JADAVPUR UNIVERSITY

MASTER DEGREE THESIS

To develop an efficient technique to solve community detection problem in Multi-view information network

A thesis submitted in fulfillment of the requirements for the degree of Master of Engineering

in

Computer Science and Engineering

by

Meghna Chowdhury
Exam Roll No.: M4CSE22039
Class Roll No.: 002010502039
Registration No.: 154163 of 2020-2021

Under the Guidance of

Prof. Nirmalya Chowdhury
Department of Computer Science and Engineering
Jadavpur University

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Kolkata - 700032

July, 2022

Declaration of Authorship

I, Meghna Chowdhury, declare that this thesis titled, “To develop an efficient technique to solve community detection problem in Multi-view information network” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a masters degree in computer science and engineering at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely of my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

To Whom It May Concern

This is to certify that the thesis entitled "To develop an efficient technique to solve community detection problem in Multi-view information network" is a bona-fide record of work carried out by Meghna Chowdhury, Examination Roll No.: M4CSE22039, University Registration No.: 154163 of 2020-2021 in partial fulfillment of the requirements for the award of the degree of Master of Engineering in Computer Science and Engineering from the Department of Computer Science and Engineering, Jadavpur University for the academic session 2020-2022. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Prof. Nirmalya Chowdhury
Department of Computer Science and Engineering
Jadavpur University
Kolkata - 700032

Prof. Anupam Sinha
Head of the Department
Department of Computer Science and Engineering
Jadavpur University
Kolkata - 700032

Prof. Chandan Mazumdar
Dean,Faculty of Engineering and Technology
Jadavpur University
Kolkata - 700032

Certificate of Approval

(Only in case the thesis is approved)

The thesis at instance is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory for its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis only for the purpose for which it is submitted.

____

(Sign of Examiner)
Date:

____

(Sign of Examiner)
Date:

JADAVPUR UNIVERSITY

Abstract

Faculty of Engineering and Technology
Department of Computer Science and Engineering

Master of Engineering

To develop an efficient technique for learning information in Multi-view network data

Approach by Meghna Chowdhury

With the advent of information stored in the form of network structured data, large-scale graph based machine learning techniques have gained sudden relevance in the field of artificial intelligence. These techniques have been used to solve problems from very diverse domains, such as humanitarian response, poverty estimation, urban planning, epidemic containment, etc. Yet the vast majority of computational tools and algorithms used in these applications do not account for the multi-view nature of social networks: people are related in myriad ways, but most graph learning models treat relations as binary. In this work, we develop a graph-based convolutional network for learning on multi-view networks by introducing preservation and collaboration parameters and effectively optimizing them using Non Dominated Sorting Genetic Algorithm. We show that this method outperforms state-of-the-art learning algorithms on multi-view networks.

Acknowledgements

I would like to extend my heartfelt gratitude to all the people who had helped to bring this thesis work to completion.

Firstly, I would express my deep and sincere gratitude to my supervisor Prof. Nirmalya Chowdhury, Department of Computer Science and Engineering, Jadavpur University, without whose continuous support and encouragement this work would not have been possible. His assistance, valuable suggestions and personal guidance throughout the duration of the project has played a pivotal role, without which I would never have been able to reach this far.

I would also like to thank Prof. Anupam Sinha, Head of the Department of Computer Science and Engineering, Jadavpur University and Prof. Chandan Mazumdar, Dean, Faculty of Engineering and Technology, Jadavpur University, for providing me with all the facilities and for their support to the activities of this research.

I am thankful to my parents who have always been my constant source of support and inspiration.

Last but not the least, I would like to thank all my friends, classmates and all respected teachers for their valuable suggestions and helpful discussions.

Regards,
Meghna Chowdhury
Exam Roll No.: M4CSE22039
University Registration No.: 154163 of 2020-2021
Department of Computer Science and Engineering
Jadavpur University

I dedicate this to my parents for their continuous support throughout my journey!

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CD	Community Detection
ASN	Node-Attributed Social Network
GNN	Graph Neural Network
GCN	Graph Convolutional Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MLP	Multilayer Perceptron
NLP	Natural Language Processing
W2V	Word2Vec
STGNN	Spatial Temporal Graph Neural Network
MOEA	Multi-Objective Evolutionary Algorithm
GA	Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
MOO	Multi Objective Optimization
GAT	Graph attention network
MGAT	Multi-view Graph Attention Network
MOGA	Multi Objective Genetic Algorithm
MVNR	Multi-View Network Representation Algorithm
NEU	Network Embedding Update

Chapter 1

Introduction to Community Detection

1.1 Community Detection

A network is considered to have community structure in the study of complex networks if its nodes can be quickly categorised into sets of nodes, with each set being densely connected to the others. In terms of graphs, a community is a subset of nodes that are closely related to one another and distantly connected to the nodes in the other communities. An illustration will help to clarify this. If we consider social media sites where we attempt to communicate with others, like Facebook, Instagram, or Twitter. After some time has passed, we eventually connect with others from various social circles. These social circles could consist of a family, friends from school, coworkers, etc. Communities are all that these social circles are. Communities are a characteristic of many networks, and a given network may contain a number of communities, each of which has a tightly connected set of nodes. Nodes from different communities can cross over. If we consider our Facebook or Instagram accounts and the people we communicate with on a regular basis. We might communicate frequently with our friends, coworkers, family, and other significant individuals in our lives. Within our social network, they create a highly dense community. Finding communities inside different networks may be crucial when analysing them. Social media algorithms can find people who share interests and keep them connected by using community detection techniques. Machine learning may utilise community detection to find groups with similar characteristics and extract groupings for a variety of purposes. This method, for instance, can be used to identify manipulative groups in social networks or stock markets. Finding groupings of nodes that are, in some way, more similar to one another than to other nodes is the goal of the fundamental problem of community detection in network analysis. One may contend that community detection and clustering are equivalent. Using the machine learning technique of clustering, related data points are grouped together based on shared characteristics. Clustering is a more general area of unsupervised machine learning that deals with a variety of attribute types, despite the fact that it can be applied to networks. However, community detection was created specifically for network analysis, which is based on a single attribute type known as edges. Additionally, single peripheral nodes sometimes become isolated from the communities they should be a part of as a result of clustering techniques. However, depending on the problem and the area, different advantages and disadvantages of using clustering and community detection algorithms may arise.

1.2 Community Detection in Multi-view Networks

A graph network can be visualised as a collection of nodes (vertices) and the edges connecting them. Since all of these networks exhibit some community characteristics, complex networks have been used to represent a variety of real-world network systems, including the World Wide Web, networks for scientific collaboration, social and biological networks, and many others. Understanding the behaviour and structure of complex networks as well as the connections between generic entities requires revealing these structures, also known as community detection. Community detection aims to group all nodes in a complex network into various clusters so that connections between nodes within a cluster are dense and connections between nodes in other clusters are sparse. It has been established that this problem is NP-hard. The complex network detection problem is significant, and since the 1930s, research on this topic has gained popularity. Numerous interdisciplinary researchers have been working on this issue, and numerous methods have been put out for various kinds of complex networks. Since 2005, surveys of community detection in graphs and networks have been published on a regular basis. Prior studies on community detection typically focused on the single-view environment. Views are separate datasets or data sources. A well-known example is the classification of online pages, where one view consists of the web page's content and the other of the hyperlinks going to it. User interactions are complicated on social networks like YouTube and Flickr. Similar to this, Facebook users connect with one another by liking, commenting on, and sharing information. Particularly, two views can be formed on the same page by the activities associated with posts and comments. It is possible to create features from each view and build a graph to detect community structures on a specific website. In addition to user interactions becoming more complicated, social media has generated an exponentially growing amount of data. More than 2 billion people were using Facebook as of the end of 2017, and up to 40 million small companies had public pages. It is discovered that there is a large correlating increase in difficulty of finding community structures within Facebook pages with an increase in users and accompanying interactions. And this is just the example of only one such network, there are multiple such networks and community detection in such multiview networks is not quite an easy task like community detection in lesser complex networks which have their network representations only in a single view format. We can say that, there still lies the problem of community detection in multi-view networks and it is of very much importance in today’s date.

