are similar to those in Figure 4.10(b) (other parameters are kept at 2 and 1 for window size and walks per node, respectively). The accuracy quickly stabilises for all the parameters, indicating that the random walk has converged, and any modification to the node embeddings after this point is extremely unlikely.

4.7 Discussion

In this chapter, we proposed two approaches: input-level and output-level interventions to improve the classification accuracy of GNN. By carefully choosing nodes from

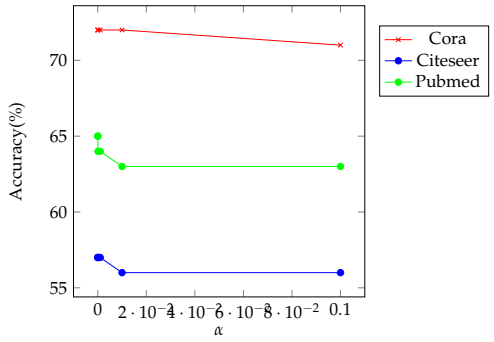


FIGURE 4.9: Sensitivity analysis on ParWalk by tuning various parameters while performing input intervention. The absorption probability α is varied from 0.1 to 10^{-6}

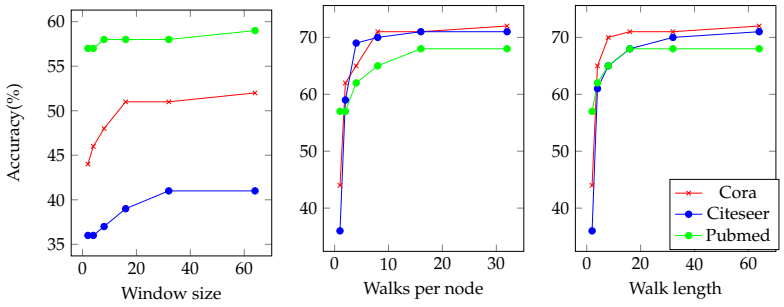


FIGURE 4.10: Sensitivity analysis on DeepWalk by tuning various parameters while performing input intervention. (a) Varying window size, while fixing walks per node = 1 and walk length = 2. (b) Varying walks per node, while fixing window size = 2 and walk length = 2. (c) Varying Walk length, while fixing window size = 2 and walks per nodes = 1

various non-contiguous subgraphs, we add a set of additional training nodes of the same class to the training set in the input intervention. To expand the training sets, we employ different iterations of random walks or node embedding methods. To increase the diversity of the nodes, we use K-means followed by K-NN. In the output intervention, we apply a similar random walk or node embedding technique to find the incorrectly labelled nodes and reclassify them using the nodes' level of confidence. We have used two methods (PaRWalk and DeepWalk) to agnostically expand the training set in the input-level intervention as well as relabel the possibly incorrectly predicted nodes of GNN in the output-level intervention technique.

When looking at the two different strategies, the intervention at the input level works significantly better than the intervention at the output level. However, when employed together, as one would anticipate, the combined procedures produce the best outcomes.

4.8 Conclusion

A graph neural network is a useful semi-supervised graph representational learning tool. However, if it is trained with a small number of training nodes, its representational capacity goes down, which affects the downstream tasks (in our case, the node classification). The problem with training nodes is that sometimes they are very difficult to obtain, and if we directly increase the number of training nodes, then it would defeat the purpose of the semi-supervised approach. In this chapter, we address this issue by proposing two novel methods: input-level intervention and output-level intervention. In the input-level intervention, we discuss how the small number of training nodes can be agnostically extended using random walks and machine learning tools. In the output-level intervention, we discuss how the random walk can be used to post-process the GNN's output. The promising results obtained in this chapter makes significant contributions and provides useful insights to the graph based deep learning field.

Part - III

Application of Complex Network and its Representation in Neuroscience

Publications:

- Anjan Chowdhury, Rajdeep Chatterjee, Geetanjali Aich, and Kuntal Ghosh. "ADHDNet: A DNN based Framework for Efficient ADHD Detection from fMRI Dataset." In 9th International Conference on Pattern Recognition and Machine Intelligence (PReMI'21), Lecture Notes in Computer Science. Springer, A. Ghosh et al. (Eds.): Volume 13102, Chapter 15, https://link.springer.com/book/9783031126994, 2024.
- Anjan Chowdhury, Swarup Chattopadhyay, and Kuntal Ghosh. "Analyzing the Progression of Alzheimer's Disease in Human Brain Networks." In Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'23), pp. 415-418. https://doi.org/10.1145/3625007.3627496, 2023.
- Snigdha Agarwal, Adarsh Raj, Anjan Chowdhury, Rajdeep Chatterjee, Geetanjali Aich, and Kuntal Ghosh. "Investigating the Impact of Standard Brain Atlases and Connectivity Measures on the Accuracy of ADHD Detection from fMRI Data Using Deep Learning.", Multimedia Tools and Applications, Volume 83, pp. 67023–67057 https://doi.org/10.1007/s11042-023-17962-7, 2024.
- Anjan Chowdhury, Swarup Chattopadhyay, Kuntal Ghosh, and Gautam Das. "NeuroANATOP: An Effective Tool to Assess the Progression of Alzheimer's disease in Human Brain Networks", Proceedings of 2023 IEEE International Conference On Knowledge Graph (ICKG), pp. 108-116, https://doi.org/10.1109/ICKG59574.2023.00019, 2023.

Chapter 5

Diagnosing ADHD disease using complex network and its representation

Chapter Summary: The functional magnetic resonance imaging (fMRI) data is used to capture the activity of each brain region. The regions inside a human brain and the relationships among them can be represented as a brain connectivity matrix, and this matrix can further be modelled as a complex network. This chapter focuses on the use of complex networks and their representation in the field of neuroscience, making a valuable addition to this domain.

Here, we study how the complex network and its representation can be used to diagnose a neuro-developmental disease called Attention Deficit Hyperactivity Disorder (ADHD). We have used various one-dimensional neural network models in our study. We have shown that the choice of a particular atlas and connectivity matrix can affect the classification results and, hence, the diagnosis.

5.1 Introduction

Inattention, hyperactivity, and impulsivity are hallmarks of the prevalent brain disorder known as Attention Deficit Hyperactivity Disorder (ADHD) [416]. ADHD is typically present in children and often lasts into adulthood. According to a previous study, 30 to 50 percent of children with ADHD continue to experience symptoms as adults [417]. It is essential to diagnose this condition as precisely as possible in order to provide paediatric patients with timely treatment.

The two primary classifications of the diagnosis of ADHD are symptomatological [418] and neurobiological [419] diagnoses. In symptomatological diagnosis, patients are continuously monitored, and some ratings are given based on different Hamilton scales. Several technologies are used in neurobiological diagnostics to collect brain data and identify diseases, including electroencephalography (EEG), positron emission tomography (PET), magnetic resonance imaging (MRI), and functional magnetic resonance imaging (fMRI) [420, 421]. The use of fMRI is seen as one of the most appropriate methods for this diagnosis because it is non-invasive and has high spatial resolution. Another factor contributing to its popularity is the fact that it uses the Blood Oxygen Dependent (BOLD) signal [422], which can be used to detect functional connectivity between different brain regions and has been shown to produce better results than other bio-signals [423, 424] like Amplitude of Low Frequency Fluctuation, Regional Homogeneity, etc. The result of fMRI is a series of 3D images taken over time, as opposed to MRI data, which only produces a single 3D image. fMRI images are thus 4D data.

In this study, we use fMRI data to differentiate between controlled or cognitively normal (CN) and ADHD using a neurobiological diagnosis. Here, we suggest a novel method for diagnosing ADHD based on complex network and its representation.

We use functional connectivity between different brain regions to build a graph[1]. We then develop a feature vector representation for that graph using various fundamental properties of the graph, such as degree centrality [69], clustering coefficient [425], etc. These constructed feature vectors are then fed into 1D-neural networks, such as 1D-CNN [426], LSTM [427], and others.

[1] In appendix A, we discussed another way to use the functional connectivity matrix. We visualize the connectivity matrix as an image and applied various 2D-neural network models to classify ADHD brain networks with the controlled brain networks

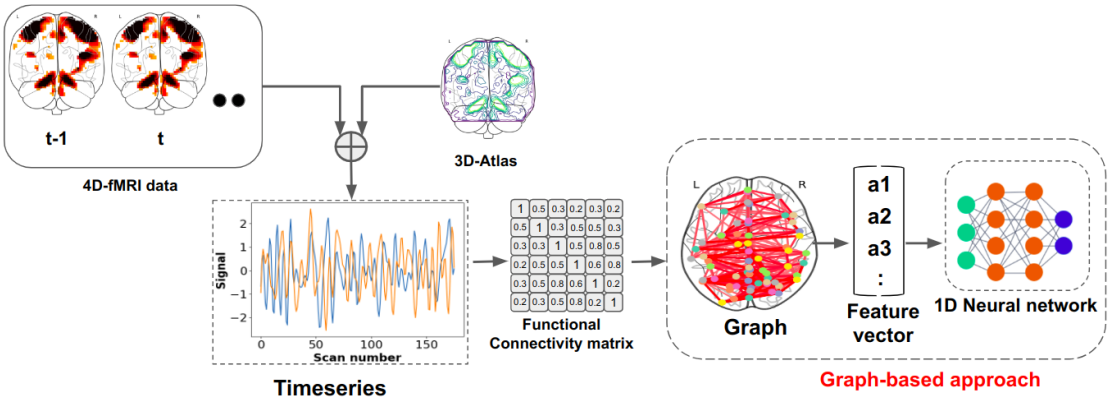


FIGURE 5.1: Workflow Diagram.

We have made use of fMRI data from the well-known ADHD-200 dataset that is publicly available [428]. Data collection was followed by preprocessing for denoising. In order to extract BOLD (timeseries) signals from various locations that are mapped by that atlas, the preprocessed data is then mapped using the atlas. The time series gathered are put to use to create functional connections (like correlations) between distinct locations. Fig. 5.1 presents a thorough workflow.

5.2 Proposed Strategies

In this section, we discuss how the preprocessed ADHD data is used to generate the features needed to effectively diagnose ADHD using some deep learning frameworks.

5.2.1 Time Series Generation

We took the preprocessed fMRI data and retrieved the BOLD signals (time series information) from a number of different brain areas. The data obtained from the fMRI scan do not include any information that pertains to the different brain regions. So, in order to extract the time series from a number of different regions of the brain, it is necessary to mask the fMRI data using a standard atlas. An atlas is a three-dimensional image that has been labelled, and each of its voxels has been assigned a coordinate. Each marker corresponds to a different region of the brain. In order to extract the time-series

signal from a specific location, signals from all of the voxels in that region are gathered, and then the average of those signals is used to generate a single signal from the collected signals.

5.2.1.1 Generating Connectivity Matrix

The next step is to associate these time series across various parts of the brain using various connectivity measures (like Pearson correlation, partial correlation, etc.) to create a connectivity matrix after the time series data for each region of the brain has been generated. In the field of neuroscience, a sort of representation known as a brain connectivity matrix is used to depict the connections or interactions that take place between various parts of the brain. The connectivity matrix contains a listing of all possible pairs of brain areas, with a numerical value denoting the degree to which a link exists between each such pair.

5.2.2 Diagnosing ADHD Using Deep Learning

Regardless of their direct connections, the brain areas' functional linkages are included in the connectivity matrix created in the previous section. This connectivity matrix was put to use in this following way:

- Transform the functional connectivity matrix into a graph, then draw out some key properties from the graph, and then feed these attributes as a feature vector into the neural network model.

In this section, we will go into further detail regarding both of them.

5.2.2.1 Using complex network representation for diagnosis the ADHD

Since the generated connectivity matrix is symmetric, we can use it as a weighted network or graph. Thus, the connectivity matrix is the adjacency matrix.

$$A = (a_{ij}) \in \mathbb{R}^{n \times n}$$

where, $a_{ij} \in \mathbb{R}$ is the weight of the edge representing the connectivity of regions i and j. Following this network generation, a number of significant attributes are calculated, including degree centrality, closeness centrality, betweenness centrality, eigenvector centrality, page rank, clustering coefficient, transitivity, local efficiency, and global efficiency.The issue now is that, aside from global efficiency, all of these properties are node-based, meaning that we are receiving features for each node. Thus, we need to convert the node-level information into network- or graph-level information in order to categorise the graphs based on this information since our ultimate goal is to distinguish the ADHD fMRI data (or network) from the controlled fMRI data (or network). For that, we take the average of the feature values. Formally, if $F \in \mathbb{R}^t$ represents the feature vector of graph G, then,

$$F = \frac{1}{|V|} \sum_{i=1}^{|V|} f_i$$

$f_i = (f_{i1}, f_{i2}, ..., f_{i9}) \in \mathbb{R}^9$ is the feature vector of node i, where, f_{i1} is the degree centrality for node i, f_{i2} is the closeness centrality for node i and so on... For the task of classifying graphs (networks), these feature vectors are fed into a one-dimensional (1D) neural network.

5.3 Data Collection and Prepossessing

5.3.1 Data collection

The present investigation employs a dataset sourced from the Neuro Bureau ADHD-200 [428, 429] competition (conducted in 2011), which comprises resting-state fMRI (rs-fMRI) data, anatomical images, and phenotypic information for each participant. This dataset is a constituent of the 1000 Functional Connectomes project [430]. Samples were obtained from the NeuroImaging Tools and Resources Collaboratory (NITRC), which were donated by the Kennedy Krieger Institute (KKI), New York University Medical Centre (NYU), NeuroIMAGE Sample (NI), and Peking University (PU). The collected samples included both male and female subjects, and their corresponding information is presented in Table 5.1. The fMRIPrep software was utilised to preprocess all of the samples, as stated in the reference [431].

	KKI	NI	NYU	Peking
Number of subjects	74	27	181	70
ADHD/Controlled	21/53	13/14	106/75	30/40

TABLE 5.1: Class distributions for all the four datasets obtained from ADHD-200 Challenge [428]

5.3.2 Data Preprocessing

In the data processing step, we have used fMRIPrep [431] to preprocess the data. It is a Python-based, robust pre-processing tool. The raw fMRI images are converted into a Brain Imaging Data Structure (BIDS) [432] format, and then the fMRIPrep is applied, which uses a set of well-known softwares like FSL [433], AFNI [434], Free-surfer [435] and ANTs [436] . The pre-processing steps we have applied with the help of fMRIPrep are: (i) remove the first few brighter images; (ii) head motion correction [162], (iii) slice time correction [164], (iv) distortion correction [166], (v) EPI to template space registration [168]. All the details of these steps can be found in the literature survey **??**.

5.4 Baselines

We used three deep-learning-based models as baselines: FCNet [355], 3D-CNN [357] and Deep-fMRI [437].

- **FCNet**: FCNet directly captures functional connectivity from fMRI time-series signals in raw form. The FCNet is made up of a fully connected network that calculates the similarity between the extracted features in a Siamese architecture and a convolutional neural network that extracts features from time-series inputs. When paired with phenotypic data, the functional connectivity calculated by FC-Net is used to categorise people as either neurotypicals or ADHD cases.
- **3D-CNN**: Convolutions are performed with the use of a three-dimensional filter by a 3D CNN to discover spatial patterns in the MRI's features. In a 3D CNN, the kernel can slide in all three dimensions, in contrast to a 2D CNN, where it can only slide in two dimensions. They demonstrated that the information obtained from fMRI data and that obtained from sMRI data are complementary to one another.

- **Deep-fMRI**: The time-series data are used as an input for the diagnosis by the deep-fMRI technique. In order to extract the features from the specific time series, a neural network is used as a feature extractor and applied to each region separately. The output of the feature extractor network is then input into a functional connectivity network, which is comprised of a few different networks that quantify similarity. Following this step, the output of the functional connectivity matrix is input into the classification network in order to complete the real diagnosis of ADHD.

5.5 Hyper-parameter Settings and the Proposed Model

Optuna [395] was utilised for the purpose of automatic hyperparameter tuning. The hyperparameter ranges utilised by Optuna for its tuning objective are presented in Table 5.2.

Name	**methods/values**
Optimizer	Adam, RMSprop, SGD
Learning rate	10^{-5} to 10^{-1}
weight Decay	5×10^{-5} to 5×10^{-1}
Loss Function	Negative Log-likelihood
Number of epochs	200

TABLE 5.2: The list of hyper-parameters used in Optuna. Optuna is a software framework that offers automatic hyperparameter optimization.

5.5.1 Structure of the Proposed model (ADHDNet)

The ADHDNet model we have proposed comprises five layers in total, encompassing both the input and output layers. The ADHDNet architecture comprises a neuron distribution of 9 for input, followed by 256, 1024, 512, 64, and 2 for output. The decision class is denoted by the labels 0 and 1. The numerical value of zero is used to denote a neurotypical individual, while the numerical value of one is used to denote an individual diagnosed with ADHD. The cross-entropy function has been selected as the loss function. The ADHDNet model exhibits a compact size of 3.15 MB and encompasses a total of 822,978 parameters that can be trained. The objective behind the development of ADHDNet was to design a concise model that possessed equitable non-linear mapping capabilities. The model's architecture is depicted in Fig. 5.2. The covariate shift

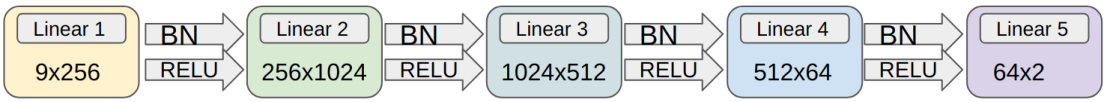


FIGURE 5.2: The architecture of the proposed model (ADHDNet). On each layer, we have used batch normalization with dropout = 0.20. We used ReLU as the activation function.

problem has been addressed by utilising ReLU and one-dimensional batch normalisation at each layer. Batch normalisation is used to transform intermediate inputs into a standard normal distribution.

5.6 Results

The accuracy of the graph-based approach is presented in Table 5.3. Bold typefaces highlight information with the maximum accuracy. The results indicate that the employment of 1D-CNN with an MSDL atlas with Pearson correlation yields the highest level of accuracy across all universities (KKI, NYU, NI, and Peking). The use of the Smith atlas on the data gathered from NI University resulted in LSTM exhibiting the least accuracy, amounting to 11.1%. Our custom made DNN gives comparable accuracy.

5.6.1 Comparisons with Baselines

The comparisons between our method and the baselines are presented in Table 5.4. As can be seen, our proposed method provides the highest level of accuracy for Peking and KKI.

5.6.2 Observations and Explanations

The precision of the categorization fluctuates and is contingent upon the selection of a specific atlas, functional connectivity matrix, and model.

One plausible explanation for the first observation could be attributed to the variations in the generated graph as we modify the atlas or connectivity measure. Alterations to the atlas lead to modifications in the quantity of regions, which subsequently

University	Atlas	Connectivity matrix		Models	
			1D-CNN	LSTM	ADHDNet
KKI	MSDL	CORR	**89.65**	88.29	89.63
		PAR-CORR	**89.65**	88.21	89.61
	Smith	CORR	61.00	61.50	61.00
		PAR-CORR	61.00	62.30	61.20
	Allen	CORR	64.20	64.30	64.22
		PAR-CORR	63.10	63.20	63.11
	Harvard-Oxford	CORR	61.00	62.10	62.21
		PAR-CORR	61.00	60.22	64.30
	BASC-64	CORR	72.40	73.33	72.22
		PAR-CORR	73.10	72.36	71.42
	BASC-444	CORR	71.54	72.42	72.52
		PAR-CORR	73.31	72.36	71.42
NYU	MSDL	CORR	**73.50**	62.50	73.43
		PAR-CORR	65.00	60.20	61.33
	Smith	CORR	61.20	61.25	49.22
		PAR-CORR	63.20	62.25	47.22
	Allen	CORR	57.24	57.23	47.20
		PAR-CORR	58.33	59.45	47.21
	Harvard-Oxford	CORR	59.60	57.30	28.20
		PAR-CORR	61.62	58.30	31.20
	BASC-64	CORR	61.11	52.27	54.90
		PAR-CORR	62.40	47.72	65.40
	BASC-444	CORR	57.14	52.30	45.80
		PAR-CORR	56.80	47.72	44.70
NI	MSDL	CORR	81.81	81.25	80.12
		PAR-CORR	**86.36**	81.81	86.26
	Smith	CORR	77.00	74.70	22.22
		PAR-CORR	44.00	11.10	25.90
	Allen	CORR	68.00	58.40	54.80
		PAR-CORR	61.50	62.30	50.14
	Harvard-Oxford	CORR	63.40	60.50	66.67
		PAR-CORR	60.20	58.90	52.70
	BASC-64	CORR	81.81	50.00	66.60
		PAR-CORR	76.40	48.90	50.00
	BASC-444	CORR	78.90	50.00	50.00
		PAR-CORR	74.20	50.00	48.20
Peking	MSDL	CORR	**87.50**	75.00	70.80
		PAR-CORR	83.33	72.00	70.80
	Smith	CORR	79.10	63.26	66.67
		PAR-CORR	84.56	61.54	65.34
	Allen	CORR	75.00	73.10	74.20
		PAR-CORR	71.00	72.60	72.21
	Harvard-Oxford	CORR	64.00	63.50	61.50
		PAR-CORR	62.00	62.22	63.20
	BASC-64	CORR	56.60	48.20	48.20
		PAR-CORR	52.30	48.20	51.76
	BASC-444	CORR	55.14	51.80	51.70
		PAR-CORR	55.14	50.20	48.23

TABLE 5.3: Accuracy in Graph-based approach. We have used four datasets (universities). For each dataset, six atlases are used and for each of the atlases, two type of connectivity matrices are used. We also compared three 1D models: 1D-CNN, LSTM and the proposed ADHDNet. CORR stands for Pearson-correlation and PAR-CORR stands for Partial-correlation

affect the number of nodes in a network. A modification in the measure of connectivity leads to alterations in the strength of associations among various regions. This subsequently results in changes in the number or value of edges in the network.

	NYU	Peking	KKI	NI
Deep Learning				
FCNet [355] (2017)	58.50	68.70	-	60.00
3D-CNN [357] (2017)	71.50	-	72.80	-
Deep fMRI [437] (2018)	73.10	62.70	-	67.90
Proposed Method (Best Accuracy)	73.50	87.50	89.65	86.36

TABLE 5.4: Accuracy comparison with baselines (%).

5.7 Discussion

In this chapter, we have used a complex network-based approaches—to classify the ADHD patients with controlled or cognitively normal (CN) people. We converted the connectivity matrix to a network and generated hand-crafted or automated features (a representation of the network). Then we applied several 1-D neural networks (1D-CNN, LSTM, and ADHDNet) for the classification task. In addition, we have used six different atlases and two different connectivity measures in this study.