1.3 Motivation

In a multi-view network, where each view represents a different kind of interaction between nodes, multi-view graph embedding aims to learn low-dimensional representations of nodes that capture diverse relationships. Numerous graph embedding techniques currently in use focus on single-view networks, which can only describe one straightforward sort of proximity interactions between objects. However, the majority of complex systems in real life have a variety of interactions between the entities. And when it comes to community detection in networks, it is also the same, the existing community detection techniques mostly focus on networks with single view.

Multi view networks are existing around us in almost every field. May it be Twitter networks, Facebook networks, LinkedIn networks, or any other real time networks, all of them show some kind of multi view property and community detection in these kind of networks is an area which still needs to be explored and it is necessary to come up with an effective solution which will solve this problem of community detection in Multi view networks.

1.4 Aim and Objective

The aim of this thesis is to develop a technique which will efficiently learn the pattern in the Multi-view networks. This requires the processing and development of datasets which consist of multi-view graph networks and represent the network in the different views that the graph consists of. In order to achieve this goal, a number of key points need to be achieved :

- Using subspace analysis to efficiently merge multiple views of the graph.
- Find the most informative subcomponents of the graph
- Using multi objective optimization to adjust the preservation and collaboration of the information in different views.
- Using a neural network model to effectively learn the data and make predictions and class labelling.

1.5 Contribution

Learning of multi-view graph network data has always been a challenge for conflicting view types. In the project we have tried to propose an efficient learning mechanism for multi-view graph data and hence making the model capable for node classification and prediction. We have used the methods from subspace analysis to efficiently merge multiple views of the same graph. Then we have used a manifold ranking procedure to identify the most informative sub-components of the graph and to prune the graph upon which learning is performed. This is achieved by optimising the preservation and collaboration factor using NSGA III. Finally, we apply a convolutional neural network, adapted to graph-structured data, to allow for semi-supervised node classification.

1.6 Thesis Organisation

The rest of the thesis is organised as follows: Chapter 2 describes the concept of deep learning, focusing mainly on the theory of Graph Neural Network, the embedding techniques of the Graph Networks and the various fields of applications of Graph Neural Network and also the concept of Graph Convolutional Networks and the main intuition behind the working of Graph convolutional Networks. Chapter 3 Contains the brief description and the mathematical representation of Multi-view information networks Chapter 4 briefly discusses the theory of optimisation techniques focusing mainly on Multi-Objective Optimisation and Genetic Algorithms for optimisation. Chapter 5 consists of a literature survey related to the Multi-view networks and community detection using Multi Objective Evolutionary Algorithms . Our proposed method is discussed in Chapter 6 and the process of data collection and preparation is discussed in Chapter 7. The experimental results are analysed in Chapter 8. Finally, we conclude and put some light on future work in Chapter 9.

Chapter 2

An Brief Introduction to Deep Learning

Deep learning is an area of machine learning that deals with artificial neural networks, which are algorithms inspired by the structure and function of the brain. It is a type of machine learning technique that employs numerous layers to extract higher-level features from unprocessed data. It is part of a larger family of machine learning algorithms based on representation learning and artificial neural networks. The learning can be supervised, semi-supervised, or unsupervised in this situation. In deep learning, the word "deep" refers to the employment of numerous layers in the network. Each level of deep learning learns to turn the data it receives into a little more abstract and composite representation. While typical programmes develop analyses using data in a linear manner, deep learning systems' hierarchical function allows machines to process data in a nonlinear manner. Another frequently stated virtue of deep learning models, in addition to scalability, is their capacity to do autonomous feature extraction from raw data, also known as feature learning.

2.1 Graph Neural Networks

Pattern recognition and data mining research has gained a boost, thanks to the recent success of neural networks. End-to-end deep learning paradigms like CNN, RNN, and autoencoders have given fresh life to machine learning applications including object detection, machine translation, and speech recognition. Neural networks have been adapted to leverage the structure and properties of graphs. Deep Learning is effective at detecting hidden patterns in Euclidean data (images, text, videos). Graph Neural Networks (GNN) are useful in applications where data is generated from non-Euclidean domains, which are represented as graphs with complicated interactions and interdependencies between items. Graph Neural Networks (GNNs) are neural models that capture the dependence of graphs via message passing between the nodes of graphs. GNNs are neural networks that can be applied directly to graphs, making node-level, edge-level, and graph-level prediction jobs simple. Where Convolutional Neural Networks (CNNs) fail, GNNs can come to the rescue. GNN is gaining traction in a variety of fields, including social networking, knowledge graphs, recommender systems, and even life science. The ability of GNN to model the dependencies between nodes in a network allows for a breakthrough in graph analysis research. A GNN can also be termed as a graph symmetry-preserving transformation that optimises all graph properties (nodes, edges, and global context) (permutation invariances).

2.1.1 Graph

Graphs are a type of data structure that represents a collection of items (nodes) and their connections (edges). Because of the great expressive power of graphs, i.e., graphs can be used as denotation of a large number of systems across various areas including social science (social networks (Wu et al., 2020), natural science (physical systems (Sanchez et al., 2018; Battaglia et al., 2016) and protein-protein interaction networks (Fout et al., 2017), knowledge graphs (Hamaguchi et al (Khalil et al., 2017). Graph analysis focuses on node categorization, connection prediction, and clustering as a unique non-Euclidean data structure for machine learning. Let's start with a definition of Graph before moving on to GNN. A graph is a data structure in computer science that is made up of two parts: vertices and edges. The set of vertices V and edges E that make up a graph G can be used to describe it. Depending on whether or not there are directional relationships between vertices, edges can be directed or undirected. The vertices are also known as nodes and a graph can be represented as

$$G = (V, E)$$

Social media networks or molecules can also be represented by graphs if the nodes are considered as users and the edges as connections.

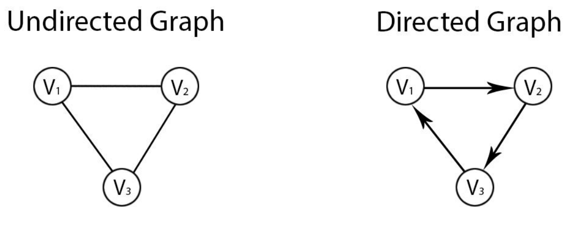


Fig 1 : Representation of an undirected and directed graph

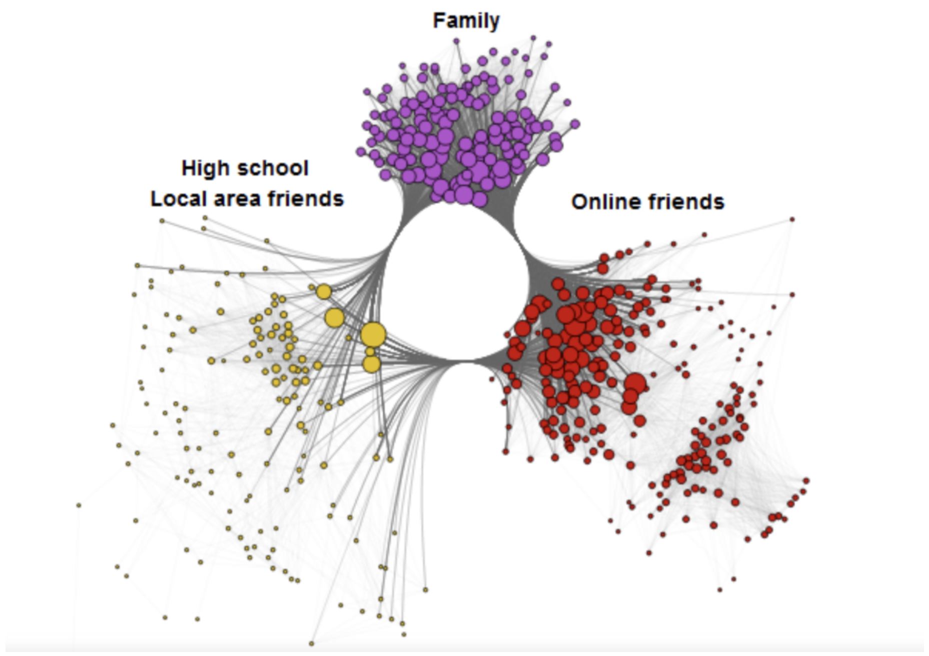


Fig 2 : Representation of a social media graph from [2]