5.8 Conclusion

The activity of each region of the brain is recorded using the functional magnetic resonance imaging (fMRI) data. A brain connectivity matrix, which can be modelled as a complex network, can depict the areas inside a human brain and the connections between them. This chapter contributes to the study of computational neurology by discussing how complex networks are used in the field of clinical diagnosis. Here, we propose a methodology showing how attention deficit hyperactivity disorder (ADHD) diagnosis can be successfully made using a complex network and its representations. In this investigation, we used a variety of one-dimensional neural network models. We have demonstrated how the classification outcomes and, consequently, the diagnosis can be significantly impacted by the selection of atlas and functional connectivity measures.

In the next chapter we shall discuss another important disease called Alzheimer's disease (AD). Unlike ADHD which is a neurodevelopmental disease, AD is a neurodegenerative disease. Thus, instead of AD classification we address another equally, if not more, challenging problem of progression of healthy brain towards AD by using complex network, their representation, and network manipulation.

Chapter 6

Investigating the Progression of the Brain Network: from a Healthy Brain to Alzheimer's Disease

Chapter Summary: In the previous chapter, we spoke about how to use a complex network and its representation to diagnose a neurodevelopmental brain disease called Attention Deficit Hyperactivity Disorder ($ADHD$). In this chapter we make another contribution to the field of neuroscience by studying another disease called Alzheimer's disease (AD). Since AD is a neurodegenerative disease, therefore instead of diagnosing, it is an equally challenging task to study the progression of healthy brain towards AD. Similar to the previous chapter a complex network is used to model the regions and their relationships in a brain. With the help of this complex network we build a mathematical model that combines both the anatomical and topological similarity between two brain regions to capture a hidden relationship among all pairs of brain regions to track how a real healthy control (HC) brain network progress towards the AD network. In other words the model generates a synthetic AD brain network from the real healthy control (HC) brain network by manipulating the HC network. To compute the anatomical similarity between two brain regions we make use of the euclidean distance between them and while calculating the topological similarity we make use of the network representation (network embeddings) of the nodes in the HC brain network.

Finally, we use several network properties to compare the synthetic brain network with the real brain network.

6.1 Introduction

Alzheimer's disease (AD) is the most common kind of neurodegenerative brain illness and is one of the more severe forms of dementia that affects elderly people [438]. Individuals with Alzheimer's disease exhibit signs of decreased memory, speech, analytical thinking, and other important cognitive skills that negatively impact a person's ability to carry out their day-to-day activities. This impairment develops as a result of the broken functional links that exist between the regions of the brain, which inhibit the typical information processing activities [439, 48]. It is hard to determine why the functional connections in the brain networks of people with Alzheimer's disease change compared to those of healthy people [440, 441]. Investigating the reason of this alternation of connection would be very significant and beneficial for gaining a better understanding of the many interactions that take place between different parts of a patient's brain and a healthy brain. By examining these changed interactions in the brain's networks would also help in the early detection and early therapy of Alzheimer's disease.

Many studies have also shown that changes in the structural and functional link pairs are widespread in Alzheimer's disease (AD) [442, 443]. In recent research, the technique of resting-state functional magnetic resonance imaging, also known as rs-fMRI, has been used to evaluate the progression of Alzheimer's Disease [444, 445]. It measures the temporal correlation of spontaneous blood oxygenation level-dependent (BOLD) signals in various parts of the brain. Studies [318] were done to find out how the functions of these different parts of the brain change as Alzheimer's disease (AD) gets worse. So, graph or network analysis [446, 46] is a better way to understand the structure of a network. It shows that AD patients' brain networks have strange patterns of functional connectivity organisation.

Network modelling is becoming more popular in the field of network neuroscience. It is used to model or simulate real brain networks [382, 383, 384]. One of the most important parts of network modelling is making a synthetic target brain network from

a real source brain network [1]. With the help of this generating process, one can simply determine the shifts that have taken place in the connection patterns between the nodes that make up the brain network. The real and synthetic target brain networks should have as many similarities as possible in order to achieve the highest level of accuracy in capturing the underlying connection patterns. To construct synthetic brain networks, the most recent methods [382, 383] of network modelling take into account both the anatomical distance (Eucledian distance) of the brain areas and the topological properties of the network nodes.

When topological features are computed based solely on local properties, there is a possibility that they will miss capturing some critical information contained within the entire network. This, in turn, will result in a model that is less accurate. In this thesis, we have used common node embedding techniques for computing topological characteristics in order to more accurately capture the global structural information of a network and get around this problem. As a result, for the purpose of computing the embedded topological properties of a network, we have utilised two distinct types of methodologies, namely (i) methods that are based on random walks and (ii) methods that are based on graph neural network (GNN) techniques. In this thesis, a new method for simulating and capturing the topological changes in a brain network as it goes from HC to AD[2] is proposed. In order to produce the connection probability for the simulation of real-brain networks, the model that has been proposed and referred to as NeuroANATOP combines the anatomical information with embedded topological information of the brain. In addition, the thesis discusses the pattern of change that occurs in a number of significant topological properties as they are applied to generated real-brain networks. These topological properties include the clustering coefficient, average path length, global efficiency, local efficiency, rich-club coefficient, and modularity. So, the primary goal of this research is to establish a method for the development of synthetic brain networks that have topological qualities that are comparable to those seen in brain networks that exist in the physical world.

Thus, the study in this chapter can be summarized as follows:

[1] With fMRI data, the real brain network can be generated. Inside the network, the "nodes" are the different regions of the brain, and the "edges" between the "nodes" are the connections between the various regions of the brain.

[2] In Appendix B, we showed the simulation of Mild Cognitive Impairment (MCI) brain network. MCI is a stage in between HC and AD.

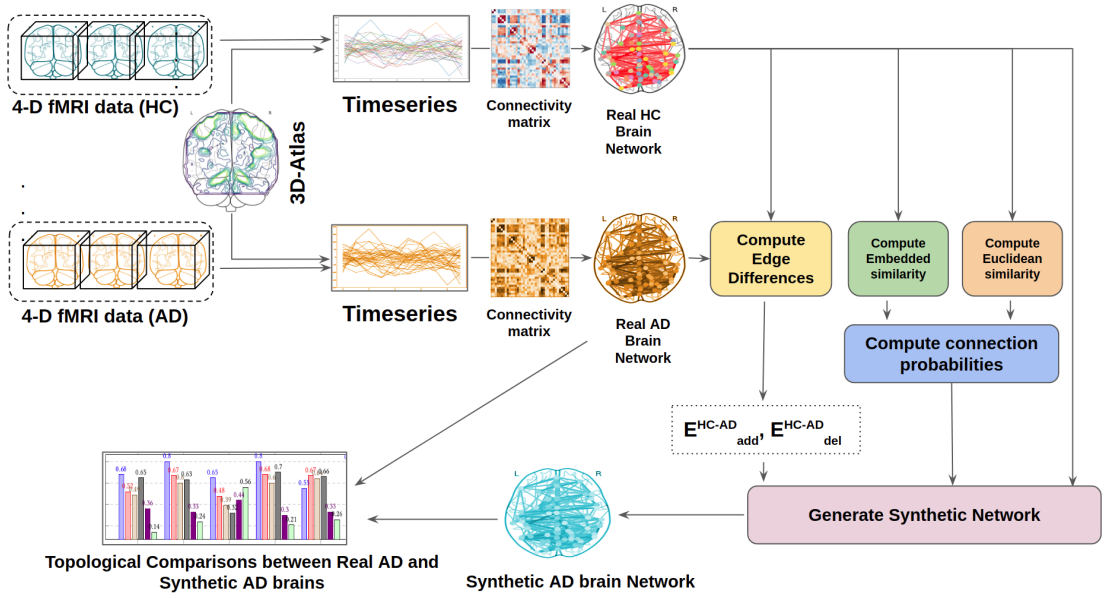


FIGURE 6.1: Generation of synthetic *AD* brain network from real *HC* brain network.

First, we Pre-processed all the raw fMRI data. Then, we masked the fMRI data with an atlas to identify the set of required regions of interest (ROIs) in the brain. Using the pre-processed real *HC* and *AD* fMRI data and the identified ROIs, we constructed actual *HC* and *AD* brain networks. After that, we used well-known node embedding methods to calculate the embedded topological similarity between nodes in the brain network. We then used the conventional Euclidean distance (ED) measure to determine how two nodes or ROIs are anatomically similar. Then the embedded topological and anatomical similarities are combined to generate the connection probability between nodes in a network. Using the resulting connection probabilities between nodes, we simulated the true *AD* brain network from *HC*. Finally, we analyzed the differences in topology and/or structure between real and simulated brain networks.

Fig. 6.1 shows the whole flowchart of the procedure for creating an *AD* brain network from a real *HC* brain network.

6.2 Proposed Strategies

In this section, we discuss the following things: (i) generation of a real brain network from the three-dimensional atlas and four-dimensional rs-fMRI data; and (ii) a framework for simulating the real *AD* brain network.

6.2.1 Generation of the real brain networks

An atlas that gives a unique brain area representing a node in the brain network is required in order to generate or construct a brain network based on the results of an rs-fMRI scan. In this study, we have used the Automated Anatomical Atlas (AAL) [447] to generate the brain network. The AAL atlas has 116 regions of interest (ROIs) representing the nodes in the network. Steps for generating the brain network include:

- First, using the AAL atlas, the 116 number of regions are masked, and for each region, average time series data is captured. This time series data is nothing but the BOLD (blood oxygen level-dependent) signal.
- Then, for each time-series generated, a pairwise pearson correlation is calculated, and a 116×116 matrix representing the pairwise correlation is generated.
- After that, we applied Fisher's r-to-z transformation [448] to this correlation matrix to convert the highly skewed distribution of the correlation coefficients into normal distributions with a stable variance.
- Then, the mean correlation matrix is created for each group HC and AD, by taking the average of the correlation matrices that are associated with each group.
- We then apply a threshold th to the mean correlation matrix of each group and generate a binary matrix. This binary matrix is a symmetric matrix. Thus, it can be thought of as an adjacency matrix. We take the value of the threshold $th = 0.5$. When the corresponding correlation coefficient in the average correlation matrix is higher than the threshold, the entries of the binary correlation matrix become 1, and when it is lower than the threshold, we replace it with 0.
- Finally, the adjacency matrix of each group's (HC and AD) represents a different network or graph G_{avg}^{HC} and G_{avg}^{AD} respectively.

6.2.2 Generation of the synthetic brain networks

After the generation of the real brain networks, the next task is to generate the synthetic brain networks for the AD (G_{syn}^{AD}) group by applying a generative model to the G_{avg}^{HC} - which is the real brain network of the HC group. Algorithm 8 provides a general

technique for creating synthetic brain networks from a genuine brain network. The comments' typefaces are highlighted in brown to help comprehend the algorithm.

Algorithm 8: Synthesizing AD networks from real HC network

Input : G^{HC}_{avg}: Real HC brain networks, M^{AD}_{add}, M^{AD}_{del}: number of edges to be added and deleted respectively to generate synthetic AD brain network from real HC brain network, a set of coordinates C representing the location of the brain regions, parameters $k1$ and $k2$.

Output: G^{AD}_{syn}: Synthetic AD brain network.

/* Generate all pair anatomical similarity matrix */

$EDS_{all} \leftarrow$ `getAllPairAS`(C); Step 1;

/* Generate all pair topological similarity matrix */

$TPS_{all} \leftarrow$ `getAllPairTS`(G^{HC}_{avg}); Step 2;

/* Generate all pair connection probability matrix */

$P_{con\_all} \leftarrow$ `getAllPairCP`$(TPS_{all}, EDS_{all}, k1, k2)$; Step 3;

/* Generate synthetic AD networks. Starting from G^{HC}_{avg}, this process iteratively add/delete edges to/from G^{HC}_{avg} using the connection probability P_{con_all} until M^{AD}_{add} number of edges added and M^{AD}_{del} number of edges deleted. */

$c_1 \leftarrow 0, c_2 \leftarrow 0$; Step 4;

while $NOT\ (c_1 \geq M^{AD}_{add}\ AND\ c_2 \geq M^{AD}_{del})$ **do**

  $r \leftarrow$ `genRandNum`$(0, 1)$;

  if $r \geq 0.5$ **then**

    /* Select a pair having no edge and maximum connection probability */

    $(u, v) \leftarrow$ `getNoEdgeMaxConnProb`$(G^{HC}_{avg}, P_{con\_all})$;

    /* add an edge between these nodes */

    $G^{HC}_{avg}(V^{HC}, E^{HC}) \leftarrow G^{HC}_{avg}(V^{HC}, E^{HC} \cup (u, v))$;

    $c_1 \leftarrow c_1 + 1$;

  else

    /* Select a pair having an edge and minimum connection probability */

    $(u, v) \leftarrow$ `getEdgeMinConnProb`$(G^{HC}_{avg}, P_{con\_all})$;

    /* Delete an edge between these nodes */

    $G^{HC}_{avg}(V^{HC}, E^{HC}) \leftarrow G^{HC}_{avg}(V^{HC}, E^{HC} - \{(u, v)\})$;

    $c_2 \leftarrow c_2 + 1$;

/* Return the synthetic brain network */

$G^{AD}_{syn} \leftarrow G^{HC}_{avg}$; Step 5;

return G^{AD}_{syn};

The explanation of the Algorithm 8 is provided in the following:

Inputs to the Algorithm.

- G^{HC}_{avg}: A real brain network of the HC group. V^{HC} is the set of nodes and E^{HC} is the set of edges.
- M^{AD}_{add}: The number of edges that need to be added in the real HC brain network G^{HC}_{avg} to generate a synthetic AD brain network G^{AD}_{syn}. It is the number of edges present in the real AD brain network G^{AD}_{avg} but absent in G^{HC}_{avg}.
- M^{AD}_{del}: The number of edges that need to be deleted in the real HC brain network G^{HC}_{avg} to generate a synthetic AD brain network G^{AD}_{syn}. It is the number of edges absent in G^{AD}_{avg} but present in G^{HC}_{avg}.
- C: A set of coordinates $\{c_1, c2, ..., c_n\}$ in the MNI space for each region inside a brain.
- $k1$ and $k2$: Preferential parameters corresponding to topological similarity and connection probabilities respectively.

The term G^{x}_{avg} is the real brain networks generated using the binary correlation matrices ADJ^{x}_{avg} where $x \in \{HC, AD\}$.

Step 1: Computation of anatomical similarity (ANS). In order to determine the degree of anatomical similarity (ANS) that exists between two regions of a brain, we compute the anatomical distance (AND). Formally,

$$ANS = \frac{1}{AND}$$

The anatomical similarity (ANS) for all the node pairs (ANS_{all}) is computed to generate the connection probabilities for simulation purposes. Over the entirety of the experiment, we computed the anatomical similarity (Euclidean similarity, EDS) between areas in a brain by using the conventional Euclidean distance, (ED) as the anatomical distance. The value that is returned by the EDS_{all} function is a representation of the Euclidean similarity between every pair of nodes in a network.

Step 2: Computation of topological similarity (TPS). To compare the structural similarity of two nodes in a network, the topological similarity between them is compared. Calculating the topological similarity between two nodes in a network often involves

computing their common neighbours. Nevertheless, this does not always succeed in capturing the global properties of a node in a network. To overcome the problem, node embedding into the vector space is necessary to retain the global characteristics. There are various node embedding techniques described in the literature. In this thesis, we used several random walk based (e.g., DeepWalk (DW) [41], Node2Vec (N2V) [105], etc.) and GNN based (Graph Convolutional Neural Network (GCN) [43], Graph SAmple and aggreGatE (SAGE) [113], etc.) node embedding techniques in our investigation. By computing the cosine distance similarity between these embedded feature vectors, the topological similarity for all pairs of nodes TPS_{all} are determined.

Step 3: Computation of Connection probability. Using TPS_{all}, EDS_{all}, $k1$ and $k2$, the connection probability $P^{CN}_{con\_all}$ for all pairs of nodes is generated. In Section 6.3, we discuss this computation in more detail.

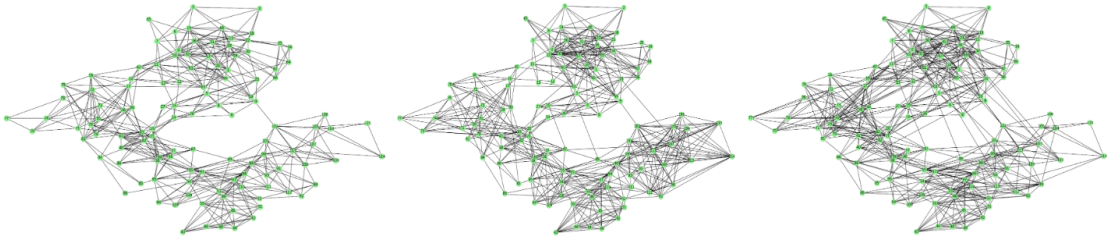
Step 4: Generation of synthetic network. Utilising the generated connection probabilities and the aforementioned edge differences, the synthetic graph G^{AD}_{syn} from G^{HC}_{avg} are generated. In the $genRandNum$ method (in Algorithm 8), a random real number r is generated between 0 and 1. Then,

- If r is greater than or equal to 0.5 then a pair of nodes is chosen such that there is no edge exist between them and the connection probability is maximum. Then an edge is added between them.
- Else, an edge with the minimum connection probability in G^{HC}_{avg} is chosen and deleted.

The above steps will continue until the number of deleted edges becomes M^{AD}_{del} and the number of added edges becomes M^{AD}_{add}.

Step 5: Returning the synthetic network. Finally, the modified HC network is the synthetic AD network and this synthetic network is returned.

Fig. 6.2(b) shows the generated synthetic Alzheimer's disease (AD) brain network (G^{AD}_{syn}) from real or true healthy control (HC) brain network (G^{HC}_{avg}).



(a) Real HC Brain network (b) Synthetic AD brain network (c) Real AD brain network

FIGURE 6.2: Three networks: (a) represents the real HC network (G^{HC}_{avg}); (b) represents the synthetic AD network (G^{AD}_{syn}) generated from HC network using the N2V embedding technique; (c) represents the real or true AD brain network (G^{AD}_{avg}). From these figures, it is evident that the proposed constructed G^{AD}_{syn} network closely resembles the G^{AD}_{avg} network.

6.3 Calculation of connection probabilities

In the earlier part of this chapter, we came to the conclusion that connection probabilities are required in order to derive the synthetic brain network G^{AD}_{syn} from the actual brain network G^{HC}_{avg}. It is necessary to make a prediction regarding whether or not there ought to be a link between a couple of nodes. To put it another way, by utilising these probabilities of connection, we are able to capture the formation pattern of these synthetic networks. In the context of a brain network, the computation of connection probability was initially proposed in [382]. They integrate the common neighbour similarity (CNS) (by taking the number of common neighbours) as the topological similarity and the Euclidean similarity (EDS) (by taking the inverse Euclidean distance) as the anatomical similarity in their model, which we will refer to as CN. This allows them to calculate the connection probability for a pair of nodes (u, v) in a network using the formula below:

$$P^{CN}_{con}(u, v) = CNS(u, v)^{k1}.EDS(u, v)^{k2} \tag{6.3.1}$$

where $k1$ and $k2$ are the two preferential parameters of the aforementioned equation, which are determined from the data.

Problem of CN**.** Although CN is a straightforward model, it only accounts for the node's local characteristics, which might not be universal across the network. As a result, we need to come up with a different method for identifying a node's distinctive

characteristics in a network. When creating synthetic AD brain network, a better connection probability can be calculated using this special property of nodes.

Proposed solution to the problem of CN. One method for displaying the distinctive qualities of individual network nodes is to vectorize each node individually. Calculating the cosine distance similarity between the node pair (u, v) allows one to gauge how similar they are topologically. In this approach, the unique qualities of each node are properly preserved, and topological similarity is increased more than when the quantity of shared neighbours is used. Thus, our goal is to present a novel version of the conventional CN model that takes into account the embedded similarity ($EBS(u, v)$) as a topological similarity (TPS) and the Euclidean similarity (EDS) as the anatomical similarity (ANS) between two nodes u and v.

In this manner, the updated version of the CN model may offer a better connection probability between nodes u and v, which will be helpful in modelling the impairments that occur when HC progresses to AD.

Formally, the connection probability between nodes u and v can be determined by:

$$P_{con}^{emb}(u, v) = TPS(u, v)^{k1}.ANS(u, v)^{k2}$$

or, equivalently,

$$P_{con}^{emb}(u, v) = EBS(u, v)^{k1}.EDS(u, v)^{k2} \tag{6.3.2}$$

$k1$ and $k2$ are two preferential parameters. By employing simulated annealing [449] to optimise the performance measure, the ideal values of $k1$ and $k2$ are discovered.

Motivation behind the proposed solution. Anatomical distance between any pair of regions was taken into consideration in earlier research by Kaiser *et al.* [450, 451] to determine connection probabilities between them. Kaiser *et al.* used an exponential decay model with a penalization parameter to simulate real-brain networks. They discovered that these simulation models, in terms of global efficiency, local efficiency, clustering coefficient, etc., did not correctly match the real-brain networks. As a result, it seems that just penalising connection probability based on anatomical distance won't be enough

to accurately mimic the topological characteristics of functional networks in human brains. In order to better synthesize the real-brain network structure, there must be some sort of link, or trade-off, between distance penalization and one or more other factors.

We examined embedded topological similarity and combined it with anatomical similarity in order to maintain the balance or trade-off. According to Eqn.6.3.2, the preferential parameters ($k1$ and $k2$) help to optimise the connection probability. The best-fitting of these connection probability models included a power-law based anatomical similarity preference as well as a power-law function of a topological similarity term. According to the empirical results, real-brain network modelling is more accurate when this two-parameter connection probability model with trade-off is used.

Construction of the proposed variants. To compute the various embedded similarities, we make use of a variety of node embedding techniques, including three distinct methods that are based on random walks and four distinct methods that are based on well-known neural network algorithms. The various embedded similarities each provide their own set of connection probabilities. The descriptions of these models are provided in table 6.1.

The feature vector that corresponds to a node v is constructed for each of these models by first taking a vector F with the same length as the number of nodes in the networkand setting all of its entries to zero. Next, one adds a 1 to the index u of F if the node u is adjacent to v. Using the well-known Infomap [30] community detection technique leads to the determination of the number of communities, which in turn leads to the assignment of the class number for the network. Each community is given a class label, with the first one starting at zero. In accordance with this, the GNNs are trained by selecting 10% of the nodes from each class to serve as the training nodes, and ultimately, the embedding vectors are obtained from the hidden layer that is next to the last one.