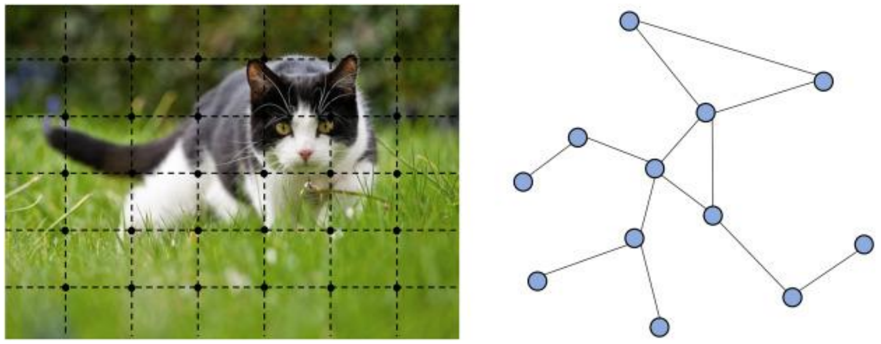


Fig. 3. Left: image in Euclidean space. Right: graph in non-Euclidean space from [1]

An adjacency matrix, mostly represented by A, is frequently used to represent a graph. A has a dimension of (n⨉n) if a graph contains n nodes. Some nodes have a collection of characteristics (for example, a user profile). The node feature matrix *X* has a dimension of (n⨉f) if the node has *f* numbers of features.

Real-world objects are frequently characterised in terms of their connections to other things, therefore graphs exist all around us. A graph is a natural representation of a collection of things and their connections. For more than a decade, researchers have been developing neural networks that operate on graph data (known as graph neural networks, or GNNs). Recent advancements have expanded their expressive power and potential. Antibacterial discovery, physics simulations, fake news identification, traffic prediction, and recommendation systems are all examples of practical uses.

2.1.1.1 Images as Graph

Images are commonly thought of as rectangular grids with image channels, and they are represented as arrays (e.g., 244x244x3 floats) . Images can also be thought of as regular-structured graphs, with each pixel representing a node and connected to adjacent pixels through an edge. Each non-border pixel has exactly eight neighbours, with each node storing a 3-dimensional vector encoding the pixel's RGB value.

The adjacency matrix is a tool for visualising the connectivity of a graph. If the nodes are ordered in such a way such that for example, if there is an image of 25 pixels then an adjacency matrix of the dimension (25x25) will be filled with some entry if there exists an edge between two nodes.

2.1.1.2 Text as Graph

Text can be digitised by assigning indices to each character, word, or token and representing text as a series of these indices. This results in a simple directed graph, in which each character or index is a node connected to the node that follows it by an edge. Though, in most cases, this is not how text and images are encoded in practice. This kind of graph representations are unnecessary because all images and text will have extremely consistent structures. Because all nodes (pixels) are connected in a grid, images have a banded pattern in their adjacency matrix. Because each word only connects to the preceding and subsequent words, the adjacency matrix for text is merely a diagonal line.

2.1.1.3 Molecules as Graph

Molecules are the fundamental building blocks of matter, consisting of atoms and electrons arranged in three dimensions. All particles interact, but we call a pair of atoms who share a covalent link when they are separated by a stable distance. Varying atoms and bonds have different distances between them (e.g. single-bonds, double-bonds). The representation of this 3D entity as a graph, with nodes representing atoms and edges representing covalent bonds, is a very convenient and popular abstraction.

2.1.1.4 Social Networks as Graph

Social networks are instruments for studying trends in people's, institutions', and organisations' collective behaviour. Individuals can be represented as nodes, and their relationships as edges, in a graph that represents groups of people.

2.1.1.5 Citation Networks as Graph

When scientists publish papers, they frequently cite the work of other scientists. These citation networks can be shown as a graph, with each paper as a node and each directed edge as a citation between two papers. We can also include information about each document in each node, such as the abstract's word embedding.

2.1.1.6 Few other miscellaneous Graph representations

We sometimes want to tag items in visual scenes in computer vision. Then, by treating these things as nodes and their relationships as edges, we can construct graphs. Machine learning models, programming code, and maths equations can all be thought of as graphs, with nodes representing variables and edges representing actions with these variables as input and output. In some of these cases, the phrase "dataflow graph" may be used.

The structure of real-world networks varies widely depending on the type of data – some graphs have many nodes but few connections between them, while others have few nodes but many connections. In terms of the amount of nodes, edges, and connectivity of nodes, graph datasets can vary widely (both within and between datasets).

2.1.2 Functions of GNN

Some examples of graphs have been discussed above, but there are certain tasks or functions which are performed on this data. In general, there are types of prediction tasks on graphs: graph-level, node-level, and edge-level. It can also be termed in this way that the challenges that can be solved by GNN are generally divided into three categories which are discussed below.

2.1.2.1 Graph Classification

The goal of a graph-level task is to anticipate an entire graph's property. For example, we could wish to forecast what a molecule will smell like or if it will attach to a disease-related receptor using a graph representation. As a result, graph classification can be defined as the process of categorising the complete graph into various groups. It's similar to an image classification task, however the goal is to find graphs. In chemistry, for example, the classification of a chemical structure into one of several categories is an example of graph classification.

2.1.2.1 Node Classification

Predicting the identity or role of each node in a graph is the focus of node-level tasks. Basically, this task includes estimating the node embedding for each node in a network. In such cases, only a piece of the graph is labelled, resulting in a semi-supervised graph. Examples are YouTube videos, Facebook friend suggestions, and other applications.

2.1.2.1 Link Prediction

The main goal of edge-level tasks is to figure out how two things in a graph are related and predict whether or not they are connected. If a recommender system in which a model is given a set of user reviews for a variety of products. The goal is to predict user preferences and optimise the recommender system so that it offers things that are relevant to their needs. Image scene understanding is an example of edge-level inference. Deep learning models can be used to anticipate the relationship between objects in a picture in addition to identifying them. We may think of this as edge-level classification: given a set of nodes that represent the objects in the image, we want to predict which of these nodes share an edge and how much that edge is worth. We might consider the graph completely connected and trim edges depending on their anticipated value to arrive at a sparse graph if we want to identify relationships between things.