As a result, each node u of the real brain networks G_{avg}^{HC} for HC has a vector representation $\vec{u}_{emb}$. After that, the topological similarity between two nodes, u and v, in a real brain network is determined by calculating their cosine distance in the embedded space.

Embedding methods	Type	Descriptions
DeepWalk (DW) [41]	Random walk	In order to provide insights into the localised structures that are present inside networks, it employs a technique called randomised path traversal. This is accomplished by the utilisation of these random pathways as sequences, which are subsequently put to use in the process of training a Skip-Gram language model [452].
Node2vec (N2V) [105]	Random walk	The learning of the vector representation of a network's nodes is the goal of the node2vec algorithm, which does this by maximising a neighborhood-preserving objective function. It is an expansion of the popular DeepWalk node embedding technique, and it is composed of the well-known state-of-the-art word embedding algorithm word2vec [336].
LINE [106]	Random walk	LINE achieves its results by optimising an objective function in such a way that it maintains both the global and local network structures.
ChebyNet [126]	Deep Learning	ChebyNet is a spectral-based Graph Convolutional Network that utilises a fast localised convolutional filter on networks as its foundation.
GCN [43]	Deep Learning	GCN is a linear approximation of Spectral-based Graph Convolutional Networks.
Graph SAGE [113]	Deep Learning	Graph SAGE relies mostly on inductive learning for its classification of nodes; however, it also functions in a transductive environment.
GAT [44]	Deep Learning	GAT is put into action by applying varying weights, or values, to the individual nodes that make up a neighbourhood.

TABLE 6.1: List of embedding methods.

$$EBS(u,v) = \frac{\vec{u}_{emb}.\vec{v}_{emb}}{||\vec{u}_{emb}||.||\vec{v}_{emb}||}$$

In conclusion, the following equation can be used to define the proposed connection probability between two nodes u and v in a real brain network.

$$P_{con}^{emb}(u,v) = \left(\frac{\vec{u}_{emb}.\vec{v}_{emb}}{||\vec{u}_{emb}||.||\vec{v}_{emb}||}\right)^{k1}.EDS(u,v)^{k2} \tag{6.3.3}$$

The proposed approach, as well as the recommended connection probability that was defined earlier, can be altered by incorporating a variety of different embedding

stat	*HC*	*AD*
Number	62	63
Male/Female	27/35	31/32
Average age	73.78	75.11
Average weight	72.45	74.57

TABLE 6.2: Structural statistics of participants belonging to healthy controls (*HC*) group and Alzheimer's disease (*AD*) group.

techniques.

We have termed the respective proposed variants as PV_i, $i \in \{1, 2, ..., 7\}$ which corresponds to the various embedding methods, namely DW, N2V, LINE, GCN, SAGE, ChebyNet, and GAT, respectively.

6.4 Data Collection and Prepossessing

6.4.1 Data collection

We have compiled the fMRI datasets, which include 125 participants. The metadata for these participants is accessible to the public and can be downloaded from the Alzheimer's Disease Neuroimaging Initiative (ADNI) repository. The ADNI repository can be accessed at the following URL: `http://adni.loni.ucla.edu`. Following the downloading of the data, we separated all of the participants into two groups: those with healthy controls (*HC*) and those with Alzheimer's disease (*AD*). Table 6.2 provides an illustration of the specifics of the clinical characteristics that differ between each of these groups.

6.4.2 Data preprocessing

As already discussed in section 2.1.13, the raw fMRI data needs to be processed to remove several artifacts. These preprocessing includes: distortion correction [167], head motion correction [163], slice-timing correction [165] and spatial smoothing [176]. Details of these steps can be found in section 2.1.13. Here, we have used Data Processing Assistant for Resting-State fMRI (DPARSF) software for the preprocessing the fMRI data.

6.4.3 Time series generation

We extracted the BOLD signals (time series data) from numerous distinct brain regions using the preprocessed fMRI data. There is no information about the various brain areas in the fMRI scan data that was collected. Therefore, the fMRI data must be masked using a standard atlas in order to extract the time series from a variety of distinct brain areas. In this study, we built the brain networks using the automated anatomical atlas (AAL). This atlas has 116 regions of interest (ROIs).

6.5 Results

6.5.1 Comparing real brain networks: G_{avg}^{HC} and G_{avg}^{AD}

The present study conducts a comparative analysis of the topological properties of the actual brain networks G_{avg}^{HC} and G_{avg}^{AD}, as illustrated in Fig. 6.3. Six topological properties, namely clustering coefficient, modularity, transitivity, local efficiency, global efficiency, and rich club index, are utilised. The figure displays three distinct groups. As depicted on the left of FIGURE 6.3 displays a set of bars that have been plotted to represent G_{avg}^{HC}, while the right figure corresponds to G_{avg}^{AD}. The results indicate that there is an increase in local efficiency, modularity, and global efficiency as the group transitions from HC to AD. Conversely, the three remaining topological properties, namely average clustering coefficient, transitivity, and rich club coefficient, exhibit a decrease in the same direction. The aforementioned results indicate that HC and AD exhibit distinct topological characteristics.

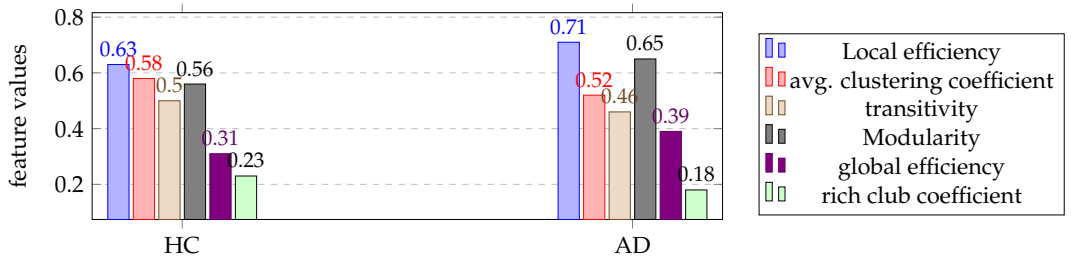


FIGURE 6.3: Comparison of various topological properties among the real brain networks of healthy control (HC) and Alzheimer's disease (AD) group.

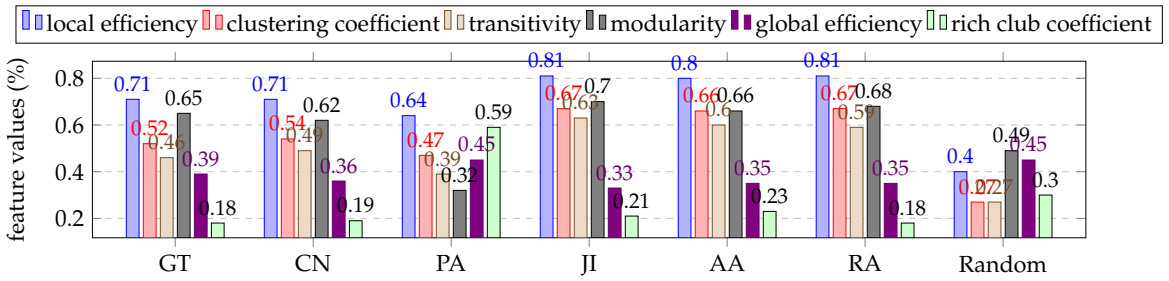
6.5.2 Simulation performance of the proposed variants

Within this section, an assessment is conducted on the efficacy of the proposed network generative model, referred to as **NeuroANATOP**, as well as other competitive models. This is achieved through a comparison of the synthetic networks generated by these models with the actual brain network. As previously stated, the six crucial topological properties were utilised in this assessment. The baselines employ comparable methodology, with the exception of the aspect of topological similarity, as indicated in Table 6.3. Furthermore, apart from the aforementioned baselines, we have incorporated a random model in which edges are added or removed in a *random* manner, in contrast to the other models. The values of $k1$ and $k2$ were assigned as 20 and 8, respectively, to produce the empirical outcomes of all the competitive models. The aforementioned values were acquired through the implementation of simulated annealing on MSI (see section 6.5.3). The parameters were held constant across all networks: The GNN employs a total of two layers. The hyperparameters for the training process include a learning rate of 0.001 and a total of 200 epochs.

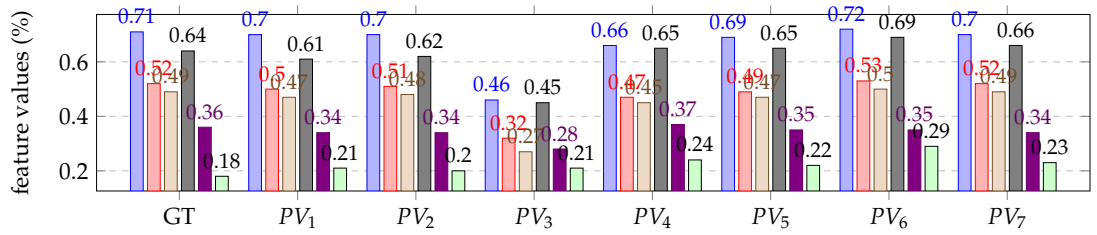
Models	Mathematical Definitions
Common Neighbour (CN) [382]	$TS(u,v) = \lvert N(u) \cap N(v) \rvert$
Jaccard Index (JI) [313]	$TS(u,v) = \frac{\lvert N(u) \cap N(v) \rvert}{\lvert N(u) \cup N(v) \rvert}$
Preferential Attachment (PA) [453]	$TS(u,v) = \lvert N(u) \rvert . \lvert N(v) \rvert$
Resource Allocation (RA) [454]	$TS(u,v) = \sum_{y \in N(u) \cap N(v)} \frac{1}{\lvert N(y) \rvert}$
Adamic–Adar (AA) [455]	$TS(u,v) = \sum_{y \in N(u) \cap N(v)} \frac{1}{log \lvert N(y) \rvert}$

TABLE 6.3: Topological similarities in the baselines.

Comparing G^{AD}_{syn} with G^{AD}_{avg}. The bar plot depicted in Fig. 6.4a illustrates six significant topological characteristics for various competitive models, including the random model. The bar plot illustrating the topological features generated by the proposed variants in comparison to the ground truth is presented in Fig. 6.4b. The leftmost group of bars in Fig. 6.4a and 6.4b represent the ground truth. The results depicted in Fig. 6.4b demonstrate that, for the majority of cases, the topological characteristics of the proposed variants (PV_2 (N2V) and PV_7 (GAT)) closely resemble those observed in the ground truth. Moreover, it can be observed from Fig. 6.4a that the proposed model outperforms the other models, with RA being the second best performer and *Random* exhibiting the poorest performance. Similarly, it can be observed that CN, JI, AA, and



(a) real *AD* brain network (leftmost) vs synthetic *AD* brain networks produced by the baselines.



(b) real *AD* brain network (leftmost) vs synthetic *AD* brain networks produced by the proposed variants.

FIGURE 6.4: Comparison of topological properties: real *AD* brain network (leftmost) vs synthetic *AD* brain networks produced by the various models. GT: Ground Truth, PV_1: Deep Walk (DW), PV_2: Node to Vec (N2V), PV_3: Line Embedding (LINE), PV_4: Graph Convolutional Neural Network (GCN), PV_5: GraphSAGE (SAGE), PV_6: ChebyNet (Cheb), PV_7: Graph Attention Network (GAT).

RA exhibit comparable performance, albeit less efficacious in comparison to the proposed approach. The performance of *PA* surpasses that of *Random*, however, it falls short in certain cases when compared to other models that are considered competitive.

6.5.3 Model performance evaluation: Modified Similarity Index (MSI)

The Similarity Index (*SI*) [382, 383] was employed to assess the performance of our study. The similarity index was adjusted by incorporating an additional value of 1 in the denominator to prevent the occurrence of a zero value in the denominator. This revised metric is denoted as *MSI*.

Formally it can be defined as:

$$MSI = 1/(1 + (E_{LE} + E_{AC} + E_T + E_M + E_{GE} + E_R)).$$

Six significant topological properties are incorporated to facilitate a rational and persuasive assessment. E_x denotes the relative error between real and synthetic brain networks corresponding to property x. The descriptions of the relative errors are provided

in Table 6.4.

Notation	Descriptions
E_{LE}	Relative error between real and synthetic brain networks corresponding to the local efficiency.
E_{AC}	Relative error between real and synthetic brain networks corresponding to the avg. clustering coefficient.
E_T	Relative error between real and synthetic brain networks corresponding to the transitivity.
E_M	Relative error between real and synthetic brain networks corresponding to the modularity.
E_{GE}	Relative error between real and synthetic brain networks corresponding to the global efficiency.
E_R	Relative error between real and synthetic brain networks corresponding to the rich club coefficient.

TABLE 6.4: Descriptions of the Errors.

Evaluating competitive models using MSI The relative errors of the topological properties of the synthetic networks generated and the ground truth network are presented in Table 6.5. Furthermore, the MSI metric is calculated to distinctly distinguish the efficacy of the suggested variations in contrast to alternative approaches. The MSI values are presented in the final column of Table 6.5.

6.5.4 Degree Distribution

The distribution of degrees is a significant structural attribute that can be utilised to capture changes in network structures during the progression of Alzheimer's disease. Comparing the degree distribution of the real brain networks and the synthetic brain networks is essential to justifying the effectiveness of the proposed generative model. Here, we generated the degree distributions of networks using a variety of competing techniques, including the suggested variations, and we compared them to the distributions of the actual target networks. The experimental result shown in Fig. 6.5 demonstrates that, when compared to the other baseline models, the degree distribution of the synthetic networks produced by the suggested variants (Fig. 6.5b) closely resembles the degree distribution of the target networks. The JI, AA, and RA methods all perform

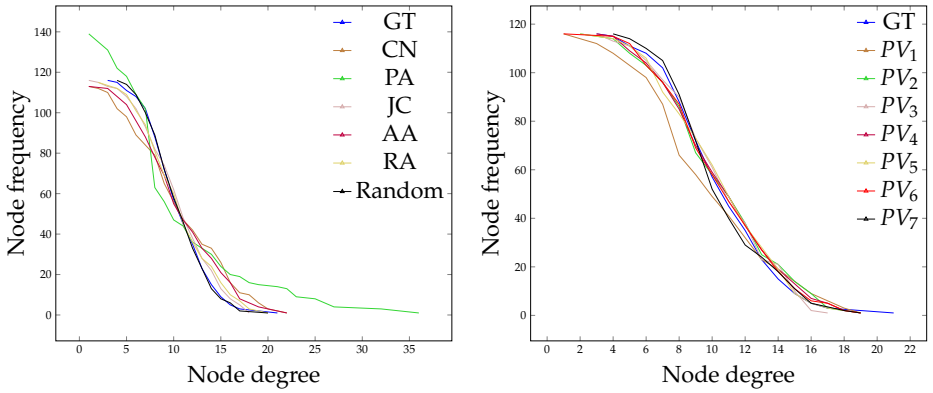
Models	E_{LE}	E_{AC}	E_{GE}	E_M	E_T	E_R	MSI
CN	0.09	0.15	0.01	0.15	0.16	0.04	0.63
JI	0.10	0.15	0.05	0.17	0.0	0.06	0.65
PA	0.07	0.05	0.33	0.07	0.41	0.06	0.50
RA	0.01	0.15	0.03	0.13	0.06	0.04	0.70
AA	0.09	0.15	0.01	0.14	0.03	0.04	0.68
PV_1	0.01	0.02	0.02	0.03	0.02	0.03	0.88
PV_2	0.01	0.01	0.01	0.02	0.02	0.02	0.92
PV_3	0.25	0.20	0.22	0.19	0.08	0.03	0.51
PV_4	0.05	0.05	0.04	0.01	0.01	0.06	0.82
PV_5	0.02	0.03	0.02	0.01	0.01	0.04	0.88
PV_6	0.01	0.01	0.01	0.05	0.01	0.11	0.83
PV_7	0.01	0.0	0.0	0.02	0.02	0.05	0.90
Random	0.31	0.25	0.16	0.19	0.12	0.06	0.47

TABLE 6.5: Comparison of MSI-values for all the models (baselines and our proposed model). Larger MSI value indicate more similarity between the synthetic (AD) and real (AD) brain networks. red cell indicates the worst performer, green cell indicates the best performer and blue cell indicates the second best performer. CN: Common Neighbor, PA: Preferential Attachment, JI: Jaccard Index, AA: Adamic–Adar, RA: Resource Allocation, PV_1: Deep Walk (DW), PV_2: Node to Vec (N2V), PV_3: Line Embedding (LINE), PV_4: Graph Convolutional Neural Network (GCN), PV_5: GraphSAGE (SAGE), PV_6: ChebyNet (Cheb), PV_7: Graph Attention Network (GAT).

well and are close to the targeted network. PA and CN methods outperform the proposed method and fail to reach the target networks. Thus, the empirical results support the proposed generative model's effectiveness in the progression of HC to AD.

6.5.5 Sensitivity of parameters $k1$ and $k2$ of the proposed ensemble model

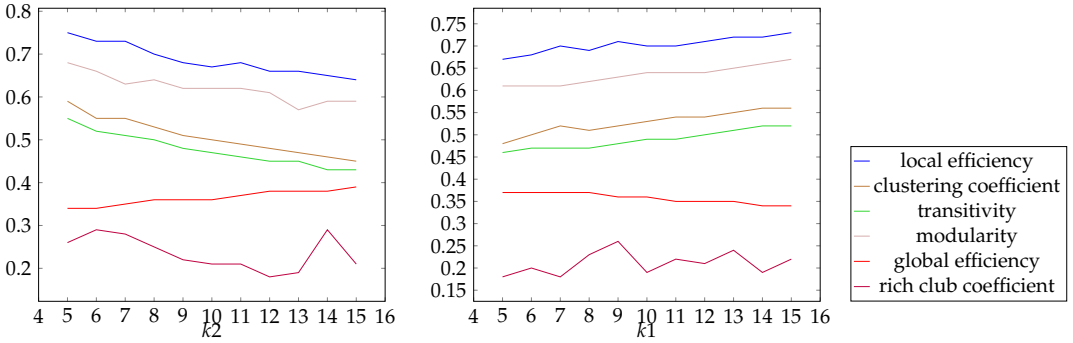
While simulating the brain network, we examine how the preferential parameters $k1$ and $k2$ affect various topological features in this section. While simulating the AD network, we have shown the effects of $k1$ and $k2$ on six topological features of real-world complex networks that correspond to the proposed variant PV_2 in Fig. 6.6. Since the considered variant (PV_2) offers greater performance compared to the competing variants, we have provided the visual findings corresponding to the proposed variant in Fig. 6.6. In Fig. 6.6a, we showed the changes in network properties while producing G_{syn}^{AD} from G_{avg}^{HC} by raising $k2$ from 5 to 15 while maintaining the value of $k1$ at 20. Fig. 6.6b shows the results of generating G_{syn}^{AD} from G_{avg}^{HC}, while altering $k1$ from 15 to 25 and maintaining $k2$ to 8.



(a) Degree distribution of real and synthetic brain networks for AD group: GT vs baselines

(b) Degree distribution of real and synthetic brain networks for AD group: GT vs proposed variants

FIGURE 6.5: Comparing the complementary cumulative degree distribution of networks generated through various generative models. GT: Ground truth, CN: Common Neighbor, PA: Preferential Attachment, JI: Jaccard Index, AA: Adamic–Adar, RA: Resource Allocation, PV_1: Deep Walk (DW), PV_2: Node to Vec (N2V), PV_3: Line Embedding (LINE), PV_4: Graph Convolutional Neural Network (GCN), PV_5: GraphSAGE (SAGE), PV_6: ChebyNet (Cheb), PV_7: Graph Attention Network (GAT).



(a) G_{avg}^{HC} to $G_{syn}^{AD}, k1 = 20, k2 \in [5, 15]$

(b) G_{avg}^{HC} to $G_{syn}^{AD}, k1 \in [15, 25], k2 = 8$

FIGURE 6.6: Changes in various topological properties in the generated synthetic networks from the real brain network upon tuning the parameters $k1$ and $k2$. G_{avg}^{HC} to G_{syn}^{AD} represents generation of synthetic network G_{syn}^{AD} from G_{avg}^{HC}

Observations: (i) local efficiency, average clustering coefficient, transitivity, modularity, and global efficiency increase when we tune $k2$ while keeping $k1$ fixed; and (ii) local efficiency, average clustering coefficient, transitivity, modularity, and global efficiency decrease when we tune $k1$ while keeping $k2$ fixed. The empirical findings support the sensitivity of the parameters $k1$ and $k2$ in replicating real brain networks for the AD. We used optimized values of $k1$ and $k2$ obtained from simulated annealing throughout the experiment.

6.5.6 A case study of brain sub-regional simulation performance

Multiple brain areas or portions may be impacted over the course of Alzheimer's disease, which may cause a reduction in daily activities. The deep temporal-cerebellar area and the frontal brain are two of the most affected brain regions in people with AD, according to authors [456, 457]. In this regard, we are investigating the network structural alterations that take place in these two significant brain sub-regions as they advance from HC to AD. Our goal is to determine whether the connection pattern seen in the frontal and cerebellar subregions of the real AD network can be accurately simulated by our suggested simulation model. Results from the experiment show that the proposed model successfully created connections within each of the sub-regions, or to put it another way, the suggested model successfully simulates the connectivity patterns in each of the sub-regions observed in the actual AD networks. Figs 6.7(a) and 6.7(b) show the axial views of the real and hypothetical simulated networks corresponding to the frontal sub-regions of a brain. Coronal views of the real frontal sub-regional networks and the simulated frontal sub-regional networks are depicted in Figs 6.7(c) and 6.7(d).

There are 18 nodes and 52 edges that make up the frontal network in the real AD brain. In comparison, the frontal sub-region of the suggested simulated network includes 60 edges, eight more than the actual network, even though it has the same number of nodes (18 nodes).

As a result, it is clear from those figures that the suggested simulated frontal subregional connection patterns closely correspond to the actual connectivity pattern of the network seen in the frontal area of the real AD brain. Similar to this, the proposed simulated model accurately reproduces the structural connectivity pattern seen in the cerebellum sub-region of a real AD brain, as shown in sub-figures 6.8(a) and 6.8(b) that show sagittal views and 6.8(c) and 6.8(d) that show coronal views, respectively.