2.1.3 Embedding techniques

Till now, the above discussion comes to a point we can not deny that the Graph Neural Networks are currently a hot topic. GNNs are all about latent representation of the graph in vector space, thus this interest is understandable. It's nothing new to represent an entity as a vector. Many instances exist in NLP, which convert a word into a vector. What makes such representations powerful is that they incorporate a notion of similarity among them, i.e two words that are similar to each other tend to be closer in the vector space (dot product is large), and they can be used in a variety of downstream problems such as classification, clustering, and so on. This is what makes GNN intriguing, because while there are numerous ways to embed a phrase or image as a vector, none of them are as elegant as GNN. The primary purpose of graph embedding methods is to compress each node's attributes into a smaller vector, allowing node similarity in the original complicated irregular spaces to be easily quantified in the embedded vector spaces using standard metrics. Few of the methods are briefly discussed below.

2.1.3.1 Word2Vec

Word embeddings are an important aspect of many NLP difficulties. They show how a machine interprets human language. We can think of them as a vectorized text representation. Word2Vec, a popular approach for producing word embeddings, has a wide range of uses, including text similarity, recommendation systems, sentiment analysis, and more.Word2Vec was introduced by Google in 2013, allowed words to be embedded in n-dimensional space, with similar words clustered together locally. This indicates that cosine distances between words that are frequently used together or in similar situations are less. Word2Vec does this by comparing the target words to their context using the Skip-Gram technique. At its most basic level, Skip-Gram employs a sliding window technique, in which it attempts to predict the surrounding words based on the target word in the centre. This means that we are effectively trying to estimate the neighbours around the target node inside our network in our use-case of trying to encode comparable nodes within the graph to be close to one another in n-dimensional space.

- **Word Embedding:** Before diving into word2vec, it's crucial to first grasp the concept of word embeddings. This is vital to understand since word2vec's overall result and output will be embeddings connected with each unique word that passes through the process. Individual words are turned into a numerical representation of the word (a vector) using word embeddings. Each word is mapped to a single vector, which is then trained in a manner similar to that of a neural network. The vectors attempt to capture various aspects of that word in relation to the rest of the text. The semantic relationship of the word, definitions, context, and so on are examples of these properties. Many things can be done with these numerical representations, such as identifying similarity or dissimilarity between words. These are obviously important as inputs to many areas of machine learning. Text cannot be processed by a machine in its raw form, therefore turning it to an embedded form allows users to feed the embedding to traditional machine learning models. The most straightforward embedding would be a one-hot encoding of text data, with each vector mapped to a category. But these kinds of simple embeddings, on the other hand, have a number of drawbacks, including the fact that they don't capture word properties and can be rather large depending on the corpus size.

Word2Vec's efficacy stems from its ability to combine vectors of similar words together. Word2Vec can create solid guesses about a word's meaning based on its appearances in the text if given a large enough dataset. Word connections with other words in the corpus are derived from these estimates. Words like "King" and "Queen," for example, are quite similar to one another. We can discover a close approximation of word similarities by performing algebraic operations on word embeddings. For example, the 2 dimensional embedding vector of "king" - the 2 dimensional embedding vector of "man" + the 2 dimensional embedding vector of "woman" yielded a vector which is very close to the embedding vector of "queen".

```
King    -    Man    +    Woman    =    Queen
```

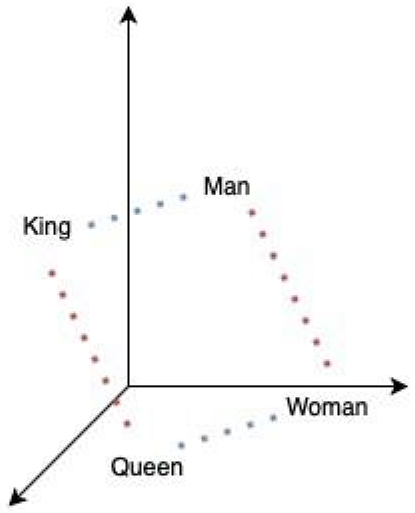


Fig 4 : 3-Dim Example of Word Embedding Behaviour [21]

Word2vec's success can be attributed to two key architectures. The architectures of skip-gram and CBOW.

- **CBOW (Continuous Bag of Words) :** A feed forward neural network is quite similar to this architecture. The goal of this model architecture is to predict a target word from a set of context words. The concept behind this model is straightforward: If a phrase is given such as "*Have a great day*," a target word will be chosen, here "a" has been chosen as target word and ["have," "great," "day"] as context words. The distributed representations of the context words will be used by this model to predict the target word.

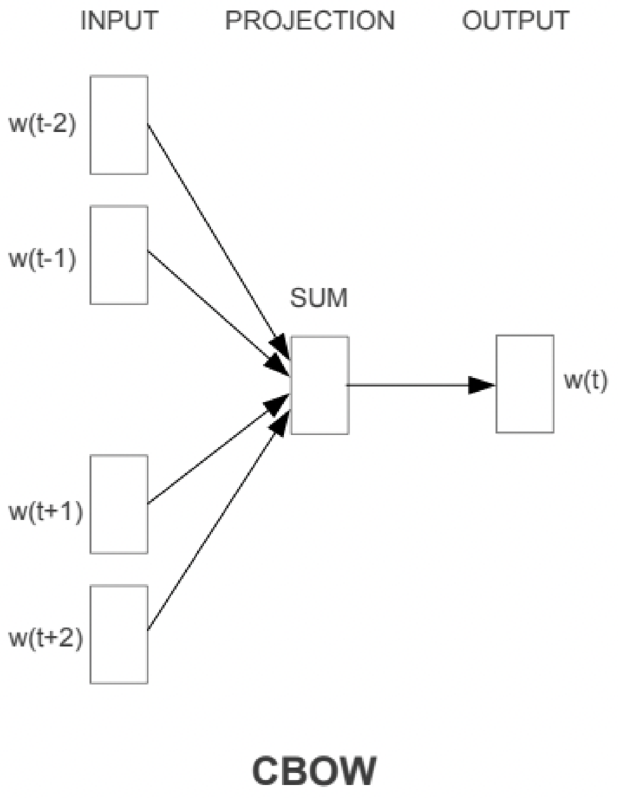


Fig 5 : CBOW Architecture. Image taken from [16]

- **Continuous Skip gram model :** We need unsupervised learning algorithms that can learn the context of each word on its own because the vocabulary of any language is enormous and cannot be categorised by humans. One of the unsupervised learning strategies for finding the most similar words for a given word is skip-gram.Skip-gram is a technique for predicting the context word for a target word. It's the opposite of the CBOW algorithm. The target word is entered, and the context words are displayed. This challenge is complicated by the fact that there are multiple context words to anticipate.

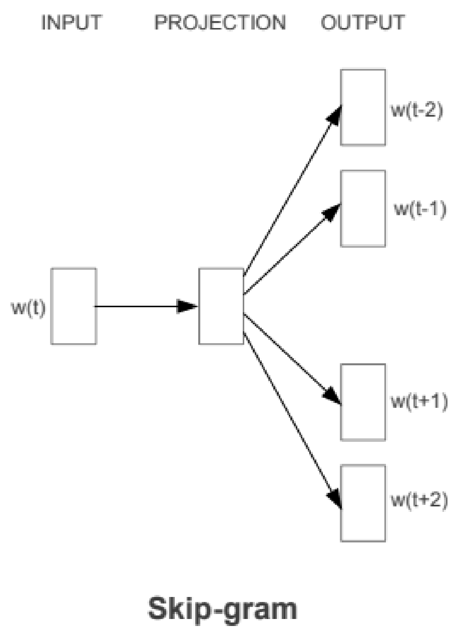


Fig 6 : The Skip-gram model architecture (Source: [16])

Here, the input given or the target word is represented by w(t). One hidden layer is present which calculates the dot product between the weight matrix and the input vector w(t). In the hidden layer there is no activation function being used. In the hidden layer, the result of the dot product that has been generated will be passed to the output layer. Output layer computes the dot product between the output vector of the hidden layer and the weight matrix of the output layer. Then we apply the softmax activation function to compute the probability of words appearing to be in the context of w(t) at a given context location.