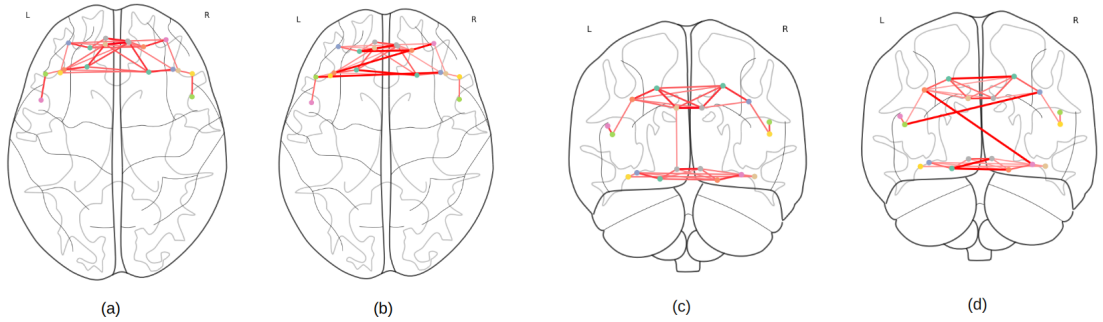


FIGURE 6.7: The axial views of the real and proposed simulated networks, corresponding to the frontal sub-region of an AD brain network, are depicted in Figures (a) and (b), respectively. The coronal views of the actual and proposed simulated frontal sub-regional networks are shown in Figures (c) and (d).

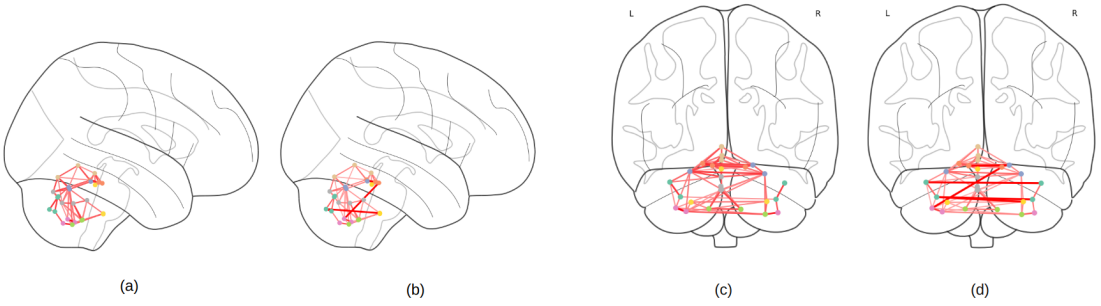


FIGURE 6.8: The sagittal views of the real and proposed simulated networks, corresponding to the cerebellum sub-region of an AD brain network, are depicted in Figures (a) and (b), respectively. The coronal views of the actual and proposed simulated cerebellum sub-regional networks are shown in Figures (c) and (d).

6.6 Discussions

We proposed a variant of generative models that combines structural and functional similarities to analyze the progression from Healthy Control (HC) to Alzheimer's Disease (AD). Given a fMRI image, the structural or anatomical similarity has been measured using Euclidean distance, and the topological similarity is measured using the distance in the embedded space by capturing the network representation. Additionally, we concentrate on adding or removing links or connections between several significant brain regions or places. Our goal is to identify a crucial connection mechanism that helps brain networks transition from HC to AD. We have demonstrated that the embedding structure and Euclidean distance can be combined to enhance the simulation performance of the suggested variant model. The simulation outcomes show that node embedding techniques have their own benefits in determining topological similarities and may thus be helpful in simulating actual AD brain network. The experimental

findings also point to the usefulness of the GNN models under consideration for computing connection probability for simulating brain networks. Finally, we can draw the conclusion that the suggested generative models may be helpful in accurately simulating the topological variations of brain networks when HC progresses to AD.

6.7 Conclusion

Alzheimer's disease is a neurodegenerative disease that gradually impairs memory, reasoning abilities, and, eventually, the capacity to do the most basic duties. Thus, it is extremely important to study the progression of a healthy brain towards Alzheimer's disease to help neurologists understand the present condition of the brain. In this chapter, we construct a mathematical model that explains how a healthy brain network can be manipulated to generate a synthetic AD brain network and thus capture the progression of a healthy brain towards AD. Experimental evidences indicate that the proposed NeuroAnatop model not only captures the progression of the Alzheimer's brain network, but also outperforms a number of competitive models. Therefore this research represents a significant contribution towards the field of computational neurology with the help of complex networks and their representation.

Chapter 7

Conclusions and Future Directions

Chapter Summary: In this final chapter, we sum up our conclusions by evaluating the thesis's contributions. Here, we talk about how these contributions could be able to help with several open issues in three areas: communities in complex networks, graph-based deep learning, and neuroscience. The thesis contains three parts, and each part contributes to a particular area. Part - I contains only one chapter that contributes to the communities in complex networks; Part - II also contains a single chapter that contributes to graph-based deep learning; and the last part, i.e., Part - III, contains two chapters where both chapters contribute to the neuroscience area in different ways. We then go on to highlight some potential future study areas based on the approaches given in this thesis. The highlighted future study options enhance the thesis's contributions in those aforementioned areas.

7.1 Conclusions

7.1.1 Communities in complex network

From social networks to biological networks, a wide range of real-world systems can be represented as complex networks. Within a complex network, finding communities has many uses, such as identifying a group of individuals participating in a specific activity in a social network or identifying a collection of proteins that collectively contribute to a particular bodily dysfunction, etc. How efficiently a certain parameter,

such modularity or entropy, is optimised determines the characteristics of community discovery methods. Since all of these optimisation issues are NP-Hard, the process becomes stochastic when the methods are approximated. Due to this stochastic nature, community structures altered in each execution.

In Part - I, we discussed how we created algorithms that made use of both manual and automated feature engineering (graph representations) on the network to tackle this challenge and this contributed to this area. We came up with two novel techniques as a result: unsupervised and semi-supervised. Inspired by the traditional histogram-based image thresholding methodologies, where the foreground image can be demarcated from the background image, we employ the same histogram-based thresholding methods to distinguish the constant community from the non-constant community in our unsupervised approach. For the semi-supervised approach, the original network was first converted to its corresponding line graph, and then, with the application of a Graph Convolutional Neural Network (GCN) the nodes of the line graph were classified, which resulted in identifying the constant communities in the original network.

It is important to note that the constant community problem does not effectively address the fundamental task of community detection, as it excludes numerous nodes that tend to move between groups. These nodes are shared by communities and play a crucial role in the system.

7.1.2 Graph based deep learning

Graph neural network (GNN) is a semi-supervised learning tool that requires only a small amount of data that has been labelled as its training data. Experimental evidence reveal that if the number of training data (nodes) is reduced then its representational capacity goes down, which affects the downstream tasks (in our case, the node classification accuracy drastically dropped). The issue with training nodes is that it can be very challenging to acquire them at times. If we were to directly increase the amount of training nodes, it would be counterproductive to use a semi-supervised approach because it would negate the point of using it.

In Part II, we address this challenge by increasing the training nodes indirectly. We proposed a novel input-level intervention technique that uses a random walk-based

method and some classic machine learning tools (clustering and classification) to agnostically increase the number of training nodes by capturing similar nodes from various non-contiguous sub-graphs of a graph. With this agnostically extended training set, the GNN learned better representation, and as a result, the node classification accuracy increased. In addition to the input-level intervention, which is a pre-processing method, we proposed a post-processing method - an output-level intervention. In this novel method, we used the output of the GNN, i.e., the nodes whose labels are predicted, and with the help of a random walk-based method, we relabeled a particular set of nodes that were possibly mispredicted by the GNN which in turn results in more accurate classification. Apparently, our input- and output-level intervention methods make a promising contribution to graph-based deep learning areas.

7.1.3 Neuroscience

The data obtained from functional magnetic resonance imaging (fMRI) are used to record the activity level of each individual region of the brain. A brain connectivity matrix constructed from the fMRI data is able to illustrate the regions that are contained within a human brain as well as the relationships (functional connections) that exist between those regions. Since this matrix is symmetric in nature, it can be considered as an adjacency matrix of a weighted complex network. In other words, a complex network represents the regions and the inter-regional relationships of a human brain. Since the human brain is an extremely complex system, it is a very challenging task to deal with certain brain diseases. Sometimes it is very important but difficult to identify certain neurodevelopmental diseases, viz., Attention Deficit Hyperactivity Disorder (ADHD) by merely observing the patient's behaviours. In a similar way, there are some neurodegenerative diseases, such as Alzheimer's disease (AD) for which studying the progression of this disease is a challenging task.

In Part - III, we contributed to the area of neuroscience by addressing these challenges. Firstly, we address the diagnosis of the ADHD disease with the help of a complex network and its representation. In addition, we have illustrated how the choice of an atlas and the functional connectivity measures can have an effect on the classification outcomes and, as a consequence, the diagnosis. Secondly, We create a mathematical model that describes how a healthy brain network may be manipulated to generate

a synthetic AD brain network, and in doing so, we are able to capture the progression of a healthy brain towards AD.

7.2 Future directions

In this chapter, we make an effort to foresee some of the ways in which the research given here could be developed further. The suggested approaches to each of the distinct problems have some limits, therefore future extension is conceivable by resolving the restrictions. We will briefly go over the proposed approaches' limits and potential future extensions in relation to each particular issue separately for each chapter.

7.2.1 Future Directions for Chapter 3

The identification of overlapping continuous communities is one of the crucial tasks we hope to do in the future. In order to detect overlapping communities, it is necessary to cluster the nodes into subgroups, with the result that some nodes will be members of more than one community (i.e., they will have more than one community ID). There are two characteristics that must be true for nodes in overlapping constant communities: (i) they must always be part of the same overlapping community (s), and (ii) nodes may have more than one label. We need to investigate which attributes are most informative for categorization in order to pinpoint the boundaries between the overlapping groups. With nodes being able to belong to many communities, overlapping communities face the additional difficulty of having a larger memory footprint. Our program is required to be executed on distributed clusters in order to grow and manage large-scale graphs for discovering overlapping communities. To implement the proposed technique and its overlapping variant, we want to tweak our data structure and take advantage of high-performance computing frameworks (HPC) like MPI/Open ACC.

7.2.2 Future Directions for Chapter 4

Apply the interventions on Inductive settings: In this chapter, we only considered the transductive settings where we were aware of the training nodes during the training. In future we shall extend our work in the inductive setting [458]. In an inductive setting,

we cannot see the test nodes during the training phase. To classify a pattern according to its characteristics, inductive learning first assumes that there is a set of rules that are to be applied by the model. In the inductive scenario, the node embedding problem is more difficult than in the transductive setting because the algorithm needs to "align" newly observed subgraphs to the node embeddings that it has already optimized for in order to generalise to unseen nodes.

Both of our intervention strategies rely on a random walk that selects a collection of nodes that are highly similar to the original nodes but are not specific to any one type of node. Therefore, the random walk must be incorporated into the inductive functions if we are to successfully transfer our methods to the inductive framework.

Extension on the hypergraph: A hypergraph is a graph in which the edges, called hyperedges, can connect to more than two vertices or nodes. Some popular works on hypergraph includes [459, 460, 461]. Applications of GNN on the hypergraph can be found in [462, 463, 464]. According to [463] graph convolution is a special form of hypergraph convolution that occurs when a non-pairwise connection degenerates into a pairwise one. In the future, we are planning to extend our work on hypergraph setting by converting the "random walk for ordinary graph" to "random walk for hypergraph" [465].

7.2.3 Future Directions for Chapter 5

Some possible future works could be: spotting the set of brain regions for which the differences between the ADHD brain and the healthy brain are maximum, so that we can explain more accurately the reason behind the illness. Also, in this study, we computed the connectivity measures among various brain regions using the time series from various brain regions. We considered these time series as a whole to compute such measures. In the future, we will consider them in a small window and slide the window to capture more details while computing the connectivity measures. The accuracy of the diagnosis may thus be improved.

7.2.4 Future Directions for Chapter 6

In Chapter 6, we developed a mathematical model to study the progression of Alzheimer's disease from a healthy brain. Sometimes a simple mathematical model may unable to capture some hidden patterns of a complex process. Since brain is a complex system, thus in the future, we shall incorporate some deep generative models like variational autoencoders (VAE) [466] or generative adversarial networks (GAN) [467] in order to capture the hidden patterns more accurately and able to improve the accuracy of the simulation process.

Publications:

Conferences

- Anjan Chowdhury, Sriram Srinivasan, Sanjukta Bhowmick, Animesh Mukherjee, and Kuntal Ghosh. "Constant community identification in million scale networks using image thresholding algorithms." In Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'21), pp. 116-120. https://doi.org/10.1007/s13278-022-00895-8, 2021.

- Anjan Chowdhury, Swarup Chattopadhyay, and Kuntal Ghosh. "Analyzing the Progression of Alzheimer's Disease in Human Brain Networks." In Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'23), pp. 415-418. https://doi.org/10.1145/3625007.3627496, 2023.

- Anjan Chowdhury, Rajdeep Chatterjee, Geetanjali Aich, and Kuntal Ghosh. "ADHDNet: A DNN based Framework for Efficient ADHD Detection from fMRI Dataset." In 9th International Conference on Pattern Recognition and Machine Intelligence (PReMI'21), Lecture Notes in Computer Science. Springer, A. Ghosh et al. (Eds.): Volume 13102, Chapter 15, https://link.springer.com/book/9783031126994, 2024.

- Anjan Chowdhury, Swarup Chattopadhyay, Kuntal Ghosh, and Gautam Das. "NeuroANATOP: An Effective Tool to Assess the Progression of Alzheimer's disease in Human Brain Networks", Proceedings of 2023 IEEE International Conference On Knowledge Graph (ICKG), pp. 108-116, https://doi.org/10.1109/ICKG59574.2023.00019, 2023.

Journals

- Anjan Chowdhury, Sriram Srinivasan, Sanjukta Bhowmick, Animesh Mukherjee, and Kuntal Ghosh. "Constant community identification in million-scale networks." Social Network Analysis and Mining 12, no. 1 (2022): 70, https://doi.org/10.1007/s13278-022-00895-8, 2022.

- Anjan Chowdhury, Sriram Srinivasan, Sanjukta Bhowmick, Animesh Mukherjee, and Kuntal Ghosh. "Improving Node Classification Accuracy of GNN through Input and Output Intervention." The ACM Transactions on Knowledge Discovery from Data (TKDD), Volume 18, Issue 1, Article No. 17, pp. 1-31, https://doi.org/10.1145/3610535, 2023.

- Snigdha Agarwal, Adarsh Raj, Anjan Chowdhury, Rajdeep Chatterjee, Geetanjali Aich, and Kuntal Ghosh. "Investigating the Impact of Standard Brain Atlases and Connectivity Measures on the Accuracy of ADHD Detection from fMRI Data Using Deep Learning.", Multimedia Tools and Applications, Volume 83, pp. 67023–67057 https://doi.org/10.1007/s11042-023-17962-7, 2024.

Book Chapter

- Swarup Chattopadhyay, Anjan Chowdhury, and Kuntal Ghosh. "Application of Machine-Learning Techniques in the Development of Neighbourhood-Based Robust Recommender Systems." In Recommender Systems, pp. 203-233. CRC Press, https://doi.org/10.1201/9781003319122-13, 2023.

Appendix A

Diagnosis of ADHD Using Image Representation of Brain Connectivity Matrix

A.1 Introduction

In Chapter 5, we discussed how the Attention Deficit Hyperactivity disorder can be diagnosed using complex network and its representation. We saw that the complex network can be constructed from the brain connectivity matrix M, where the each cell $M[i][j]$ in this matrix represents the relationships between two brain regions i and j.

In this appendix, in a parallel approach, we treat the brain connectivity matrix M as an image representation which immediately fed into a 2D neural network (such as the 2D-CNN [377], ResNet-50 [468], EfficientNetB6 [469], etc) to diagnose ADHD. Fig. A.1 presents a thorough workflow.

A.2 Classification Approach Using Image

With this method, rather than constructing complex networks, we transformed the functional connectivity matrix into an image and then fed it directly into a two-dimensional

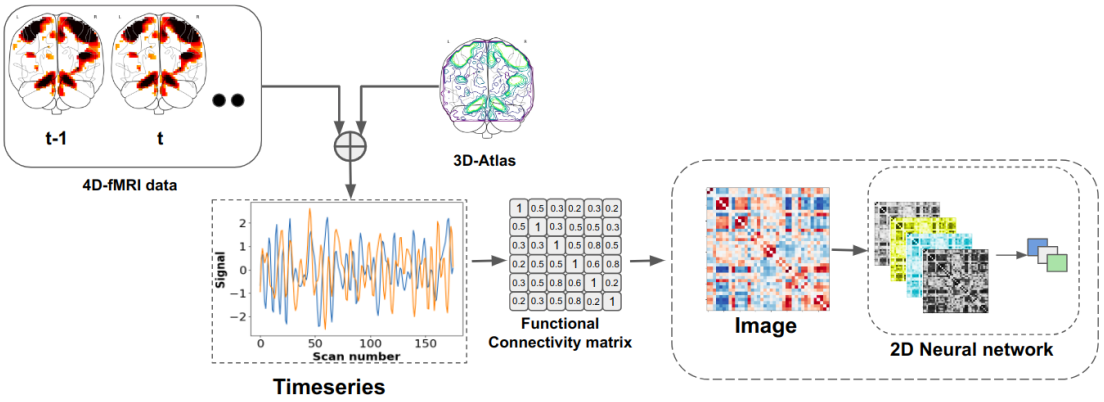


FIGURE A.1: Workflow Diagram.

(2D) neural network. This allowed us to avoid the complexity of the network generation process. The idea behind this is to represent the connectivity patterns that exist between all of the different regions with intensity values. The primary reason for this is to cut down on the additional work of manually extracting features and generating graphs.

A.3 Results

A.3.1 Results using image-based approach

The accuracy of the image-based approach is presented in Table A.1. Bold typefaces highlight information with the maximum accuracy. The comparative analysis indicates that the graph-based approach outperforms the image-based approach. EfficientNetB3 achieved peak accuracy of 86.67% and 71.87% for the KKI and NYU university datasets, respectively. The connectivity matrix used was Pearson-correlation, while the atlas employed was MSDL. The results obtained from the NI dataset indicate that ResNet-50 and EfficientNetB6 exhibit the highest level of accuracy, specifically 90.9%, when utilising the MSDL atlas and partial correlation. The results obtained from the Peking dataset indicate that the EfficientNets exhibit superior performance, achieving an accuracy rate of 75%. This outcome was observed when utilising the MSDL atlas and Pearson correlation. The EfficientNetB6 model exhibits improved accuracy in partial-correlation, achieving a rate of 75%.

University	Atlas	Connectivity matrix		Models		
			2D-CNN	ResNet-50	Efficient-NetB3	Efficient-NetB6
KKI	MSDL	CORR	80.00	73.33	**86.67**	86.60
		PAR-CORR	73.33	80.00	73.33	86.60
	Smith	CORR	76.92	79.61	61.53	46.15
		PAR-CORR	82.30	10.69	69.23	15.38
	Allen	CORR	69.23	53.84	76.92	76.92
		PAR-CORR	56.15	40.80	13.31	68.81
	Harvard-Oxford	CORR	73.58	60.06	48.68	51.58
		PAR-CORR	64.30	72.52	75.47	74.70
	BASC-64	CORR	78.80	68.44	58.10	80.00
		PAR-CORR	31.48	66.66	73.93	71.33
	BASC-444	CORR	65.85	65.85	42.83	72.22
		PAR-CORR	27.40	74.69	62.09	73.33
NYU	MSDL	CORR	51.22	65.85	**71.87**	68.75
		PAR-CORR	58.54	65.85	65.62	68.75
	Smith	CORR	43.75	56.25	31.25	62.48
		PAR-CORR	52.25	43.75	25.00	56.25
	Allen	CORR	37.30	53.65	68.44	42.74
		PAR-CORR	25.48	36.93	60.97	65.88
	Harvard-Oxford	CORR	48.61	60.83	48.37	46.34
		PAR-CORR	26.48	58.16	29.47	64.60
	BASC-64	CORR	58.53	51.21	69.82	59.92
		PAR-CORR	13.22	62.60	38.59	63.41
	BASC-444	CORR	57.61	46.80	59.76	43.14
		PAR-CORR	33.53	37.91	32.10	36.80
NI	MSDL	CORR	72.73	81.81	72.70	90.00
		PAR-CORR	72.73	**90.90**	81.82	**90.90**
	Smith	CORR	66.90	75.00	25.00	54.62
		PAR-CORR	41.50	25.00	50.00	22.33
	Allen	CORR	46.60	50.00	68.64	62.38
		PAR-CORR	53.40	64.14	63.21	65.10
	Harvard-Oxford	CORR	64.70	35.29	41.17	70.80
		PAR-CORR	36.36	41.17	58.82	35.29
	BASC-64	CORR	41.17	64.70	52.94	47.64
		PAR-CORR	57.10	47.05	29.41	76.47
	BASC-444	CORR	48.02	64.71	57.61	58.82
		PAR-CORR	42.90	73.69	32.13	58.82
Peking	MSDL	CORR	51.28	66.67	**75.00**	**75.00**
		PAR-CORR	58.98	64.11	68.00	**75.00**
	Smith	CORR	25.00	55.96	71.79	67.50
		PAR-CORR	61.53	49.24	58.97	51.83
	Allen	CORR	64.50	51.03	69.25	46.79
		PAR-CORR	36.79	32.22	40.45	61.20
	Harvard-Oxford	CORR	33.93	61.53	49.66	57.84
		PAR-CORR	56.41	43.58	60.14	69.56
	BASC-64	CORR	45.48	58.97	64.10	51.77
		PAR-CORR	33.60	31.33	53.84	47.93
	BASC-444	CORR	56.00	56.00	61.53	43.58
		PAR-CORR	53.84	52.50	69.23	61.53

TABLE A.1: Accuracy in image-based approach. We have used four datasets (universities). For each dataset, six atlases are used and for each atlases, two type of connectivity matrices are used. We also compared four 2D models: 2D-CNN, ResNet-50, EfficientNetB3 and EfficientNetB6. CORR stands for Pearson-correlation and PAR-CORR stands for Partial-correlation

A.3.2 Results using Combined Approach

In addition to utilising image-based and graph-based methodologies, we have implemented a hybrid approach that integrates both techniques. The feature vectors of the aforementioned methods were concatenated and subsequently used as input for several 1D models. In order to merge a 2D image with a 1D feature vector, it is necessary to first flatten the image and transform it into a 1D vector. Based on our observations, the combination of these two approaches did not yield a significant increase in accuracy. The accuracy values are presented in Table A.2.

A.3.3 Detailed comparisons with Baselines

The comparisons between our method and the baselines are presented in Table A.3. As can be seen, the graph-based method provides the highest level of accuracy for Peking and KKI, whereas the combined approach provides the highest level of accuracy for NYU. The technique that exclusively relies on images offers the highest level of accuracy for NI.