2.1.3.2 Node2Vec

The transformation of network structure into numerical representation, which can then be passed on to typical machine learning techniques, is a major difficulty when working with networks. Node2Vec is a user-friendly approach for mapping nodes in a graph G to an embedding space. In general, the embedding space has less dimensions than the original graph G's number of nodes. Within the original graph, the method strives to preserve the initial structure. In other words, similar nodes in the network will have similar embeddings in the embedding space. Each node in the network is represented by a vector in these embedding spaces. Graph embeddings are frequently utilised as input features in machine learning problems including link prediction, community detection, classification, and other similar tasks. Node2Vec is a Random Walk based graph embedding method. Now, In probability theory, a random walk is a method for finding the likely location of a point subject to random motions, given the probabilities (which are the same at each step) of travelling a certain distance in a certain direction. Random walks are a type of Markov process in which future behaviour is unaffected by previous behaviour. In general, scientists find it challenging to graphically communicate the data they're working with when dealing with very huge graphs. To examine how a graph looks, one typical technique is to construct node embeddings for it and then visualise the embeddings in a lower-dimensional space. This enables us to detect possible clusters or groups forming in very vast networks on a visual level. To grasp the concepts of node2vec, a basic understanding of word2vec is required.

- **Random Walk :** Understanding what random walks are and how they work is essential to comprehending how node2vec operates. It will be discussed from a high-level perspective. As a high-level perspective, the simplest way to compare a random walk is to walk. If we assume that each step we take is determined probabilistically, then we have progressed in a specific direction based on a probabilistic outcome at each index of time. This algorithm investigates the relationship between each step and its distance from the beginning position. We could now be wondering how these odds of going from one node to the next are determined. The following formula is introduced by Node2Vec for estimating the likelihood of travelling to node x if we were previously at node v. Here, π_{vx} is the unnormalized transition probability between nodes x and v and z is the normalisation

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

(Equation obtained from [12])

constant.Therefore, if there is no edge linking x and v, the probability is 0, but if there is an edge, a normalised probability of travelling from v to x is found. If each edge had a weight, it would be the simplest method to impose a bias to impact the random walks. In the case of unweighted networks, however, this would not work. A guided random walk with two parameters, p and q, has been presented to solve this problem. The probability of a random walk returning to the previous node is given by p, and the probability of a random walk passing through a previously unknown region of the graph is given by q.

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

(Equation obtained from [12])

Here, the shortest path between nodes t and x is represented by d_{tx} . It can be visually illustrated as:

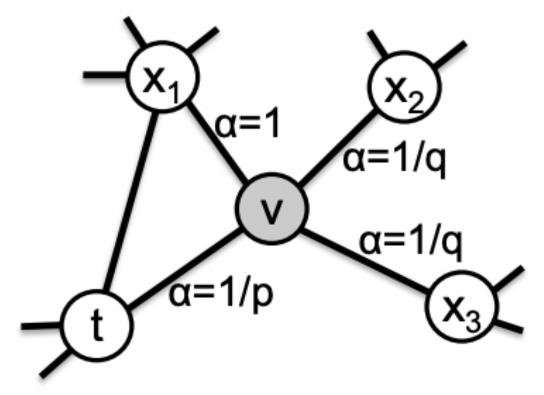


Fig 7 : Illustration of the random walk procedure in node2vec. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α. [12]

- **Skip-Gram Architecture :** When an input word is present, the skip-gram model is a simple neural network with one hidden layer trained to predict the chance of a given word being present. As seen in the diagram below, the process can be visually illustrated as:

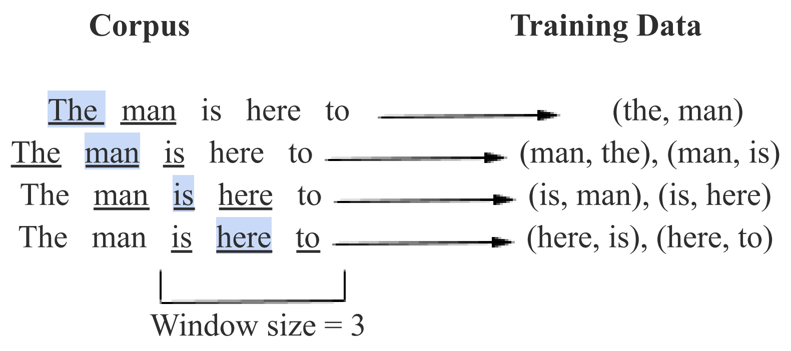


Fig 8 : Example of generating training data for skip-gram model. Window size is 3

As seen above, a target word is chosen from a corpus of literature using a rolling window. Pairwise combinations of the target word and all other words in the window make up the training data. This is the neural network's training data as a consequence. We can essentially output a likelihood of a word being a context word for a specific target once the model has been trained.

A corpus can be represented as an N-dimensional vector, with each element representing a word in the corpus. There should be a pair of target and context words during the training phase, and the input array will have 0 in all entries except the target word. The target word will have a value of 1. Each word's embedding representation will be learned by the hidden layer, resulting in a d-dimensional embedding space. A dense layer with a softmax activation function is used as the output layer. The output layer will essentially produce a vector of the same size as the input, with each element containing a probability. The similarity between the target term and the corresponding word in the corpus is shown by this likelihood.

So, now that we have briefly gone through the random walks method and skip-gram architecture, we can now come to the point where we can say that the procedure for using node2vec is quite straightforward; it starts by inputting a graph and extracting a series of random walks from it. The walks can then be seen as a directed series of words, with each node representing a single phrase. The skip-gram model is then fed the created random walks. As previously stated, the skip-gram model works with words and sentences, with each node in the random walk representing a word and the complete walk representing a sentence. The skip-gram model produces an embedding for each node as a consequence (or word in this analogy).

2.1.3.3 DeepWalk

Because graph data structures may describe complex relationships, new approaches to study and classify entities defined by their interactions have emerged. While these analyses are effective in identifying diverse structures within communities, they lack the ability to encode graph features for use in traditional machine learning techniques. Co-interactions inside graphs can be collected and encoded by basic neural networks into embeddings usable by the aforementioned ML methods with the help of DeepWalk. While there are papers that provide easy introductions to the DeepWalk algorithm, there are few that include code and describe implementation specifics for these systems that could be discovered. Model parametrization, deployment considerations, and handling unseen data are all covered in detail. DeepWalk can transform a graph from force layout visualisation to vector space visualisation while maintaining some of the structural properties. DeepWalk makes use of random path-making over graphs to uncover latent patterns in the network, which are subsequently learned and encoded by neural networks to produce our final embeddings. These random pathways are created in a very straightforward manner: Starting at the target root, a random neighbour of that node is chosen and added to the path, followed by another random neighbour of that node, and so on until the necessary number of steps has been taken. This regular sampling of network pathways provides a list of product-ids in the e-commerce case. These IDs are then treated as if they were tokens in a sentence, and a Word2Vec model is used to learn the state-space from them. The DeepWalk procedure can be explained in the following simple steps:

i. Perform N "random steps" starting from that node for each node.
ii. Each walk should be treated as a series of node-id strings.
iii. Train a word2vec model on these string sequences using the Skip-Gram technique, given a list of them.

2.1.4 Application of GNN

Graph-structured data can be found almost everywhere. The challenges that GNNs solve can be divided into the following groups:

1. **Node Classification :** Here, the goal is to use the labels of their neighbours to determine the labelling of samples (shown as nodes). Problems of this nature are usually taught in a semi-supervised manner, with only a portion of the graph labelled.

2. **Graph Classification :** Here, the goal is to categorise the entire graph into distinct groups. It's similar to picture classification, except the aim is now a graph. Graph classification has a wide range of applications, from detecting whether a protein is an enzyme or not in bioinformatics to categorising articles in natural language processing or social network analysis.

3. **Graph Visualisation :** At the intersection of geometric graph theory and information visualisation, this is a field of mathematics and computer science. It is focused with the visual depiction of graphs that helps the user comprehend the graphs by revealing structures and anomalies that may be present in the data.

4. **Link Prediction :** In this case, the algorithm must comprehend the relationship between entities in graphs and attempt to forecast whether two entities are connected. In social networks, it's critical to infer social connections or suggest potential buddies to users. It's also been used to solve problems with recommender systems and forecast criminal linkages.

5. **Graph Clustering :** The clustering of data in the form of graphs is referred to as graph clustering. On graph data, there are two different types of clustering. Vertex clustering is a technique for grouping nodes in a graph into highly connected regions based on edge weights or edge distances. The second type of graph clustering considers graphs to be clustered objects that are grouped together based on their similarity.

6. **GNNs in computer vision :** Machines can differentiate and identify objects in pictures and videos using standard CNNs. Although more work needs to be done before machines can have the same visual intuition as humans. GNN architectures, on the other hand, can be used to solve image categorization challenges. Scene graph generation is one of these issues, in which the model attempts to interpret an image into a semantic graph of objects and their semantic relationships. Scene graph generation models take an image and discover and recognise things, as well as anticipate semantic relationships between pairs of objects. However, the number of GNN applications in computer vision continues to expand. It incorporates human-object interaction, image classification from a few shots, and more.

7. **GNN in Natural Language Processing :** Text is a sort of sequential data that can be characterised by an RNN or an LSTM, according to NLP. Graphs, on the other hand, are widely employed in NLP tasks due to their naturalness and ease of depiction. Recently, there has been a rise of interest in using GNNs to solve a variety of NLP challenges, including text categorization, machine translation using semantics, user geolocation, relation extraction, and question answering. Every node is an entity, and edges define the relationships between them. The challenge of question answering is not new in NLP research. However, it was constrained by the existing database. The methods can be generalised to previously undiscovered nodes using techniques like GraphSage (Hamilton et al.).

8. **GNNs in traffic :** In a smart transportation system, forecasting traffic speed, volume, or density of roadways in traffic networks is critical. STGNNs can be used to solve the traffic forecast problem. Considering the traffic network as a spatial-temporal graph, with nodes representing road sensors, edges representing the distance between pairs of nodes, and each node having the average traffic speed within a window as a dynamic input feature.

9. **GNNs in Chemistry :** GNNs can be used by chemists to investigate the graph structure of molecules or compounds. The nodes in these graphs are atoms, while the edges are chemical bonds.

10. **GNNs in other domains :** The use of GNNs is not restricted to the domains and tasks listed above. Program verification, programme reasoning, social influence prediction,

recommender systems, electrical health records modelling, brain networks, and adversarial attack protection have all been attempted using GNNs.

2.2 Graph Convolutional Networks

As previously discussed, Neural Networks have had tremendous success in the recent decade. However, early Neural Networks could only be implemented with regular or Euclidean data, despite the fact that a lot of data in the actual world has non-Euclidean graph structures. Recent advances in Graph Neural Networks have been driven by the non-regularity of data structures. Graphs can be found in a wide range of applications, from social research to biology to computer vision. Graphs' special capability allows them to capture the structural relationships between data, allowing them to extract more insights than studying data separately. However, solving learning problems on graphs can be difficult because (1) many forms of data, such as photos and text data, are not originally structured as graphs, and (2) the underlying connectivity patterns for graph-structured data are often complex and diverse. On the other hand, representation learning has had a lot of success in a lot of different fields. As a result, one possible option is to discover how to represent graphs in a low-dimensional Euclidean space while preserving graph features. Despite significant efforts to address the graph representation learning problem, many of them still suffer from shallow learning methods. As a result, different variations of Graph Neural Networks have been developed in recent years, with Graph Convolutional Networks (GCN) being one of them. GCNs are also one of the most fundamental Graph Neural Networks variations. We are already familiar with convolution layers in Convolutional Neural Networks, and 'convolution' in GCNs is essentially the same procedure. It is the process of multiplying the input neurons by a series of weights known as filters or kernels. CNNs may learn information from surrounding cells thanks to the filters, which operate as a sliding window across the entire image. The same filter will be utilised throughout the image within the same layer, which is known as weight sharing. When using CNN to identify images of cats vs. non-cats, for example, the same filter will be applied in the same layer to detect the cat's nose and ears.

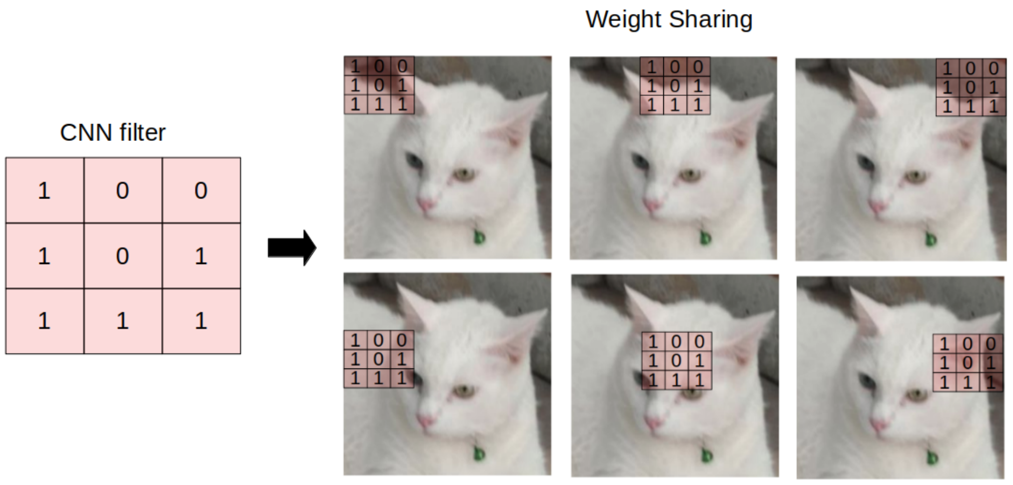


Fig 9 : The same weight (or kernel, or filter in CNNs) is applied throughout the image [20]

Similar actions are performed by GCNs, in which the model learns the features by analysing surrounding nodes. The main distinction between CNNs and GNNs is that CNNs are designed to work with normal (Euclidean) organised data, whereas GNNs are a generalised variant of CNNs in which the number of nodes connections varies and the nodes are not in any particular order (irregular on non-Euclidean structured data).

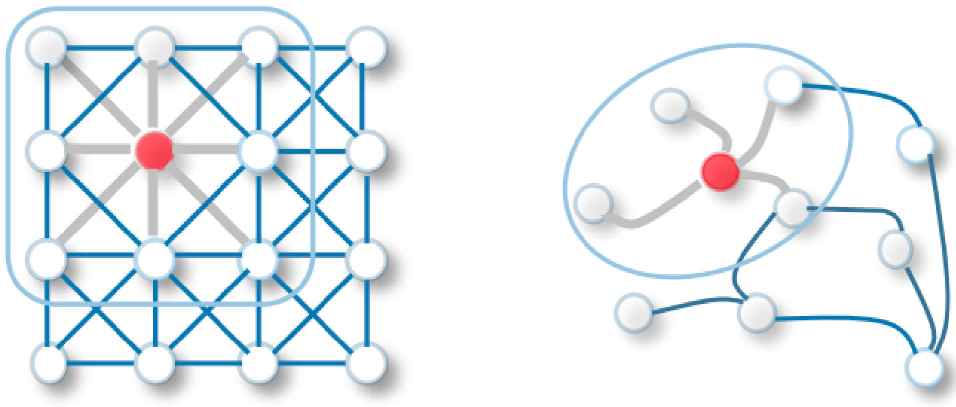


Fig 10 : Illustration of 2D Convolutional Neural Networks (left) and Graph Convolutional Networks (right) [24]

Spatial Graph Convolutional Networks and Spectral Graph Convolutional Networks are the two main types of GCN algorithms.