A.3.4 Observations and Explanations

Along with the previous finding, this image based and the combined approachs reveal several noteworthy observations.

- Similar to the graph based approach, The accuracy of the categorization of the image based approach fluctuates and is contingent upon the selection of a specific atlas, functional connectivity matrix, and model.
- According to the results, the graph-based methodology exhibits superior accuracy in comparison to the image-based approach.
- The efficacy of the combined approach in enhancing accuracy is either negligible or nonexistent.

One plausible explanation for the first observation could be attributed to the variations in the image as we modify the atlas or connectivity measure. Alterations to the

University	Atlas	Connectivity matrix		Models	
			1D-CNN	LSTM	ADHDNet
KKI	MSDL	CORR	80.00	80.70	**82.60**
		PAR-CORR	80.00	**82.60**	**82.60**
	Smith	CORR	78.41	81.00	80.00
		PAR-CORR	84.00	84.00	78.40
	Allen	CORR	65.27	75.48	72.15
		PAR-CORR	45.91	68.40	56.11
	Harvard-Oxford	CORR	72.34	65.15	62.22
		PAR-CORR	24.63	72.60	54.66
	BASC-64	CORR	57.69	58.64	59.65
		PAR-CORR	56.10	51.00	45.90
	BASC-444	CORR	46.47	10.34	49.75
		PAR-CORR	12.89	22.20	35.48
NYU	MSDL	CORR	**74.48**	69.50	32.72
		PAR-CORR	64.86	67.20	67.27
	Smith	CORR	30.40	48.38	61.55
		PAR-CORR	61.90	45.00	68.35
	Allen	CORR	45.37	56.70	46.60
		PAR-CORR	44.25	19.47	12.94
	Harvard-Oxford	CORR	37.89	35.48	16.50
		PAR-CORR	37.89	49.10	60.00
	BASC-64	CORR	57.65	54.19	62.48
		PAR-CORR	52.00	60.00	57.14
	BASC-444	CORR	42.30	60.13	59.48
		PAR-CORR	42.30	57.25	57.14
NI	MSDL	CORR	73.33	66.60	63.33
		PAR-CORR	66.66	22.20	18.50
	Smith	CORR	33.33	65.00	69.80
		PAR-CORR	28.57	22.20	51.60
	Allen	CORR	56.23	45.61	49.10
		PAR-CORR	68.61	62.34	36.79
	Harvard-Oxford	CORR	45.37	15.47	51.64
		PAR-CORR	12.30	59.48	26.40
	BASC-64	CORR	75.00	71.25	80.00
		PAR-CORR	50.00	69.14	61.81
	BASC-444	CORR	51.00	**81.81**	80.00
		PAR-CORR	64.21	25.34	36.18
Peking	MSDL	CORR	**86.11**	65.45	71.60
		PAR-CORR	73.14	58.19	45.15
	Smith	CORR	28.57	16.75	23.80
		PAR-CORR	76.14	14.28	33.33
	Allen	CORR	14.28	85.71	14.56
		PAR-CORR	61.90	14.28	71.42
	Harvard-Oxford	CORR	14.28	14.28	15.60
		PAR-CORR	19.04	19.04	33.33
	BASC-64	CORR	51.00	55.40	55.65
		PAR-CORR	51.00	55.40	52.00
	BASC-444	CORR	52.00	54.15	52.89
		PAR-CORR	53.14	54.20	55.40

TABLE A.2: Accuracy in combined approach. We have used four datasets (universities). For each dataset, six atlases are used and for each atlases, two type of connectivity matrices are used. We also compared three 1D models: 1D-CNN, LSTM and the proposed ADHDNet. CORR stands for Pearson-correlation and PAR-CORR stands for Partial-correlation.

	NYU	Peking	KKI	NI
Deep Learning				
FCNet [355] (2017)	58.50	68.70	-	60.00
3D-CNN [357] (2017)	71.50	-	72.80	-
Deep fMRI [437] (2018)	73.10	62.70	-	67.90
Proposed Method - I (best Graph-based)	73.50	87.50	89.65	86.36
Proposed Method - II (best image-based)	71.87	75.00	86.67	90.90
Proposed Method - III (best combined)	74.48	86.11	82.60	81.81

TABLE A.3: Accuracy comparison with baselines (%). We have used Tables 5.3, A.1 and A.2 and pick the best accuracies of the proposed methods.

atlas lead to modifications in the quantity of regions, which subsequently affect the dimensions of the image in an image-based methodology. A modification in the measure of connectivity leads to alterations in the strength of associations among various regions. This subsequently results in changes in the pixel intensity in the image-based approach.

One plausible explanation for the second observation is that the graph provides a greater amount of information than the image. Images solely contain spatial information, which is conveyed through the arrangement and intensity of pixels. However, information pertaining to the correlation between regions cannot be inferred from the image alone. Conversely, within a graphical representation, the existence of an edge denotes the correlation between nodes or regions within the brain, while the numerical value assigned to the edge signifies the magnitude of the correlation.

Concatenating the two pieces of data in the third observation may result in some noise being introduced into the feature space, lowering accuracy in some situations.

A.4 Discussion

In the image-based approach, we consider the functional connectivity matrix as an image and apply different two-dimensional deep learning models (2D-CNN, ResNet-50, EfficientNetB3, EfficientNetB6). From the results, we observed that the graph- or network-based approach shows better results than the image-based approach as well as the combined approach.

Appendix B

Investigating the Progression of the Brain Network: from a Healthy Brain to Mild Cognitive Impairment

B.1 Introduction

The condition known as mild cognitive impairment (MCI) causes a slight but noticeable decline in cognitive abilities such as memory and the ability to think. A person diagnosed with MCI has a significantly increased risk of developing Alzheimer's disease or another kind of dementia. On the other hand, mild cognitive impairment (MCI) is a stage of cognitive function that can be thought of as falling somewhere in the middle of those that are compatible with normal ageing and those that fit the criteria for Alzheimer's disease (AD) [470]. Past studies have revealed that between thirty and fifty percent of people who have MCI would develop Alzheimer's disease within three to five years after their diagnosis [471]. Thus in addition with AD, it is equally important to figure out the progression towards MCI from HC.

In this appendix, we discuss the results of our experiment using the same model that we have used in Chapter 6 to study the progression from HC to AD.

stat	*HC*	*MCI*
Number	62	64
Male/Female	27/35	34/30
Average age	73.78	75.81
Average weight	72.45	82.63

TABLE B.1: Structural statistics of participants belonging to healthy controls (*HC*) group and mild cognitive impairment (*MCI*) group.

B.2 Data Collection

We have collected 64 *MCI* fMRI data from the ADNI repository can be accessed at the following URL: http://adni.loni.ucla.edu. Following the downloading of the data, we separated all of the participants into two groups: those with healthy controls (*HC*) and those with mild cognitive impairment (*AD*). Table B.1 provides an illustration of the specifics of the clinical characteristics that differ between each of these two groups.

B.3 Results

B.3.1 Comparing real brain networks: G_{avg}^{HC} and G_{avg}^{MCI}

The present study conducts a comparative analysis of the topological properties of the actual brain networks G_{avg}^{HC} and G_{avg}^{MCI}, as illustrated in Fig. B.1. As depicted on the left of FIGURE B.1 displays a set of bars that have been plotted to represent G_{avg}^{HC}. The right figure represents G_{avg}^{MCI}. The results indicate that there is an increase in local efficiency, modularity, and global efficiency as the group transitions from *HC* to *MCI*. Conversely, the three remaining topological properties, namely average clustering coefficient, transitivity, and rich club coefficient, exhibit a decrease in the same direction. The aforementioned results indicate that the three cohorts, namely *HC* and *MCI*, exhibit distinct topological characteristics.

B.3.2 Simulation performance of the proposed variants

Comparing G_{syn}^{MCI} with G_{avg}^{MCI}. This section presents an analysis of the performance evaluation of our model and its baselines. Specifically, we compare the real and synthetic

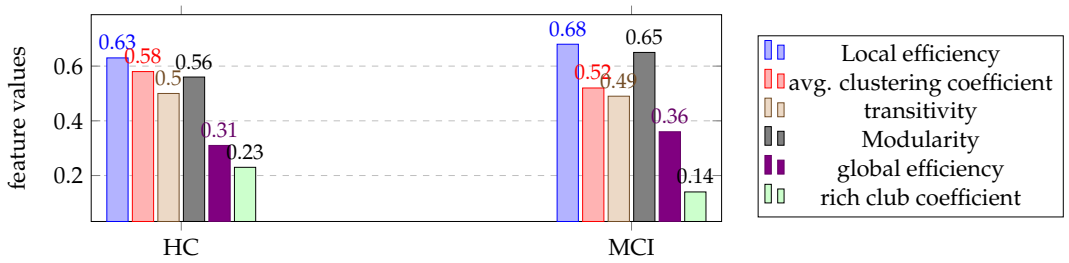
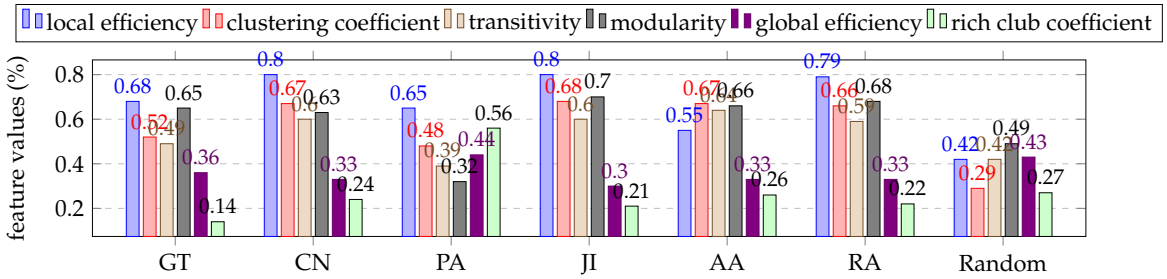


FIGURE B.1: Comparison of various topological properties among the real brain networks of healthy control (HC) and mild cognitive impairment (MCI) group.

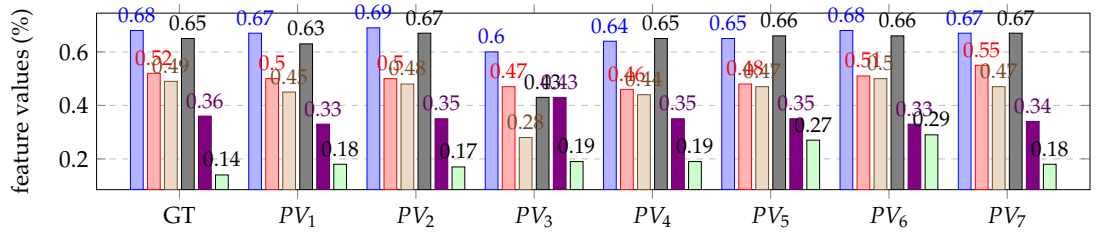
brain networks of individuals with Mild Cognitive Impairment (MCI). The bar plot in FIGURE B.2a displays six distinct topological features that have been produced by various competitive models, including the random model. The bar plot illustrating the topological features generated by the proposed variants in comparison to the ground truth is presented in FIGURE. B.2b. The group of bars depicted in the left section of FIGURE. B.2a and FIGURE. B.2b represent the ground truth (GT) network. It is evident that, in nearly all instances, the suggested alternatives demonstrate comparable topological characteristics when compared to the ground truths. The proposed variants, namely PV_2 (N2V) and PV_7 (GAT), exhibit superior performance relative to other variants with respect to accurately reproducing the majority of the topological characteristics that are closer to the ground truth. The results indicate that *Random* exhibits the lowest performance among the baselines, while AA ranks second in terms of performance. Conversely, the proposed variant demonstrates the highest level of performance. The performance of CN, JI, and RA is suboptimal in comparison to the proposed variants. The comparative analysis reveals that the efficacy of PA is superior to that of random; however, it is comparatively inferior to the performance of other competitive models.

B.3.3 Model performance evaluation: Modified Similarity Index (MSI)

Evaluating competitive models using MSI The relative errors of the topological properties of the synthetic networks generated and the ground truth network are presented in Table B.2. Furthermore, the MSI metric is calculated to distinctly distinguish the efficacy of the suggested variations in contrast to alternative approaches. The MSI values are presented in the final column of Table B.2.



(a) real *MCI* brain network (leftmost) vs synthetic *MCI* brain networks produced by the baselines.

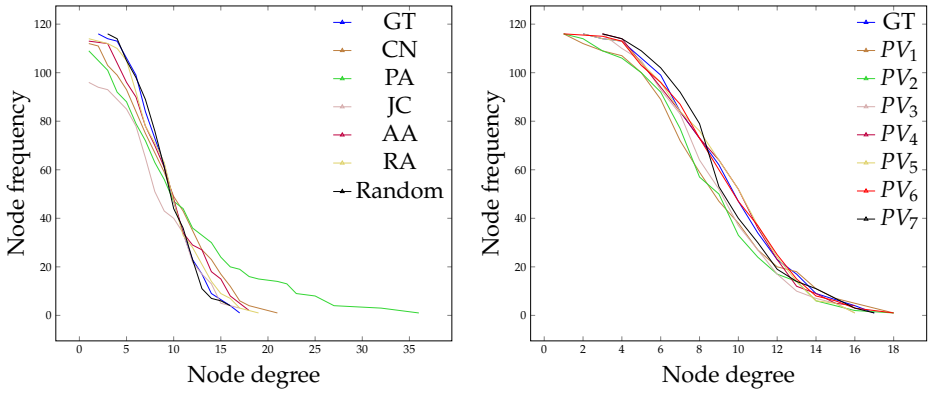


(b) real *MCI* brain network (leftmost) vs synthetic *MCI* brain networks produced by the proposed variants.

FIGURE B.2: Comparison of topological properties: real *MCI* brain network (leftmost) vs synthetic *MCI* brain networks produced by the various models. GT: Ground Truth, PV_1: Deep Walk (DW), PV_2: Node to Vec (N2V), PV_3: Line Embedding (LINE), PV_4: Graph Convolutional Neural Network (GCN), PV_5: GraphSAGE (SAGE), PV_6: ChebyNet (Cheb), PV_7: Graph Attention Network (GAT).

Models	E_{LE}	E_{AC}	E_M	E_T	E_{GE}	E_R	MSI
CN	0.12	0.15	0.02	0.11	0.10	0.03	0.65
JI	0.12	0.04	0.05	0.11	0.08	0.03	0.69
PA	0.04	0.03	0.04	0.1	0.42	0.08	0.58
RA	0.11	0.14	0.03	0.01	0.12	0.03	0.69
AA	0.15	0.01	0.01	0.15	0.07	0.03	0.70
PV_1	0.01	0.02	0.03	0.02	0.03	0.04	0.86
PV_2	0.01	0.02	0.01	0.02	0.01	0.03	0.91
PV_3	0.08	0.16	0.21	0.22	0.07	0.05	0.56
PV_4	0.04	0.06	0.05	0.0	0.01	0.05	0.83
PV_5	0.03	0.04	0.02	0.01	0.01	0.13	0.81
PV_6	0.0	0.01	0.01	0.01	0.03	0.15	0.83
PV_7	0.01	0.03	0.02	0.02	0.02	0.04	0.87
Random	0.23	0.16	0.16	0.07	0.13	0.13	0.53

TABLE B.2: Comparison of MSI-values for all the models (baselines and our proposed model). Larger MSI value indicate more similarity between the synthetic (MCI) and real (MCI) brain networks. red cell indicates the worst performer, green cell indicates the best performer and blue cell indicates the second best performer. CN: Common Neighbor, PA: Preferential Attachment, JI: Jaccard Index, AA: Adamic–Adar, RA: Resource Allocation. PV_1: Deep Walk (DW), PV_2: Node to Vec (N2V), PV_3: Line Embedding (LINE), PV_4: Graph Convolutional Neural Network (GCN), PV_5: GraphSAGE (SAGE), PV_6: ChebyNet (Cheb), PV_7: Graph Attention Network (GAT)



(a) Degree distribution of real and synthetic brain networks for *MCI* group: GT vs baselines

(b) Degree distribution of real and synthetic brain networks for *MCI* group: : GT vs proposed variants

FIGURE B.3: Comparing the complementary cumulative degree distribution of networks generated through various generative models. GT: Ground truth, CN: Common Neighbor, PA: Preferential Attachment, JI: Jaccard Index, AA: Adamic–Adar, RA: Resource Allocation, PV_1: Deep Walk (DW), PV_2: Node to Vec (N2V), PV_3: Line Embedding (LINE), PV_4: Graph Convolutional Neural Network (GCN), PV_5: GraphSAGE (SAGE), PV_6: ChebyNet (Cheb), PV_7: Graph Attention Network (GAT).

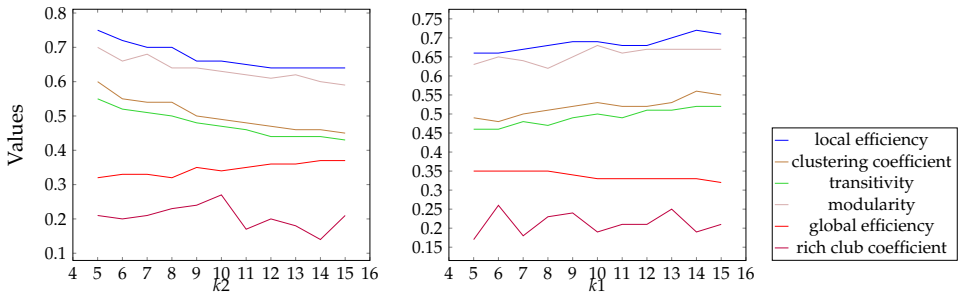
B.3.4 Degree Distribution

The experimental result shown in Fig. B.3 demonstrates that, when compared to the other baseline models, the degree distribution of the synthetic networks produced by the suggested variants (Fig. B.3b) closely resembles the degree distribution of the target networks. The JI, AA, and RA methods all perform well and are close to the targeted network. PA and CN methods outperform the proposed method and fail to reach the target networks. Thus, the empirical results support the proposed generative model's effectiveness in the progression of *HC* to *MCI*.

B.3.5 Sensitivity of parameters $k1$ and $k2$ of the proposed ensemble model

Fig. B.4a, shows the changes in network properties while producing G_{syn}^{MCI} from G_{avg}^{HC} by raising $k2$ from 5 to 15 while maintaining the value of $k1$ at 20.

Fig. B.4b shows the results of generating G_{syn}^{MCI} from G_{avg}^{HC}, while altering $k1$ from 15 to 25 and maintaining $k2$ to 8.



(a) G_{avg}^{HC} to $G_{syn}^{MCI}, k1 = 20, k2 \in [5, 15]$ (b) G_{avg}^{HC} to $G_{syn}^{MCI}, k1 \in [15, 25], k2 = 8$

FIGURE B.4: Changes in various topological properties in the generated synthetic networks from the real brain network upon tuning the parameters $k1$ and $k2$. G_{avg}^{HC} to G_{syn}^{MCI} represents generation of synthetic graph G_{syn}^{MCI} from G_{avg}^{HC}.

Bibliography

[1] Leonhard Euler. Commentarii academiae scientiarum petropolitanae. *Solutio problematis ad geometriam situs pertinentis*, 8:128–140, 1736.

[2] Wai-Kai Chen. *Graph theory and its engineering applications*, volume 5. World Scientific, 1997.

[3] Svante Janson, Andrzej Rucinski, and Tomasz Luczak. *Random graphs*. John Wiley & Sons, 2011.

[4] Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.

[5] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[6] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

[7] Stanley Wasserman and Katherine Faust. Social network analysis: Methods and applications. 1994.

[8] Cornelis J Stam and Jaap C Reijneveld. Graph theoretical analysis of complex networks in the brain. *Nonlinear biomedical physics*, 1(1):1–19, 2007.

[9] Thiago Alexandre Salgueiro Pardo, Lucas Antiqueira, Maria das Graças Volpe Nunes, Osvaldo N Oliveira, and Luciano Da Fontoura Costa. Using complex networks for language processing: The case of summary evaluation. In *2006 International Conference on Communications, Circuits and Systems*, volume 4, pages 2678–2682. IEEE, 2006.

[10] Mingming Fan and Yunsong Li. The application of computer graphics processing in visual communication design. *Journal of Intelligent & Fuzzy Systems*, 39(4):5183–5191, 2020.

[11] Zhan Su, Zuyi Lin, Jun Ai, and Hui Li. Rating prediction in recommender systems based on user behavior probability and complex network modeling. *IEEE Access*, 9:30739–30749, 2021.

[12] Andor Háznagy, István Fi, András London, and Tamás Németh. Complex network analysis of public transportation networks: A comprehensive study. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 371–378. IEEE, 2015.

[13] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99, 2002.

[14] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

[15] Donald Ervin Knuth. *The Stanford GraphBase: a platform for combinatorial computing*, volume 1. AcM Press New York, 1993.

[16] Xianchao Tang. Dolphin social network. figshare. dataset. 2014.

[17] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[18] Stefan Bornholdt and Heinz Georg Schuster. Handbook of graphs and networks. *From Genome to the Internet, Willey-VCH (2003 Weinheim)*, 2001.

[19] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[20] Hamed Sarvari, Ehab Abozinadah, Alex Mbaziira, and Damon McCoy. Constructing and analyzing criminal networks. In *2014 IEEE security and privacy workshops*, pages 84–91. IEEE, 2014.

[21] Marcel Salathé and James H Jones. Dynamics and control of diseases in networks with community structure. *PLoS computational biology*, 6(4):e1000736, 2010.

[22] Joel J Bechtel, William A Kelley, Teresa A Coons, M Gerry Klein, Daniel D Slagel, and Thomas L Petty. Lung cancer detection in patients with airflow obstruction identified in a primary care outpatient practice. *Chest*, 127(4):1140–1145, 2005.

[23] Mert Ozer, Nyunsu Kim, and Hasan Davulcu. Community detection in political twitter networks using nonnegative matrix factorization methods. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 81–88. IEEE, 2016.

[24] Frederic Guerrero-Solé. Community detection in political discussions on twitter: An application of the retweet overlap network method to the catalan process toward independence. *Social science computer review*, 35(2):244–261, 2017.

[25] Mohammad Javad Mosadegh and Mehdi Behboudi. Using social network paradigm for developing a conceptual framework in crm. *Australian Journal of Business and Management Research*, 1(4):63, 2011.

[26] Sabrine Ben Abdrabbah, Raouia Ayachi, and Nahla Ben Amor. Collaborative filtering based on dynamic community detection. *Dynamic Networks and Knowledge Discovery*, 85, 2014.