2.3.1 Concept behind GCN

GCN is a kind of convolutional neural network that can work directly on graphs and take advantage of structure information. It tackles the problem of categorising nodes (such as documents) in a graph (such as a citation network) when only a small subset of nodes have labels (semi-supervised learning).

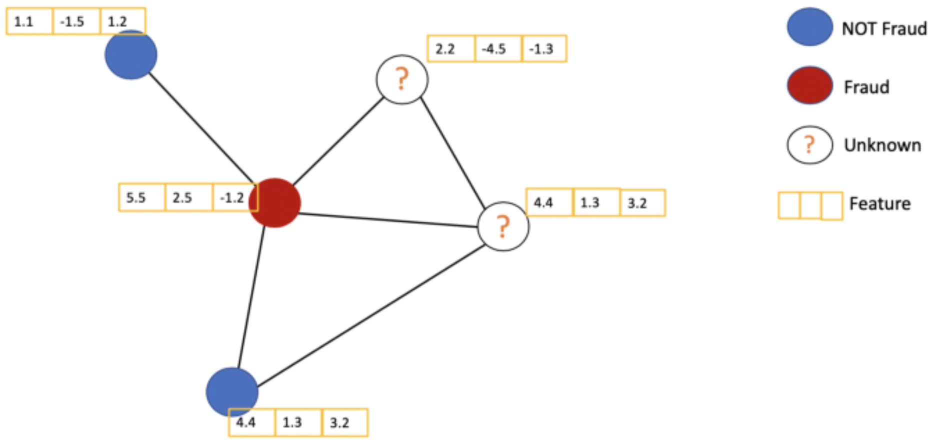


Fig 11 : Example of Semi-supervised learning on Graphs. Some nodes don't have labels (unknown nodes).[27]

The notion came from Images and was subsequently applied to Graphs, as the name "Convolutional" suggests. Graphs, on the other hand, are far more difficult when Images have a definite structure.

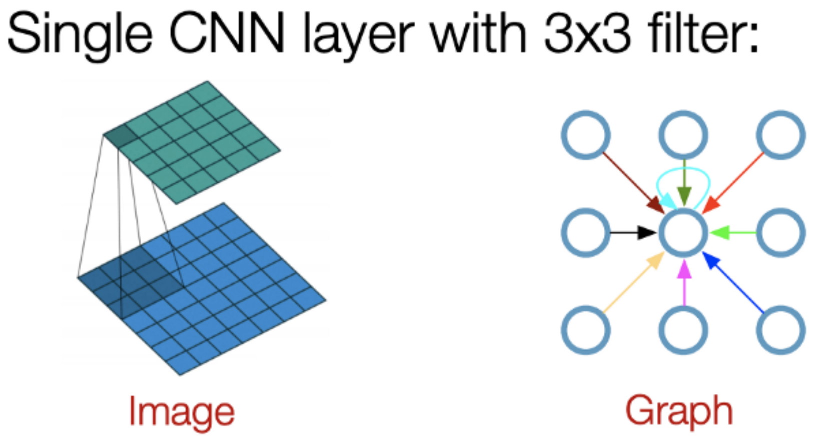


Fig 12 : Convolution idea from images to graphs.[29]

The main principle behind GCN is that for each node, the feature information from all of its neighbours are obtained, as well as the node's own feature. Let's pretend the average() function is being used. This will proceed in the same manner with all of the nodes. Finally, a neural network will be used to process these average results**.**

A simple example of a citation network is shown in the diagram below. The nodes represent research papers, while the edges indicate citations. Here, there is a pre-processing stage. The papers are turned into vectors (using NLP embedding, e.g., tf–idf) and used instead of using the raw papers as features.

Now, if we have a look at the green node. First, it can be seen that all of its neighbours' feature values are collected, including its own, and then their average values are considered. Then, the result will be fed into a neural network, which will provide a result vector.

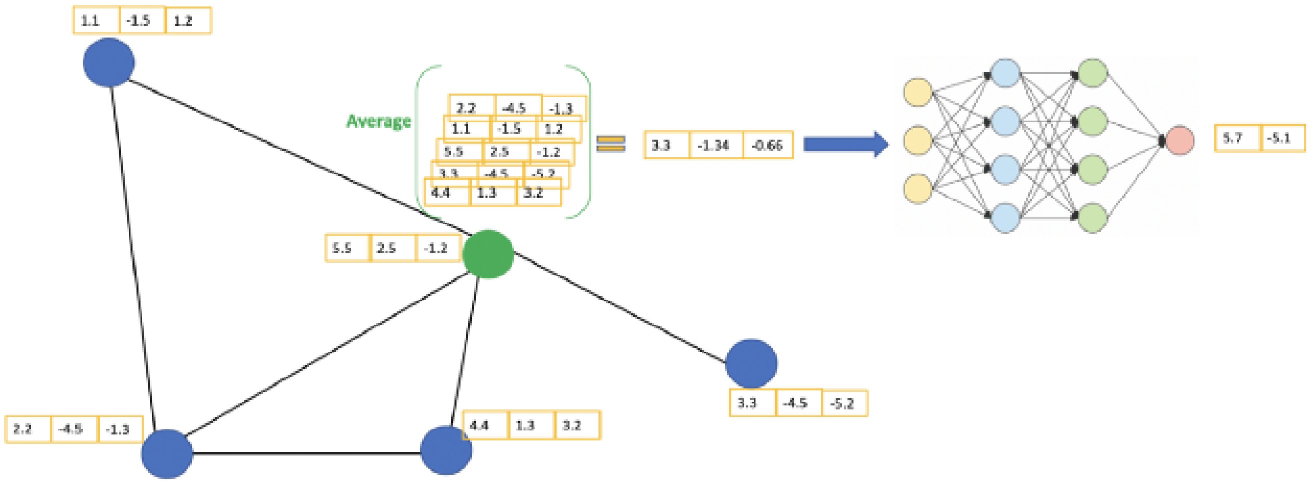


Fig 13 : The main idea of GCN. If the green node is considered, first, the average of all its neighbours are taken, including itself. After that, the average value is passed through a neural network. Note that, in GCN, a fully connected layer is used. In this example, we get 2-dimension vectors as the output (2 nodes at the fully connected layer). [27]

In practice, rather than just using the average function, some more sophisticated aggregate functions can be utilised. To create a deeper GCN, more layers can be stacked on top of each other. A layer's output will be used as the input for the next layer.