[27] Deepika Lalwani, Durvasula VLN Somayajulu, and P Radha Krishna. A community driven social recommendation system. In *2015 IEEE International conference on big data (big data)*, pages 821–826. IEEE, 2015.

[28] Yilmaz Atay, Ismail Koc, Ismail Babaoglu, and Halife Kodaz. Community detection from biological and social networks: A comparative analysis of metaheuristic algorithms. *Applied Soft Computing*, 50:194–211, 2017.

[29] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[30] Bergstrom Rosvall, M. Maps of random walks on complex networks reveal community structure. *PNAS*, 2008.

[31] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 9 2007.

[32] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2 2004.

[33] C. Bothorel J. D. Cruz and F. Poulet. Entropy based community detection in augmented social networks. *2011 International Conference on Computational Aspects of Social Networks (CASoN), Salamanca*, pages 163–168, 2011.

[34] Michihiro Kuramochi and George Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE transactions on Knowledge and Data Engineering*, 16(9):1038–1051, 2004.

[35] Drew Conway. Modeling network evolution using graph motifs. *arXiv preprint arXiv:1105.0902*, 2011.

[36] Van Thuy Hoang, Hyeon-Ju Jeon, Eun-Soon You, Yoewon Yoon, Sungyeop Jung, and O-Joun Lee. Graph representation learning and its applications: A survey. *Sensors*, 23(8):4168, 2023.

[37] Fix, Evelyn, and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review*, 57(3):238–247, 1989.

[38] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[39] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[40] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[41] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD (Special Interest Group on Knowledge Discovery and Data Mining) international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[42] Xin Liu, Tsuyoshi Murata, Kyoung-Sook Kim, Chatchawan Kotarasu, and Chenyi Zhuang. A general view for network embedding as matrix factorization. In *Proceedings of the Twelfth ACM international conference on web search and data mining*, pages 375–383, 2019.

[43] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR Í7, 2017.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *6th International Conference on Learning Representations*, 2017.

[45] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[46] Olaf Sporns. The human connectome: a complex network. *Annals of the new York Academy of Sciences*, 1224(1):109–125, 2011.

[47] Ellen Doernberg and Eric Hollander. Neurodevelopmental disorders (asd and adhd): Dsm-5, icd-10, and icd-11. *Journal of CNS spectrums*, 21(4):295–299, 2016.

[48] Frank Jessen, Rebecca E Amariglio, Martin Van Boxtel, Monique Breteler, Mathieu Ceccaldi, Gaël Chételat, Bruno Dubois, Carole Dufouil, Kathryn A Ellis, Wiesje M Van Der Flier, et al. A conceptual framework for research on subjective cognitive decline in preclinical alzheimer's disease. *Alzheimer's & dementia*, 10(6):844–852, 2014.

[49] Beatrice Heim, Florian Krismer, Roberto De Marzi, and Klaus Seppi. Magnetic resonance imaging for the diagnosis of parkinson's disease. *Journal of neural transmission*, 124:915–964, 2017.

[50] Harary Frank and Norman Robert Z. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 1960.

[51] Nikos K Logothetis and Brian A Wandell. Interpreting the bold signal. *Annu. Rev. Physiol.*, 66:735–769, 2004.

[52] C Vasudev. *Graph theory with applications*. New Age International, 2006.

[53] Sadegh Nobari, Xuesong Lu, Panagiotis Karras, and Stéphane Bressan. Fast random graph generation. In *Proceedings of the 14th international conference on extending database technology*, pages 331–342, 2011.

[54] Tomasz Luczak, Andrzej Ruciński, and Bernd Voigt. Ramsey properties of random graphs. *Journal of Combinatorial Theory, Series B*, 56(1):55–68, 1992.

[55] Mikhail Drobyshevskiy and Denis Turdakov. Random graph modeling: A survey of the concepts. *ACM computing surveys (CSUR)*, 52(6):1–36, 2019.

[56] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the national academy of sciences*, 99(suppl_1):2566–2572, 2002.

[57] Hanbaek Lyu, Facundo Mémoli, and David Sivakoff. Sampling random graph homomorphisms and applications to network data analysis. *Journal of Machine Learning Research*, 24(9):1–79, 2023.

[58] Alexei Vázquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Modeling of Protein Interaction Networks. *Complexus*, 1(1):38–44, 12 2002.

[59] Carlos J. Melián and Jordi Bascompte. Food web structure and habitat loss. *Ecology Letters*, 5(1):37–46, 2002.

[60] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[61] Jongkwang Kim and Thomas Wilhelm. What is a complex graph? *Physica A: Statistical Mechanics and its Applications*, 387(11):2637–2652, 2008.

[62] Nov 2023.

[63] Thomas Wilhelm and Rainer Brüggemann. Information theoretic measures for the maturity of ecosystems. *Integrative Systems Approaches to Natural and Social Dynamics: Systems Science 2000*, pages 263–273, 2001.

[64] Jens Christian Claussen. Offdiagonal complexity: A computationally quick complexity measure for graphs and networks. *Physica A: Statistical Mechanics and its Applications*, 375(1):365–373, 2007.

[65] Luıs A Nunes Amaral, Antonio Scala, Marc Barthelemy, and H Eugene Stanley. Classes of small-world networks. *Proceedings of the national academy of sciences*, 97(21):11149–11152, 2000.

[66] Philippe G. H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *J. ACM*, 21(4):569–575, 10 1974.

[67] Nick Roussopoulos. A max m, n algorithm for determining the graph h from its line graph g. *Information Processing Letters*, 2:108 – 112, 9 1973.

[68] Babu M. Madan Chalancon Guilhem, Kruse Kai. Clustering coefficient. *Springer New York*, 422, 2013.

[69] Junlong Zhang and Yu Luo. Degree centrality, betweenness centrality, and closeness centrality in social network. In *2017 2nd international conference on modelling, simulation and applied mathematics (MSAM2017)*, pages 300–303. Atlantis Press, 2017.

[70] Brian K Kay, Michael P Williamson, and Marius Sudol. The importance of being proline: the interaction of proline-rich motifs in signaling proteins with their cognate domains. *The FASEB journal*, 14(2):231–241, 2000.

[71] Jie Feng and Jing Xu. Identification of pathogenic genes and transcription factors in glaucoma. *Molecular medicine reports*, 20(1):216–224, 2019.

[72] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. In *International workshop on frontiers in algorithmics*, pages 186–195. Springer, 2008.

[73] Chaoqun Ni, Cassidy Sugimoto, and Jiepu Jiang. Degree, closeness, and betweenness: Application of group centrality measurements to explore macro-disciplinary evolution diachronically. In *Proceedings of ISSI*, volume 1, pages 1–13, 2011.

[74] Minoo Ashtiani, Ali Salehzadeh-Yazdi, Zahra Razaghi-Moghadam, Holger Hennig, Olaf Wolkenhauer, Mehdi Mirzaie, and Mohieddin Jafari. A systematic survey of centrality measures for protein-protein interaction networks. *BMC systems biology*, 12:1–17, 2018.

[75] Alicia S Arroyo, Romain Lannes, Eric Bapteste, and Iñaki Ruiz-Trillo. Gene similarity networks unveil a potential novel unicellular group closely related to animals from the tara oceans expedition. *Genome Biology and Evolution*, 12(9):1664–1678, 2020.

[76] Mahdi Jalili, Ali Salehzadeh-Yazdi, Shailendra Gupta, Olaf Wolkenhauer, Marjan Yaghmaie, Osbaldo Resendis-Antonio, and Kamran Alimoghaddam. Evolution of centrality measurements for the detection of essential proteins in biological networks. *Frontiers in physiology*, 7:375, 2016.

[77] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[78] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[79] Karthik Raman, Nandita Damaraju, and Govind Krishna Joshi. The organisational structure of protein networks: revisiting the centrality–lethality hypothesis. *Systems and Synthetic Biology*, 8:73–81, 2014.

[80] Abdulhadi Ibrahim H Bima, Ayman Zaky Elsamanoudy, Walaa F Albaqami, Zeenath Khan, Snijesh Valiya Parambath, Nuha Al-Rayes, Prabhakar Rao Kaipa, Ramu Elango, Babajan Banaganapalli, and Noor A Shaik. Integrative system biology and mathematical modeling of genetic networks identifies shared biomarkers for obesity and diabetes. *Mathematical biosciences and engineering: MBE*, 19(3):2310–2329, 2022.

[81] Emil Mededovic, Vaggelis G Douros, and Petri Mähönen. Node centrality metrics for hotspots analysis in telecom big data. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 417–422. IEEE, 2019.

[82] Britta Ruhnau. Eigenvector-centrality—a node-centrality? *Social networks*, 22(4):357–365, 2000.

[83] Jack McKay Fletcher and Thomas Wennekers. From structure to activity: Using centrality measures to predict neuronal activity. *International journal of neural systems*, 28(02):1750013, 2018.

[84] Cesi Cruz, Julien Labonne, and Pablo Querubin. Politician family networks and electoral outcomes: Evidence from the philippines. *American Economic Review*, 107(10):3006–3037, 2017.

[85] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[86] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.

[87] Bryan Ek, Caitlin VerSchneider, and Darren A Narayan. Global efficiency of graphs. *AKCE International Journal of Graphs and Combinatorics*, 12(1):1–13, 2015.

[88] Vito Latora and Massimo Marchiori. Is the boston subway a small-world network? *Physica a: statistical mechanics and its applications*, 314(1-4):109–113, 2002.

[89] Christopher J Honey, Rolf Kötter, Michael Breakspear, and Olaf Sporns. Network structure of cerebral cortex shapes functional connectivity on multiple time scales. *Proceedings of the National Academy of Sciences*, 104(24):10240–10245, 2007.

[90] Giorgio Gallo and Stefano Pallottino. Shortest path algorithms. *Annals of operations research*, 13(1):1–79, 1988.

[91] Vincent Labatut. Generalized measures for the evaluation of community detection methods. *Social and Information Networks*, 5 2016.

[92] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, 2008.

[93] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. On the permanence of vertices in network communities. In *KDD*, pages 1396–1405. ACM, 2014.

[94] Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*, 2011.

[95] Ka Yee Yeung and Walter L Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[96] Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. Metrics for community analysis: A survey. *ACM Computing Surveys (CSUR)*, 50(4):1–37, 2017.

[97] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Sanjukta Bhowmick, and Animesh Mukherjee. Constant communities in complex networks. *Scientific Reports*, 3(1):1825, 2013.

[98] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9:e15, 2020.

[99] Pearson Karl. On lines and planes of closest fit to systems of points in space. *LIII*, 1901.

[100] Soumya Dey, Ravishankar Rao, and Mubarak Shah. Exploiting the brain's network structure in identifying adhd subjects. *Frontiers in systems neuroscience*, 6:75, 11 2012.

[101] Sandra L Robinson and Rebecca J Bennett. A typology of deviant workplace behaviors: A multidimensional scaling study. *Academy of management journal*, 38(2):555–572, 1995.

[102] Oksana Samko, A David Marshall, and Paul L Rosin. Selection of the optimal parameter value for the isomap algorithm. *Pattern Recognition Letters*, 27(9):968–979, 2006.

[103] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[104] Mehrtash T Harandi, Conrad Sanderson, Sareh Shirazi, and Brian C Lovell. Graph embedding discriminant analysis on grassmannian manifolds for improved image set matching. In *CVPR 2011*, pages 2705–2712. IEEE, 2011.

[105] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[106] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 1067–1077, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[107] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008.

[108] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, volume 2015, pages 2111–2117, 2015.

[109] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Homophily, structure, and content augmented network representation learning. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 609–618. IEEE, 2016.

[110] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[111] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

[112] Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018.

[113] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[114] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[115] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1416–1424, 2018.

[116] Alison A Motsinger, Stephen L Lee, George Mellick, and Marylyn D Ritchie. Gpnn: Power studies and applications of a neural network method for detecting gene-gene interactions in studies of human disease. *BMC bioinformatics*, 7:1–10, 2006.

[117] Qiongkai Xu, Qing Wang, Chenchen Xu, and Lizhen Qu. Attentive graph-based recursive neural network for collective vertex classification. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2403–2406, 2017.

[118] Anne Trafton. Artificial intelligence yields new antibiotic, 2020.

[119] Walaa S Aburayan, Rayan Y Booq, Nouf S BinSaleh, Haya A Alfassam, Abrar A Bakr, Haitham A Bukhary, Essam J Alyamani, and Essam A Tawfik. The delivery of the novel drug 'halicin'using electrospun fibers for the treatment of pressure ulcer against pathogenic bacteria. *Pharmaceutics*, 12(12):1189, 2020.

[120] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[121] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2013.

[122] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[123] Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[124] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[125] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pages 1106–1114. PMLR, 2018.

[126] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[127] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[128] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[129] António dos Anjos. and Hamid Reza Shahbazkia. Bi-level image thresholding - a fast method. In *Proceedings of the First International Conference on Bio-inspired Systems and Signal Processing - Volume 2: BIOSIGNALS, (BIOSTEC 2008)*, pages 70–76. INSTICC, 2008.

[130] Nobuyuki Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[131] Ping-Sung Liao, Tse-Sheng Chen, and Pau-Choo Chung. A fast algorithm for multilevel thresholding. *J. Inf. Sci. Eng.*, 17(5):713–727, 2001.

[132] Laura Fratiglioni and Chengxuan Qiu. Prevention of common neurodegenerative disorders in the elderly. *Experimental gerontology*, 44(1-2):46–50, 2009.

[133] Bastiaan R Bloem, Michael S Okun, and Christine Klein. Parkinson's disease. *The Lancet*, 397(10291):2284–2303, 2021.

[134] Francis O Walker. Huntington's disease. *The Lancet*, 369(9557):218–228, 2007.

[135] Michael Rutter. Concepts of autism: a review of research. *Child Psychology & Psychiatry & Allied Disciplines*, 1968.

[136] Robert A McCutcheon, Tiago Reis Marques, and Oliver D Howes. Schizophrenia—an overview. *JAMA psychiatry*, 77(2):201–210, 2020.

[137] Rudy J Castellani, Raj K Rolston, and Mark A Smith. Alzheimer disease. *Disease-a-month: DM*, 56(9):484, 2010.

[138] Justin M Long and David M Holtzman. Alzheimer disease: an update on pathobiology and treatment strategies. *Cell*, 179(2):312–339, 2019.

[139] David S Knopman, Helene Amieva, Ronald C Petersen, Gäel Chételat, David M Holtzman, Bradley T Hyman, Ralph A Nixon, and David T Jones. Alzheimer disease. *Nature reviews Disease primers*, 7(1):33, 2021.

[140] Zeinab Breijyeh and Rafik Karaman. Comprehensive review on alzheimer's disease: causes and treatment. *Molecules*, 25(24):5789, 2020.

[141] Russell A Barkley and Kevin R Murphy. *Attention-deficit hyperactivity disorder: A clinical workbook*. Guilford Press, 2006.

[142] Stephen V Faraone, Joseph Biederman, Thomas Spencer, Tim Wilens, Larry J Seidman, Eric Mick, and Alysa E Doyle. Attention-deficit/hyperactivity disorder in adults: an overview. *Biological psychiatry*, 48(1):9–20, 2000.

[143] Amy Pipe, Nisha Ravindran, Angela Paric, Beth Patterson, Michael Van Ameringen, and Arun V Ravindran. Treatments for child and adolescent attention deficit hyperactivity disorder in low and middle-income countries: A narrative review. *Asian Journal of Psychiatry*, page 103232, 2022.

[144] Mark A Stein, Courtney Zulauf-McCurdy, and Lourdes M DelRosso. Attention deficit hyperactivity disorder medications and sleep. *Child and Adolescent Psychiatric Clinics*, 31(3):499–514, 2022.

[145] Natalie A Gonzalez, Navya Sakhamuri, Sreekartthik Athiyaman, Bhawna Randhi, Sai Dheeraj Gutlapalli, Jingxiong Pu, Maheen F Zaidi, Maithily Patel, Lakshmi Malvika Atluri, and Ana P Arcia Franchini. A systematic review of yoga and meditation for attention-deficit/hyperactivity disorder in children. *Cureus*, 15(3), 2023.

[146] Saeideh Goharinejad, Samira Goharinejad, Sadrieh Hajesmaeel-Gohari, and Kambiz Bahaadinbeigy. The usefulness of virtual, augmented, and mixed reality technologies in the diagnosis and treatment of attention deficit hyperactivity

disorder in children: an overview of relevant studies. *BMC psychiatry*, 22(1):1–13, 2022.

[147] Marinus T Vlaardingerbroek and Jacques A Boer. *Magnetic resonance imaging: theory and practice*. Springer Science & Business Media, 2013.

[148] CD Binnie and PF Prior. Electroencephalography. *Journal of Neurology, Neurosurgery & Psychiatry*, 57(11):1308–1319, 1994.

[149] Malcolm Proudfoot, Mark W Woolrich, Anna C Nobre, and Martin R Turner. Magnetoencephalography. *Practical neurology*, 14(5):336–343, 2014.

[150] DL Keene, S Whiting, and ECG Ventureyra. Electrocorticography. *Epileptic Disorders*, 2(1):57–64, 2000.

[151] Denis Le Bihan, Jean-François Mangin, Cyril Poupon, Chris A Clark, Sabina Pappata, Nicolas Molko, and Hughes Chabriat. Diffusion tensor imaging: concepts and applications. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 13(4):534–546, 2001.

[152] Paul M Matthews and Peter Jezzard. Functional magnetic resonance imaging. *Journal of Neurology, Neurosurgery & Psychiatry*, 75(1):6–12, 2004.

[153] Stefan J Teipel, Michel Grothe, Simone Lista, Nicola Toschi, Francesco G Garaci, and Harald Hampel. Relevance of magnetic resonance imaging for early detection and diagnosis of alzheimer disease. *Medical Clinics*, 97(3):399–424, 2013.

[154] Nancy C Andreasen, James C Ehrhardt, Victor W Swayze, Randall Jay Alliger, William TC Yuh, Gregg Cohen, and Steven Ziebell. Magnetic resonance imaging of the brain in schizophrenia: the pathophysiologic significance of structural abnormalities. *Archives of general psychiatry*, 47(1):35–44, 1990.

[155] Pietro Perona, Takahiro Shiota, and Jitendra Malik. Anisotropic diffusion. *Geometry-driven diffusion in computer vision*, pages 73–92, 1994.

[156] Seiji Ogawa and Tso-Ming Lee. Magnetic resonance imaging of blood vessels at high fields: in vivo and in vitro measurements and image simulation. *Magnetic resonance in medicine*, 16(1):9–18, 1990.

[157] Klaas E Stephan, Lee M Harrison, Will D Penny, and Karl J Friston. Biophysical models of fmri responses. *Current opinion in neurobiology*, 14(5):629–635, 2004.

[158] D Summers. Harvard whole brain atlas: www. med. harvard. edu/aanlib/home. html. *Journal of Neurology, Neurosurgery & Psychiatry*, 74(3):288–288, 2003.

[159] Edmund T Rolls, Chu-Chung Huang, Ching-Po Lin, Jianfeng Feng, and Marc Joliot. Automated anatomical labelling atlas 3. *Neuroimage*, 206:116189, 2020.

[160] Gaël Varoquaux, Alexandre Gramfort, Fabian Pedregosa, Vincent Michel, and Bertrand Thirion. Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In *Information Processing in Medical Imaging: 22nd International Conference, IPMI 2011, Kloster Irsee, Germany, July 3-8, 2011. Proceedings 22*, pages 562–573. Springer, 2011.

[161] Molly G Bright and Kevin Murphy. Is fmri "noise" really noise? resting state nuisance regressors remove variance with network structure. *NeuroImage*, 114:158–169, 2015.

[162] Jonathan D Power, Bradley L Schlaggar, and Steven E Petersen. Recent progress and outstanding issues in motion correction in resting state fmri. *Neuroimage*, 105:536–551, 2015.

[163] Júlia F Soares, Rodolfo Abreu, Ana Cláudia Lima, Sónia Batista, Lívia Sousa, Miguel Castelo-Branco, and João V Duarte. On the optimal strategy for tackling head motion in fmri data. In *BIOSIGNALS*, pages 306–313, 2021.

[164] Ronald Sladky, Karl J Friston, Jasmin Tröstl, Ross Cunnington, Ewald Moser, and Christian Windischberger. Slice-timing effects and their correction in functional mri. *Neuroimage*, 58(2):588–594, 2011.

[165] David B Parker and Qolamreza R Razlighi. The benefit of slice timing correction in common fmri preprocessing pipelines. *Frontiers in neuroscience*, 13:821, 2019.

[166] Chloe Hutton, Andreas Bork, Oliver Josephs, Ralf Deichmann, John Ashburner, and Robert Turner. Image distortion correction in fmri: a quantitative evaluation. *Neuroimage*, 16(1):217–240, 2002.

[167] Peter Jezzard and Robert S Balaban. Correction for geometric distortion in echo planar images from b0 field variations. *Magnetic resonance in medicine*, 34(1):65–73, 1995.

[168] J-F Mangin, Jessica Lebenberg, Sandrine Lefranc, Nicole Labra, Guillaume Auzias, M Labit, Miguel Guevara, Hartmut Mohlberg, Pauline Roca, Pamela Guevara, et al. Spatial normalization of brain images and beyond, 2016.

[169] Keith J Worsley and Karl J Friston. Analysis of fmri time-series revisited—again. *Neuroimage*, 2(3):173–181, 1995.

[170] Maxim Zaitsev, Julian Maclaren, and Michael Herbst. Motion artifacts in mri: A complex problem with many partial solutions. *Journal of Magnetic Resonance Imaging*, 42(4):887–901, 2015.

[171] Sadie E Yancey, David J Rotenberg, Fred Tam, Mark Chiew, Shawn Ranieri, L Biswas, Kevan JT Anderson, S Nicole Baker, Graham A Wright, and Simon J Graham. Spin-history artifact during functional mri: potential for adaptive correction. *Medical physics*, 38(8):4634–4646, 2011.

[172] Raimon HR Pruim, Maarten Mennes, Daan van Rooij, Alberto Llera, Jan K Buitelaar, and Christian F Beckmann. Ica-aroma: A robust ica-based strategy for removing motion artifacts from fmri data. *Neuroimage*, 112:267–277, 2015.

[173] Giulio Ferrazzi, Maria Kuklisova Murgasova, Tomoki Arichi, Christina Malamateniou, Matthew J Fox, Antonios Makropoulos, Joanna Allsop, Mary Rutherford, Shaihan Malik, Paul Aljabar, et al. Resting state fmri in the moving fetus: a robust framework for motion, bias field and spin history correction. *Neuroimage*, 101:555–568, 2014.

[174] John Ashburner and Karl J Friston. Rigid body registration. *Statistical parametric mapping: The analysis of functional brain images*, pages 49–62, 2007.

[175] Stephen C Strother. Evaluating fmri preprocessing pipelines. *IEEE Engineering in Medicine and Biology Magazine*, 25(2):27–41, 2006.

[176] Michal Mikl, Radek Mareček, Petr Hluštík, Martina Pavlicová, Aleš Drastich, Pavel Chlebus, Milan Brázdil, and Petr Krupa. Effects of spatial smoothing on fmri group inferences. *Magnetic resonance imaging*, 26(4):490–503, 2008.

[177] Shahab Saquib Sohail, Jamshed Siddiqui, and Rashid Ali. Feature extraction and analysis of online reviews for the recommendation of books using opinion mining technique. *Perspectives in Science*, 8:754–756, 2016.

[178] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. *arXiv preprint arXiv:1006.5278*, 2010.

[179] Shulong Chen and Yuxing Peng. Matrix factorization for recommendation with explicit and implicit feedback. *Knowledge-Based Systems*, 158:109–117, 2018.

[180] Hartmut Döhner, Elihu Estey, David Grimwade, Sergio Amadori, Frederick R Appelbaum, Thomas Büchner, Hervé Dombret, Benjamin L Ebert, Pierre Fenaux, Richard A Larson, et al. Diagnosis and management of aml in adults: 2017 eln recommendations from an international expert panel. *Blood, The Journal of the American Society of Hematology*, 129(4):424–447, 2017.

[181] Markus Zanker and Markus Jessenitschnig. Case-studies on exploiting explicit customer requirements in recommender systems. *User Modeling and User-Adapted Interaction*, 19(1):133–166, 2009.

[182] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[183] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[184] Shahab Saquib Sohail, Jamshed Siddiqui, and Rashid Ali. Classifications of recommender systems: A review. *Journal of Engineering Science & Technology Review*, 10(4), 2017.

[185] Mehrbakhsh Nilashi, Othman bin Ibrahim, and Norafida Ithnin. Hybrid recommendation approaches for multi-criteria collaborative filtering. *Expert Systems with Applications*, 41(8):3879–3900, 2014.

[186] Tulasi K Paradarami, Nathaniel D Bastian, and Jennifer L Wightman. A hybrid recommender system using artificial neural networks. *Expert Systems with Applications*, 83:300–313, 2017.

[187] Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. *Group recommender systems: An introduction*. Springer, 2018.

[188] Derek Bridge, Mehmet H Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(3):315–320, 2005.

[189] John K Tarus, Zhendong Niu, and Abdallah Yousif. A hybrid knowledge-based recommender system for e-learning based on ontology and sequential pattern mining. *Future Generation Computer Systems*, 72:37–48, 2017.

[190] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1):bbab340, 2022.

[191] Yifei Wang, Shiyang Chen, Guobin Chen, Ethan Shurberg, Hang Liu, and Pengyu Hong. Motif-based graph representation learning with application to chemical molecules. In *Informatics*, volume 10, page 8. MDPI, 2023.

[192] Rafaël Van Belle, Bart Baesens, and Jochen De Weerdt. Catchm: A novel network-based credit card fraud detection method using node representation learning. *Decision Support Systems*, 164:113866, 2023.

[193] Lingwen Liu, Guangqi Wen, Peng Cao, Tianshun Hong, Jinzhu Yang, Xizhe Zhang, and Osmar R Zaiane. Braintgl: A dynamic graph representation learning model for brain network analysis. *Computers in Biology and Medicine*, page 106521, 2023.

[194] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[195] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[196] Xuebin Zheng, Bingxin Zhou, Ming Li, Yu Guang Wang, and Junbin Gao. Mathnet: Haar-like wavelet multiresolution analysis for graph representation learning. *Knowledge-Based Systems*, page 110609, 2023.

[197] Ying Zhong and Chenze Huang. A dynamic graph representation learning based on temporal graph transformer. *Alexandria Engineering Journal*, 63:359–369, 2023.

[198] Ghazaleh Niknam, Soheila Molaei, Hadi Zare, David Clifton, and Shirui Pan. Graph representation learning based on deep generative gaussian mixture models. *Neurocomputing*, 523:157–169, 2023.

[199] Yinghan Shen, Xuhui Jiang, Zijian Li, Yuanzhuo Wang, Chengjin Xu, Huawei Shen, and Xueqi Cheng. Uniskgrep: A unified representation learning framework of social network and knowledge graph. *Neural Networks*, 158:142–153, 2023.

[200] Wei Ju, Yiyang Gu, Xiao Luo, Yifan Wang, Haochen Yuan, Huasong Zhong, and Ming Zhang. Unsupervised graph-level representation learning with hierarchical contrasts. *Neural Networks*, 158:359–368, 2023.

[201] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7797–7805, 2022.

[202] Zhenyu Hou, Yukuo Cen, Yuxiao Dong, Jie Zhang, and Jie Tang. Automated unsupervised graph representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[203] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[204] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.

[205] Van Thuy Hoang, Hyeon-Ju Jeon, Eun-Soon You, Yoewon Yoon, Sungyeop Jung, and O-Joun Lee. Graph representation learning and its applications: A survey. *Sensors*, 23(8), 2023.

[206] Lamia Nabil Mahdy, Kadry Ali Ezzat, Haytham H Elmousalami, Hassan Aboul Ella, and Aboul Ella Hassanien. Automatic x-ray covid-19 lung image classification system based on multi-level thresholding and support vector machine. *MedRxiv*, 2020.

[207] Zahid Iqbal, Muhammad Attique Khan, Muhammad Sharif, Jamal Hussain Shah, Muhammad Habib ur Rehman, and Kashif Javed. An automated detection and classification of citrus plant diseases using image processing techniques: A review. *Computers and Electronics in Agriculture*, 153:12–32, 2018.

[208] Javeria Amin, Muhammad Sharif, Mussarat Yasmin, and Steven Lawrence Fernandes. A distinctive approach in brain tumor detection and classification using mri. *Pattern Recognition Letters*, 139:118–127, 2020.

[209] Farid Garcia-Lamont, Jair Cervantes, Asdrúbal López, and Lisbeth Rodriguez. Segmentation of images by color features: A survey. *Neurocomputing*, 292:1–27, 2018.

[210] Anju Asokan and J Anitha. Change detection techniques for remote sensing applications: a survey. *Earth Science Informatics*, 12(2):143–160, 2019.

[211] Siddharth Singh Chouhan, Ajay Kaul, and Uday Pratap Singh. Soft computing approaches for image segmentation: a survey. *Multimedia Tools and Applications*, 77(21):28483–28537, 2018.

[212] F. Scarselli M. Gori A. C. Tsoi M. Hagenbuchner and G. Monfardini. The graph neural network model. *Applied Network Science*, 20(1):61–80, 2009.

[213] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5171–5181, Red Hook, NY, USA, 2018. Curran Associates Inc.

[214] Boning Li, Yingce Xia, Shufang Xie, Lijun Wu, and Tao Qin. Distance-enhanced graph neural network for link prediction. *ICML 2021 Workshop on Computational Biology*, 2021.

[215] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. NIPS'16, page 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc.

[216] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Le Song. *Graph Neural Networks*, pages 27–37. Springer Nature Singapore, Singapore, 2022.

[217] Tamara T Mueller, Johannes C Paetzold, Chinmay Prabhakar, Dmitrii Usynin, Daniel Rueckert, and Georgios Kaissis. Differentially private graph classification with gnns. *arXiv preprint arXiv:2202.02575*, 2022.

[218] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs, 2018.

[219] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[220] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020.

[221] Jianyong Sun, Wei Zheng, Qingfu Zhang, and Zongben Xu. Graph neural network encoding for community detection in attribute networks. *IEEE Transactions on Cybernetics*, 2021.

[222] Linhao Luo, Yixiang Fang, Xin Cao, Xiaofeng Zhang, and Wenjie Zhang. Detecting communities from heterogeneous graphs: A context path-based graph neural network model. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1170–1180, 2021.

[223] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks, 2019.

[224] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260*, 2020.

[225] Stavros Souravlas, Sofia Anastasiadou, and Stefanos Katsavounis. A survey on the recent advances of deep community detection. *Applied Sciences*, 11(16):7179, 2021.

[226] Joan Bruna and X Li. Community detection with graph neural networks. *stat*, 1050:27, 2017.

[227] Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*, 2019.

[228] Atefeh Moradan, Andrew Draganov, Davide Mottin, and Ira Assent. Ucode: Unified community detection with graph convolutional networks. *arXiv preprint arXiv:2112.14822*, 2021.

[229] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[230] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[231] Mohamed EL-MOUSSAOUI, Tarik AGOUTI, Abdessadek TIKNIOUINE, and Mohamed EL ADNANI. A comprehensive literature review on community detection: Approaches and applications. *Procedia Computer Science*, 151:295 – 302, 2019. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

[232] Maria A. Riolo and M. E. J. Newman. Consistency of community structure in complex networks. *Phys. Rev. E*, 101(5), 5 2020.

[233] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Sanjukta Bhowmick, and Animesh Mukherjee. Constant communities in complex networks. *Scientific Reports*, 3(1), 2013.

[234] Andrea Lancichinetti and Santo Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2(1), 3 2012.

[235] Lucas G. S. Jeub, Olaf Sporns, and Santo Fortunato. Multiresolution consensus clustering in networks, 2017.

[236] Valérie Poulin and François Théberge. Ensemble clustering for graphs: comparisons and applications. *Applied Network Science*, 4(1):51, 2019.

[237] T. Chakraborty, N. Park, and V. S. Subrahmanian. Ensemble-based algorithms to detect disjoint and overlapping communities in networks, 8 2016.

[238] Tandon Aditya, Albeshri Aiiad, Thayananthan Vijey, Alhalabi Wadee, and Fortunato Santo. Fast consensus clustering in complex networks. *Phys. Rev. E*, (4), 4 2019.

[239] Weir William, Emmons Scott, Gibson Ryan, Taylor Dane, and Mucha Peter. Post-processing partitions to identify domains of modularity optimization. *Algorithms*, 10(3), 2017.

[240] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46):4, 1993.

[241] Jae Dong Noh and Heiko Rieger. Random walks on complex networks. *Physical Review Letters*, 92(11), 3 2004.

[242] Jia-Yu Pan, Hyung-jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 07 2004.

[243] Xiao-Ming Wu, Zhenguo Li, Anthony Man-Cho So, John Wright, and Shih-Fu Chang. Learning with partially absorbing random walks. NIPS'12, page 3077–3085, Red Hook, NY, USA, 2012. Curran Associates Inc.

[244] Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. Towards scaling fully personalized pageRank: algorithms, lower bounds, and experiments. *Internet Math.*, 2(3):333–358, 2005.

[245] Jianbing Shen, Yunfan Du, Wenguan Wang, and Xuelong Li. Lazy random walks for superpixel segmentation. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 23:1451–62, 04 2014.

[246] Giorgio Valentini, Elena Casiraghi, Luca Cappelletti, Vida Ravanmehr, Tommaso Fontana, Justin Reese, and Peter Robinson. Het-node2vec: second order random walk sampling for heterogeneous multigraphs embedding, 2021.

[247] Marco Gori, Augusto Pucci, V Roma, and I Siena. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*, volume 7, pages 2766–2771, 2007.

[248] Marco Gori and Augusto Pucci. Research paper recommender systems: A random-walk based approach. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 778–781. IEEE, 2006.

[249] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (europhysics Letters)*, 89(5):58007, 2010.

[250] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 8 2014.

[251] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[252] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. ICML'03, page 912–919. AAAI Press, 2003.

[253] Purnamrita Sarkar and Andrew W Moore. Random walks in social networks and their applications: a survey. In *Social Network Data Analytics*, pages 43–77. Springer, 2011.

[254] Raffaella Burioni and Davide Cassi. Random walks on graphs: ideas, techniques and results. *Journal of Physics A: Mathematical and General*, 38(8):R45, 2005.

[255] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, and Xiangjie Kong. Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2):95–107, 2019.

[256] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications, 2017. cite arxiv:1709.05584Comment: Published in the IEEE Data Engineering Bulletin, September 2017; version with minor corrections.

[257] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[258] Zexi Huang, Arlei Silva, and Ambuj Singh. A broader picture of random-walk based graph embedding. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 685–695, 2021.

[259] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 459–467, 2018.

[260] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 258–265, 2017.

[261] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. Netsmf: Large-scale network embedding as sparse matrix factorization. In *The World Wide Web Conference*, pages 1509–1520, 2019.

[262] Michael T Schaub, Jean-Charles Delvenne, Renaud Lambiotte, and Mauricio Barahona. Multiscale dynamical embeddings of complex networks. *Physical Review E*, 99(6):062308, 2019.

[263] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research*, 23(89):1–64, 2022.

[264] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[265] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE transactions on knowledge and data engineering*, 31(5):833–852, 2018.

[266] Yaojing Wang, Yuan Yao, Hanghang Tong, Feng Xu, and Jian Lu. A brief review of network embedding. *Big Data Mining and Analytics*, 2(1):35–47, 2018.

[267] Semi-supervised learning. 2006.

[268] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[269] Yanwen Chong, Yun Ding, Qing Yan, and Shaoming Pan. Graph-based semi-supervised learning: A review. *Neurocomputing*, 408:216–230, 2020.

[270] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 912–919, Washington, DC, USA, 2003. AAAI Press.

[271] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 321–328, Cambridge, MA, USA, 2003. MIT Press.

[272] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[273] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *International workshop on artificial intelligence and statistics*, pages 57–64. PMLR, 2005.

[274] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 290–297, 2003.

[275] Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1):209–239, 2004.

[276] Franca Hoffmann, Bamdad Hosseini, Zhi Ren, and Andrew M Stuart. Consistency of semi-supervised learning algorithms on graphs: Probit and one-hot methods. *Journal of Machine Learning Research*, 21:1–55, 2020.

[277] Ronan Collobert, Fabian Sinz, Jason Weston, Léon Bottou, and Thorsten Joachims. Large scale transductive svms. *Journal of Machine Learning Research*, 7(8), 2006.

[278] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. *Advances in neural information processing systems*, 14, 2001.

[279] Xiao-Ming Wu, Zhenguo Li, Anthony So, John Wright, and Shih-Fu Chang. Learning with partially absorbing random walks. *Advances in neural information processing systems*, 25, 2012.

[280] Zhen Wang, Long Zhang, Rong Wang, Feiping Nie, and Xuelong Li. Semi-supervised learning via bipartite graph construction with adaptive neighbors. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[281] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. *Advances in neural information processing systems*, 22, 2009.

[282] Kai Zhang, James T Kwok, and Bahram Parvin. Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1233–1240, 2009.

[283] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*, pages 624–638. Springer, 2004.

[284] Xueyuan Zhou and Mikhail Belkin. Semi-supervised learning by higher order regularization. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 892–900. JMLR Workshop and Conference Proceedings, 2011.

[285] Jian Li, Yong Liu, Rong Yin, and Weiping Wang. Approximate manifold regularization: Scalable algorithm and generalization analysis. In *IJCAI*, pages 2887–2893, 2019.

[286] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006.

[287] Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[288] Yunsheng Song, Jing Zhang, and Chao Zhang. A survey of large-scale graph-based semi-supervised classification algorithms. *International Journal of Cognitive Computing in Engineering*, 3:188–198, 2022.

[289] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[290] Z. Li et al. Mv-gcn: Multi-view graph convolutional networks for link prediction. *in IEEE Access*, 7:176317–176328, 2019.

[291] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[292] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In Dale Schuurmans and Michael P. Wellman, editors, *AAAI*, pages 1145–1152. AAAI Press, 2016.

[293] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019.

[294] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2224–2232, Cambridge, MA, USA, 2015. MIT Press.

[295] Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[296] Q. Li, Z. Han, and X.-M. Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI, 2018.

[297] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[298] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. *When Does Label Smoothing Help?* Curran Associates Inc., Red Hook, NY, USA, 2019.

[299] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[300] Kun Zhan and Chaoxi Niu. Mutual teaching for graph convolutional networks. *Future Gener. Comput. Syst.*, 115:837–843, 2021.

[301] Xiao Luo, Wei Ju, Meng Qu, Chong Chen, Minghua Deng, Xian-Sheng Hua, and Ming Zhang. Dualgraph: Improving semi-supervised graph classification via dual contrastive learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 699–712, 2022.

[302] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[303] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 820–828, Red Hook, NY, USA, 2016. Curran Associates Inc.

[304] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[305] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020.

[306] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[307] Wei Ju, Xiao Luo, Zeyu Ma, Junwei Yang, Minghua Deng, and Ming Zhang. Ghnn: Graph harmonic neural networks for semi-supervised graph-level classification. *Neural Networks*, 151:70–79, 2022.

[308] Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *International Conference on Machine Learning*, pages 1576–1586. PMLR, 2020.

[309] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific reports*, 3(1):1–14, 2013.

[310] Antoine Scherrer, Pierre Borgnat, Eric Fleury, J-L Guillaume, and Céline Robardet. Description and simulation of dynamic mobility networks. *Computer Networks*, 52(15):2842–2858, 2008.

[311] Mahdi Jalili, Yasin Orouskhani, Milad Asgari, Nazanin Alipourfard, and Matjaž Perc. Link prediction in multiplex online social networks. *Royal Society open science*, 4(2):160863, 2017.

[312] Mark Ed Newman, Albert-László Ed Barabási, and Duncan J Watts. The structure and dynamics of networks. 2006.

[313] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[314] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

[315] Liming Pan, Tao Zhou, Linyuan Lü, and Chin-Kun Hu. Predicting missing links and identifying spurious links via likelihood analysis. *Scientific reports*, 6(1):1–10, 2016.

[316] Lin Zhu, Su-Ping Deng, Zhu-Hong You, and De-Shuang Huang. Identifying spurious interactions in the protein-protein interaction networks using local similarity preserving embedding. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(2):345–352, 2015.

[317] Swarup Chattopadhyay and CA Murthy. Generation of power-law networks by employing various attachment schemes: Structural properties emulating real world networks. *Information Sciences*, 397:219–242, 2017.

[318] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009.

[319] Olaf Sporns. Structure and function of complex brain networks. *Dialogues in clinical neuroscience*, 15(3):247, 2013.

[320] Andrea Avena-Koenigsberger, Bratislav Misic, and Olaf Sporns. Communication dynamics in complex brain networks. *Nature reviews neuroscience*, 19(1):17–33, 2018.

[321] Danielle S Bassett and Olaf Sporns. Network neuroscience. *Nature neuroscience*, 20(3):353–364, 2017.

[322] Chiyu Feng, Ahmed Elazab, Peng Yang, Tianfu Wang, Feng Zhou, Huoyou Hu, Xiaohua Xiao, and Baiying Lei. Deep learning framework for alzheimer's disease diagnosis via 3d-cnn and fsbi-lstm. *IEEE Access*, 7:63605–63618, 2019.

[323] Shangran Qiu, Prajakta S Joshi, Matthew I Miller, Chonghua Xue, Xiao Zhou, Cody Karjadi, Gary H Chang, Anant S Joshi, Brigid Dwyer, Shuhan Zhu, et al. Development and validation of an interpretable deep learning framework for alzheimer's disease classification. *Brain*, 143(6):1920–1933, 2020.

[324] Mohsin Raza, Muhammad Awais, Waqas Ellahi, Nauman Aslam, Huan Xuan Nguyen, and Hoa Le-Minh. Diagnosis and monitoring of alzheimer's patients using classical and deep learning techniques. *Expert Systems with Applications*, 136:353–364, 2019.

[325] Fabrizio Vecchio, Francesca Miraglia, Francesca Piludu, Giuseppe Granata, Roberto Romanello, Massimo Caulo, Valeria Onofrj, Placido Bramanti, Cesare Colosimo, and Paolo Maria Rossini. "small world" architecture in brain connectivity and hippocampal volume in alzheimer's disease: a study via graph theory from eeg data. *Brain imaging and behavior*, 11(2):473–485, 2017.

[326] Hao Jiang, Peng Cao, MingYi Xu, Jinzhu Yang, and Osmar Zaiane. Hi-gcn: A hierarchical graph convolution network for graph embedding learning of brain network and brain disorders prediction. *Computers in Biology and Medicine*, 127:104096, 2020.

[327] Ramesh Kumar Lama and Goo-Rak Kwon. Diagnosis of alzheimer's disease using brain network. *Frontiers in Neuroscience*, 15:15, 2021.

[328] Ju Xiang, Ning-Rui Zhang, Jia-Shuai Zhang, Xiao-Yi Lv, and Min Li. Prgefne: predicting disease-related genes by fast network embedding. *Methods*, 192:3–12, 2021.

[329] Klaus Hahn, Nicholas Myers, Sergei Prigarin, Karsten Rodenacker, Alexander Kurz, Hans Förstl, Claus Zimmer, Afra M Wohlschläger, and Christian Sorg. Selectively and progressively disrupted structural connectivity of functional brain networks in alzheimer's disease—revealed by a novel framework to analyze edge distributions of networks detecting disruptions with strong statistical evidence. *Neuroimage*, 81:96–109, 2013.

[330] Mark Mapstone, Amrita K Cheema, Massimo S Fiandaca, Xiaogang Zhong, Timothy R Mhyre, Linda H MacArthur, William J Hall, Susan G Fisher, Derick R Peterson, James M Haley, et al. Plasma phospholipids identify antecedent memory impairment in older adults. *Nature medicine*, 20(4):415–418, 2014.

[331] Chun-Yi Lo, Pei-Ning Wang, Kun-Hsien Chou, Jinhui Wang, Yong He, and Ching-Po Lin. Diffusion tensor tractography reveals abnormal topological organization in structural cortical networks in alzheimer's disease. *Journal of Neuroscience*, 30(50):16876–16885, 2010.

[332] Martin Dyrba, Michael Ewers, Martin Wegrzyn, Ingo Kilimann, Claudia Plant, Annahita Oswald, Thomas Meindl, Michela Pievani, Arun LW Bokde, Andreas Fellgiebel, et al. Robust automated detection of microstructural white matter degeneration in alzheimer's disease using machine learning classification of multicenter dti data. *PloS one*, 8(5):e64925, 2013.

[333] A Veeramuthu, S Meenakshi, and V Priya Darsini. Brain image classification using learning machine approach and brain structure analysis. *Procedia Computer Science*, 50:388–394, 2015.

[334] Lu Meng and Jing Xiang. Brain network analysis and classification based on convolutional neural network. *Frontiers in computational neuroscience*, page 95, 2018.

[335] Laura Dubreuil-Vall, Giulio Ruffini, and Joan A Camprodon. Deep learning convolutional neural networks discriminate adult adhd from healthy individuals on the basis of event-related spectral eeg. *Frontiers in neuroscience*, 14:251, 2020.

[336] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[337] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[338] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, Dipanjan Sengupta, Michael W Cole, Nicholas B Turk-Browne, and Philip S Yu. Deep graph similarity learning for brain data analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2743–2751, 2019.

[339] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and Philip S Yu. Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, 35(3):688–725, 2021.

[340] Carrie Morris and Islem Rekik. Autism spectrum disorder diagnosis using sparse graph embedding of morphological brain networks. pages 12–20, 2017.

[341] Yihong Zhao, Huaihou Chen, and R. Ogden. Wavelet-based weighted lasso and screening approaches in functional linear regression. *Journal of Computational and Graphical Statistics*, 24:00–00, 11 2014.

[342] Hüseyin Öztoprak, Mehmet Toycan, Yaşar Kemal Alp, Orhan Arıkan, Elvin Doğutepe, and Sirel Karakaş. Machine-based learning system: Classification of adhd and non-adhd participants. In *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2017.

[343] Marta Nuñez-Garcia, Sonja Simpraga, Maria Angeles Jurado, Maite Garolera, Roser Pueyo, and Laura Igual. Fadr: Functional-anatomical discriminative regions for rest fmri characterization. In Luping Zhou, Li Wang, Qian Wang, and Yinghuan Shi, editors, *Machine Learning in Medical Imaging*, pages 61–68, Cham, 2015. Springer International Publishing.

[344] Atif Riaz, Muhammad Asad, Eduardo Alonso, and Greg Slabaugh. Fusion of fmri and non-imaging data for adhd classification. *Computerized Medical Imaging and Graphics*, 65:115–128, 2018. Advances in Biomedical Image Processing.

[345] Yibin Tang, Chun Wang, Ying Chen, Ning Sun, Aimin Jiang, and Zhishun Wang. Identifying adhd individuals from resting-state functional connectivity using subspace clustering and binary hypothesis testing. *Journal of attention disorders*, 25(5):736–748, 2021.

[346] Soumya Dey, Ravishankar Rao, and Mubarak Shah. Attributed graph distance measure for automatic detection of attention deficit hyperactive disordered subjects. *Frontiers in neural circuits*, 8:64, 06 2014.

[347] Anderson Siqueira, Claudinei Biazoli, William Comfort, Luis Rohde, and João Sato. Abnormal functional resting-state networks in adhd: Graph theory and pattern recognition analysis of fmri data. *BioMed research international*, 2014:380531, 08 2014.

[348] Ilke Öztekin, Mark A Finlayson, Paulo A Graziano, and Anthony S Dick. Is there any incremental benefit to conducting neuroimaging and neurocognitive assessments in the diagnosis of adhd in young children? a machine learning investigation. *Developmental cognitive neuroscience*, 49:100966, 2021.

[349] Bo Miao, LL Zhang, JL Guan, QF Meng, and YL Zhang. Classification of adhd individuals and neurotypicals using reliable relief: A resting-state study. *IEEE Access*, 7:62163–62171, 2019.

[350] Shuaiqi Liu, Ling Zhao, Xu Wang, Qi Xin, Jie Zhao, David S Guttery, and Yu-Dong Zhang. Deep spatio-temporal representation and ensemble classification for attention deficit/hyperactivity disorder. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:1–10, 2020.

[351] Hui Tian Tor, Chui Ping Ooi, Nikki SJ Lim-Ashworth, Joel Koh En Wei, V Jahmunah, Shu Lih Oh, U Rajendra Acharya, and Daniel Shuen Sheng Fung. Automated detection of conduct disorder and attention deficit hyperactivity disorder using decomposition and nonlinear techniques with eeg signals. *Computer Methods and Programs in Biomedicine*, 200:105941, 2021.

[352] Atif Riaz, Muhammad Asad, Eduardo Alonso, and Greg Slabaugh. Fusion of fmri and non-imaging data for adhd classification. *Computerized Medical Imaging and Graphics*, 65:115–128, 2018. Advances in Biomedical Image Processing.

[353] Delbert Dueck. *Affinity propagation: clustering data by passing messages*. University of Toronto Toronto, ON, Canada, 2009.

[354] A. Kautzky, Thomas Vanicek, Clement Philippe, G. Kranz, Wolfgang Wadsak, Markus Mitterhauser, A. Hartmann, A. Hahn, Marcus Hacker, D. Rujescu, S. Kasper, and Rupert Lanzenberger. Machine learning classification of adhd and hc by multimodal serotonergic data. *Translational Psychiatry*, 10:104, 04 2020.

[355] Atif Riaz, Muhammad Asad, SM Al-Arif, Eduardo Alonso, Danai Dima, Philip Corr, and Greg Slabaugh. Fcnet: a convolutional neural network for calculating functional connectivity from functional mri. In *International Workshop on Connectomics in Neuroimaging*, pages 70–78. Springer, 2017.

[356] Tao Zhang, Cunbo Li, Peiyang Li, Yueheng Peng, Xiaodong Kang, Chenyang Jiang, Fali Li, Xuyang Zhu, Dezhong Yao, Bharat Biswal, et al. Separated channel attention convolutional neural network (sc-cnn-attention) to identify adhd in multi-site rs-fmri dataset. *Entropy*, 22(8):893, 2020.

[357] Liang Zou, Jiannan Zheng, Chunyan Miao, Martin J Mckeown, and Z Jane Wang. 3d cnn based automatic diagnosis of attention deficit hyperactivity disorder using functional and structural mri. *Ieee Access*, 5:23626–23636, 2017.

[358] Zhenyu Mao, Yi Su, Guangquan Xu, Xueping Wang, Yu Huang, Weihua Yue, Li Sun, and Naixue Xiong. Spatio-temporal deep learning method for adhd fmri classification. *Information Sciences*, 499:1–11, 2019.

[359] Laura Dubreuil-Vall, Giulio Ruffini, and Joan A. Camprodon. Deep learning convolutional neural networks discriminate adult adhd from healthy individuals on the basis of event-related spectral eeg. *Frontiers in Neuroscience*, 14:251, 2020.

[360] He Chen, Yan Song, and Xiaoli Li. Use of deep learning to detect personalized spatial-frequency anomalies in eegs of children with adhd. *Journal of Neural Engineering*, 16, 08 2019.

[361] Tosun M. Effects of spectral features of eeg signals recorded with different channels and recording statuses on adhd classification with deep learning. *Physical and engineering sciences in medicine*, 44:693–702, 2021.

[362] Zhenyu Mao, Yi Su, Guangquan Xu, Xueping Wang, Yu Huang, Weihua Yue, Li Sun, and Naixue Xiong. Spatio-temporal deep learning method for adhd fmri classification. *Information Sciences*, 499:1–11, 2019.

[363] Chengfeng Dou, Shikun Zhang, Hanping Wang, Li Sun, Yu Huang, and Weihua Yue. Adhd fmri short-time analysis method for edge computing based on multi-instance learning. *Journal of Systems Architecture*, 111:101834, 2020.

[364] Vikas Khullar, Karuna Salgotra, Harjit Pal Singh, and Davinder Pal Sharma. Deep learning-based binary classification of adhd using resting state mr images. *Augmented Human Research*, 6(1):1–9, 2021.

[365] He Chen, Yan Song, and Xiaoli Li. Use of deep learning to detect personalized spatial-frequency abnormalities in eegs of children with adhd. *Journal of neural engineering*, 16(6):066046, 2019.

[366] Mustafa Tosun. Effects of spectral features of eeg signals recorded with different channels and recording statuses on adhd classification with deep learning. *Physical and Engineering Sciences in Medicine*, 44(3):693–702, 2021.

[367] Raffaella Franciotti, Davide Nardini, Mirella Russo, Marco Onofrj, Stefano L Sensi, Alzheimer's Disease Neuroimaging Initiative, et al. Comparison of machine learning-based approaches to predict the conversion to alzheimer's disease from mild cognitive impairment. *Neuroscience*, 514:143–152, 2023.

[368] M Senthil Kumar, H Azath, AK Velmurugan, K Padmanaban, and Murugan Subbiah. Prediction of alzheimer's disease using hybrid machine learning technique. In *AIP Conference Proceedings*, volume 2523. AIP Publishing, 2023.

[369] Afiya Parveen Begum and Prabha Selvaraj. Multiclass diagnosis of alzheimer's disease analysis using machine learning and deep learning techniques. *International Journal of Image and Graphics*, page 2450031, 2023.

[370] Xiaoyi Raymond Gao, Marion Chiariglione, Ke Qin, Karen Nuytemans, Douglas W Scharre, Yi-Ju Li, and Eden R Martin. Explainable machine learning aggregates polygenic risk scores and electronic health records for alzheimer's disease prediction. *Scientific Reports*, 13(1):450, 2023.

[371] Yasunobu Nohara, Koutarou Matsumoto, Hidehisa Soejima, and Naoki Nakashima. Explanation of machine learning models using shapley additive explanation and application for real data in hospital. *Computer Methods and Programs in Biomedicine*, 214:106584, 2022.

[372] Salim Lahmiri. Integrating convolutional neural networks, knn, and bayesian optimization for efficient diagnosis of alzheimer's disease in magnetic resonance images. *Biomedical Signal Processing and Control*, 80:104375, 2023.

[373] Syrine Neffati, Khaoula Ben Abdellafou, Ines Jaffel, Okba Taouali, and Kais Bouzrara. An improved machine learning technique based on downsized kpca for alzheimer's disease classification. *International Journal of Imaging Systems and Technology*, 29(2):121–131, 2019.

[374] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[375] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700. JMLR Workshop and Conference Proceedings, 2010.

[376] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[377] Richard Kinh Gian Do Rikiya Yamashita, Mizuho Nishio and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9:611–629, 2018.

[378] EL-Geneedy Marwa, Hossam El-Din Moustafa, Fahmi Khalifa, Hatem Khater, and Eman AbdElhalim. An mri-based deep learning approach for accurate detection of alzheimer's disease. *Alexandria Engineering Journal*, 63:211–221, 2023.

[379] Jyoti Islam and Yanqing Zhang. Brain mri analysis for alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks. *Brain informatics*, 5:1–14, 2018.

[380] Saman Sarraf, Arman Sarraf, Danielle D DeSouza, John AE Anderson, Milton Kabia, and Alzheimer's Disease Neuroimaging Initiative. Ovitad: Optimized vision transformer to predict various stages of alzheimer's disease using resting-state fmri and structural mri data. *Brain Sciences*, 13(2):260, 2023.

[381] Rahul Sharma, Tripti Goel, M Tanveer, CT Lin, and R Murugan. Deep learning based diagnosis and prognosis of alzheimer's disease: A comprehensive review. *IEEE Transactions on Cognitive and Developmental Systems*, 2023.

[382] Petra E Vértes, Aaron F Alexander-Bloch, Nitin Gogtay, Jay N Giedd, Judith L Rapoport, and Edward T Bullmore. Simple models of human brain functional networks. *Proceedings of the National Academy of Sciences*, 109(15):5868–5873, 2012.

[383] Richard F Betzel, Andrea Avena-Koenigsberger, Joaquín Goñi, Ye He, Marcel A De Reus, Alessandra Griffa, Petra E Vértes, Bratislav Mišic, Jean-Philippe Thiran, Patric Hagmann, et al. Generative models of the human connectome. *Neuroimage*, 124:1054–1064, 2016.

[384] Danielle S Bassett, Perry Zurn, and Joshua I Gold. On the nature and use of models in network neuroscience. *Nature Reviews Neuroscience*, 19(9):566–578, 2018.

[385] Joana Cabral, Morten L Kringelbach, and Gustavo Deco. Exploring the network dynamics underlying brain activity during rest. *Progress in neurobiology*, 114:102–131, 2014.

[386] Azriel Rosenfeld and Pilar De La Torre. Histogram concavity analysis as an aid in threshold selection. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(2):231–235, 1983.

[387] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2010.

[388] Pablo Gleiser and Leon Danon. Community structure in jazz. *Advances in Complex Systems*, 6:565, 2003.

[389] David Lusseau. The emergent properties of a dolphin social network. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(suppl_2):S186–S188, 2003.

[390] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68(6), 12 2003.

[391] Scott P. Kolodziej, Mohsen Aznaveh, Matthew Bullock, Jarrett David, Timothy A. Davis, Matthew Henderson, Yifan Hu, and Read Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019.

[392] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. On the permanence of vertices in network communities. pages 1396–1405, 2014.

[393] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[394] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[395] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[396] Aditya Tandon, Aiiad Albeshri, Vijey Thayananthan, Wadee Alhalabi, and Santo Fortunato. Fast consensus clustering in complex networks. *Physical Review E*, 99(4), 4 2019.

[397] Rohit Parimi and Doina Caragea. Community detection on large graph datasets for recommender systems. In *2014 IEEE international conference on data mining workshop*, pages 589–596. IEEE, 2014.

[398] Shumeet Baluja, Rohan Seth, Dharshi Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, pages 895–904, 2008.

[399] Network dataset. Bookcrossing (implicit). *KONECT*, 2022.

[400] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018.

[401] Network dataset. Dblp computer science bibliography. *KONECT*, 2022.

[402] Network dataset. Movielens 20m. *Kaggle*, 2022.

[403] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[404] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, 2008.

[405] Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 288–304, Cham, 2017. Springer International Publishing.

[406] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[407] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[408] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021.

[409] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297, 1967.

[410] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[411] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

[412] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes, 2015.

[413] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

[414] Han Yang, Xiao Yan, Xinyan Dai, Yongqiang Chen, and James Cheng. Self-enhanced gnn: Improving graph neural networks using model outputs. In *International Joint Conference on Neural Networks*, 2021.

[415] Xingxing Liang, Yang Ma, Guangquan Cheng, Changjun Fan, Yuling Yang, and Zhong Liu. Meta-path-based heterogeneous graph neural networks in academic network. *International Journal of Machine Learning and Cybernetics*, 13(6):1553–1569, 2022.

[416] Ilina Singh. Beyond polemics: science and ethics of adhd. *Nature Reviews Neuroscience*, 9(12):957–964, 2008.

[417] Guilherme Polanczyk and Peter Jensen. Epidemiologic considerations in attention deficit hyperactivity disorder: a review and update. *Child and adolescent psychiatric clinics of North America*, 17(2):245–260, 2008.

[418] Steven A Safren, Michael W Otto, Susan Sprich, Carol L Winett, Timothy E Wilens, and Joseph Biederman. Cognitive-behavioral therapy for adhd in medication-treated adults with continued symptoms. *Behaviour research and therapy*, 43(7):831–842, 2005.

[419] Agatha Lenartowicz and Sandra K Loo. Use of eeg to diagnose adhd. *Current psychiatry reports*, 16(11):1–11, 2014.

[420] John Dellabadia Jr, William L Bell, John W Keyes Jr, Vincent P Mathews, and Stephen S Glazier. Assessment and cost comparison of sleep-deprived eeg, mri and pet in the prediction of surgical treatment for epilepsy. *Seizure*, 11(5):303–309, 2002.

[421] Samuele Cortese, Clare Kelly, Camille Chabernaud, Erika Proal, Adriana Di Martino, Michael P Milham, and F Xavier Castellanos. Toward systems neuroscience of adhd: a meta-analysis of 55 fmri studies. *American Journal of Psychiatry*, 169(10):1038–1055, 2012.

[422] Nikos K Logothetis. The neural basis of the blood–oxygen–level–dependent functional magnetic resonance imaging signal. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 357(1424):1003–1037, 2002.

[423] Elseline Hoekzema, Susana Carmona, J Antoni Ramos-Quiroga, Vanesa Richarte Fernandez, Rosa Bosch, Juan Carlos Soliva, Mariana Rovira, Antonio Bulbena, Adolf Tobena, Miguel Casas, et al. An independent components and functional connectivity analysis of resting state fmri data points to neural network dysregulation in adult adhd. *Human brain mapping*, 35(4):1261–1272, 2014.

[424] Dai Dai, Jieqiong Wang, Jing Hua, and Huiguang He. Classification of adhd children through multimodal magnetic resonance imaging. *Frontiers in systems neuroscience*, 6:63, 2012.

[425] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. Generalizations of the clustering coefficient to weighted complex networks. *Physical Review E*, 75(2):027105, 2007.

[426] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

[427] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9:1735–1780, November 1997.

[428] The adhd-200 sample. http://fcon_1000.projects.nitrc.org/indi/adhd200/. Accessed: October 22, 2022.

[429] Neuroimaging tools and resources collaboratory. https://www.nitrc.org/. Accessed: October 22, 2022.

[430] 1000 functional connectomes project. https://www.nitrc.org/projects/fcon_1000/. Accessed: October 22, 2022.

[431] fmriprep: A robust preprocessing pipeline for fmri data. https://fmriprep.org/en/stable/. Accessed: October 22, 2022.

[432] Krzysztof J Gorgolewski, Tibor Auer, Vince D Calhoun, R Cameron Craddock, Samir Das, Eugene P Duff, Guillaume Flandin, Satrajit S Ghosh, Tristan Glatard, Yaroslav O Halchenko, et al. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific data*, 3(1):1–9, 2016.

[433] Mark Jenkinson, Christian F Beckmann, Timothy EJ Behrens, Mark W Woolrich, and Stephen M Smith. Fsl. *Neuroimage*, 62(2):782–790, 2012.

[434] Robert W Cox. Afni: software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical research*, 29(3):162–173, 1996.

[435] Bruce Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012.

[436] Brian B Avants, Nick Tustison, Gang Song, et al. Advanced normalization tools (ants). *Insight j*, 2(365):1–35, 2009.

[437] Atif Riaz, Muhammad Asad, SM Masudur Rahman Al Arif, Eduardo Alonso, Danai Dima, Philip Corr, and Greg Slabaugh. Deep fmri: An end-to-end deep network for classification of fmri data. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 1419–1422. IEEE, 2018.

[438] Alzheimer's Association et al. 2018 alzheimer's disease facts and figures. *Alzheimer's & Dementia*, 14(3):367–429, 2018.

[439] Kun Wang, Meng Liang, Liang Wang, Lixia Tian, Xinqing Zhang, Kuncheng Li, and Tianzi Jiang. Altered functional connectivity in early alzheimer's disease: A resting-state fmri study. *Human brain mapping*, 28(10):967–978, 2007.

[440] Yong He, Zhang Chen, and Alan Evans. Structural insights into aberrant topological patterns of large-scale cortical networks in alzheimer's disease. *Journal of Neuroscience*, 28(18):4756–4766, 2008.

[441] Betty M Tijms, Alle Meije Wink, Willem de Haan, Wiesje M van der Flier, Cornelis J Stam, Philip Scheltens, and Frederik Barkhof. Alzheimer's disease: connecting findings from graph theoretical studies of brain networks. *Neurobiology of aging*, 34(8):2023–2036, 2013.

[442] Xavier Delbeuck, Fabienne Collette, and Martial Van der Linden. Is alzheimer's disease a disconnection syndrome?: Evidence from a crossmodal audio-visual illusory experiment. *Neuropsychologia*, 45(14):3315–3323, 2007.

[443] Julio Acosta-Cabronero, Stephanie Alley, Guy B Williams, George Pengas, and Peter J Nestor. Diffusion tensor metrics as biomarkers in alzheimer's disease. *PloS one*, 7(11):e49072, 2012.

[444] Kaustubh Supekar, Vinod Menon, Daniel Rubin, Mark Musen, and Michael D Greicius. Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS computational biology*, 4(6):e1000100, 2008.

[445] Nhat Trung Doan, Andreas Engvig, Karin Persson, Dag Alnæs, Tobias Kaufmann, Jaroslav Rokicki, Aldo Córdova-Palomera, Torgeir Moberget, Anne Brækhus, Maria Lage Barca, et al. Dissociable diffusion mri patterns of white matter microstructure and connectivity in alzheimer's disease spectrum. *Scientific reports*, 7(1):1–12, 2017.

[446] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.

[447] Nathalie Tzourio-Mazoyer, Brigitte Landeau, Dimitri Papathanassiou, Fabrice Crivello, Olivier Etard, Nicolas Delcroix, Bernard Mazoyer, and Marc Joliot. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *Neuroimage*, 15(1):273–289, 2002.

[448] Jorge L Mendoza. Fisher transformations for correlations corrected for selection and missing data. *Psychometrika*, 58:601–615, 1993.

[449] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. pages 7–15, 1987.

[450] Marcus Kaiser and Claus C Hilgetag. Spatial growth of real-world networks. *Physical Review E*, 69(3):036103, 2004.

[451] Claus C Kaiser, Marcus Hilgetag. Modelling the development of cortical systems networks. *Neurocomputing*, 58:297–302, 2004.

[452] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *LREC*, volume 6, pages 1222–1225, 2006.

[453] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[454] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[455] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[456] Michael A DeTure and Dennis W Dickson. The neuropathological diagnosis of alzheimer's disease. *Molecular neurodegeneration*, 14(1):1–18, 2019.

[457] Eriola Hoxha, Pellegrino Lippiello, Fabio Zurlo, Ilaria Balbo, Rita Santamaria, Filippo Tempia, and Maria Concetta Miniaci. The emerging role of altered cerebellar synaptic processing in alzheimer's disease. *Frontiers in aging neuroscience*, 10:396, 2018.

[458] Monica Bianchini, Anas Belahcen, and Franco Scarselli. A comparative study of inductive and transductive learning with feedforward neural networks. In *Conference of the Italian Association for Artificial Intelligence*, pages 283–293. Springer, 2016.

[459] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19, 2006.

[460] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and TH Hubert Chan. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *International Conference on Machine Learning*, pages 4026–4034. PMLR, 2017.

[461] Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2570–2581. SIAM, 2019.

[462] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019.

[463] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[464] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. *arXiv preprint arXiv:1911.02613*, 2019.

[465] Timoteo Carletti, Federico Battiston, Giulia Cencetti, and Duccio Fanelli. Random walks on hypergraphs. *Physical review E*, 101(2):022308, 2020.

[466] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[467] Alankrita Aggarwal, Mamta Mittal, and Gopi Battineni. Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1):100004, 2021.

[468] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[469] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[470] Ronald C Petersen, Glenn E Smith, Stephen C Waring, Robert J Ivnik, Eric G Tangalos, and Emre Kokmen. Mild cognitive impairment: clinical characterization and outcome. *Archives of neurology*, 56(3):303–308, 1999.

[471] P Fischer, S Jungwirth, S Zehetmayer, S Weissgram, S Hoenigschnabl, E Gelpi, W Krampla, and KH Tragl. Conversion from subtypes of mild cognitive impairment to alzheimer dementia. *Neurology*, 68(4):288–291, 2007.