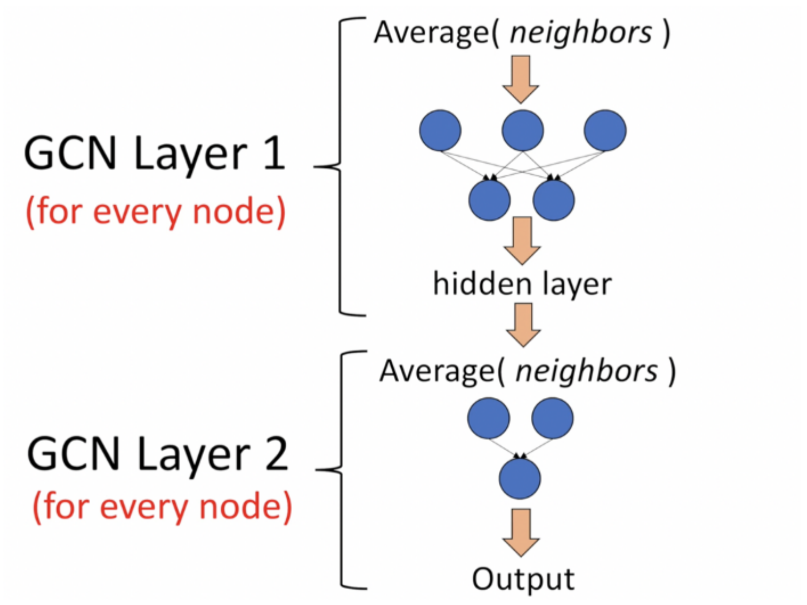
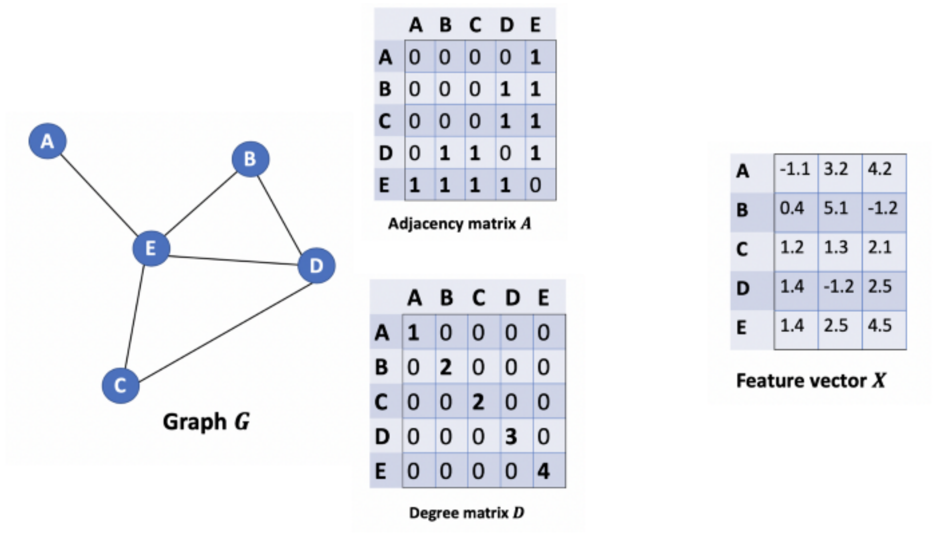


Fig 14 : Example of 2-layer GCN: The output of the first layer is the input of the second layer. Again, note that the neural network in GCN is simply a fully connected layer [28]

2.3.2 Intuition and the maths behind GCN

Given an undirected Graph $G = (V, E)$ with N nodes $v_i \in V$, edges $(v_i, v_j) \in E$, an Adjacency Matrix $A \in R^{N \times N}$ (binary or weighted), degree matrix $D_{ii} = \sum_j A_{ij}$ and feature vector matrix $A \in R^{N \times C}$ where, N is number of nodes, C is the number of dimensions of a feature vector.

Now if a graph G as below is considered



	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix *A*

	A	B	C	D	E
A	1	0	0	0	0
B	0	2	0	0	0
C	0	0	2	0	0
D	0	0	0	3	0
E	0	0	0	0	4

Degree matrix *D*

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector *X*

Fig 15: From the graph G, we have an adjacency matrix A and a Degree matrix D. We also have feature matrix X.[27]

The multiplication of A and X is done to get the feature values from neighbours of each node. Looking at the first row of the adjacency matrix, we can see that node A is connected to node E. The feature vector of E, which A connects to, is the first row of the resulting matrix (Figure below). In the same way, the second row of the resulting matrix is the sum of D and E's feature vectors. We can acquire the sum of all neighbours' vectors by doing so.

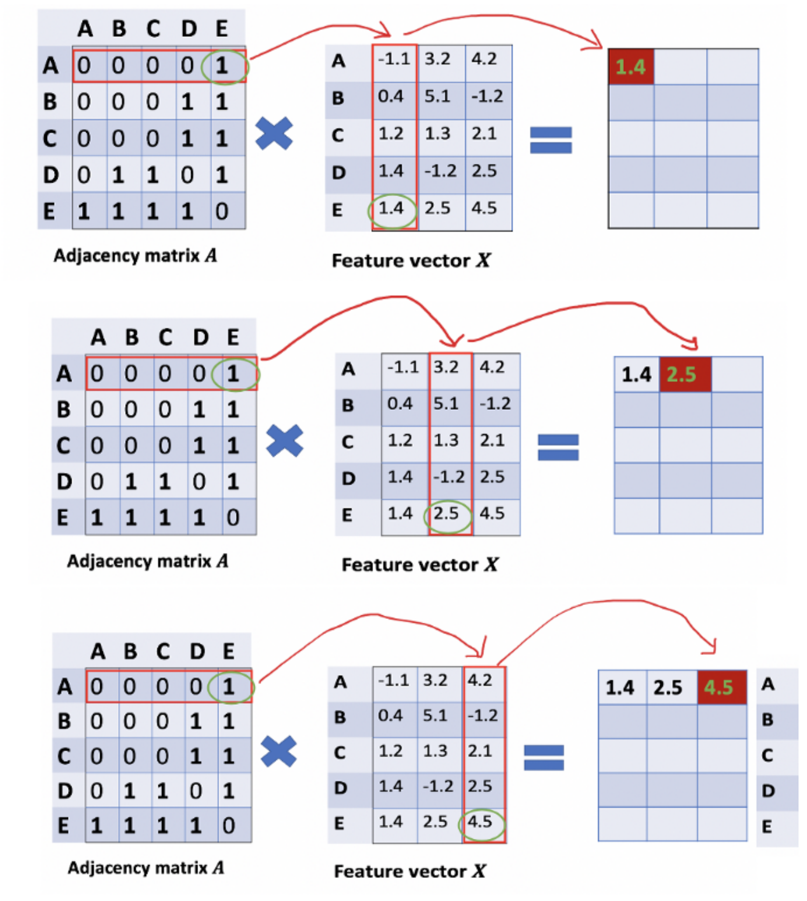


Fig 16 : Calculating the first row of the "sum vector matrix" AX [27]

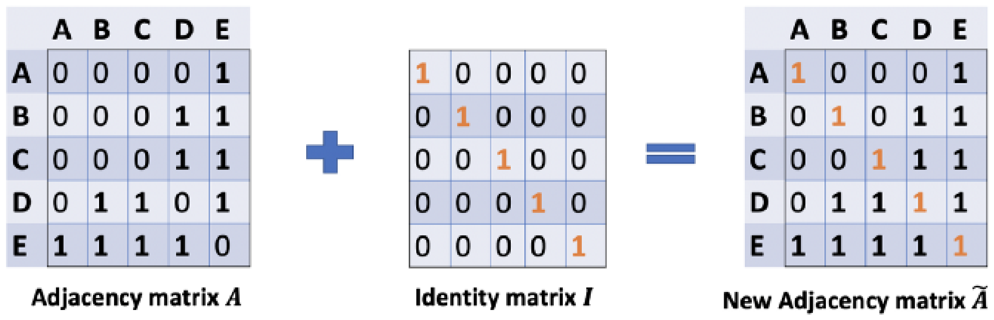
But there are still some problems that need to be improved.

1) We're missing the node's own feature. The first row of the result matrix, for example, should also include node A's characteristics.
2) Instead of using the sum() function, we should use the average, or better yet, the weighted average, of the feature vectors of our neighbours. Why don't we employ the sum() method? The reason for this is that when employing the sum() function, high-degree nodes are more likely to acquire large v vectors, whereas low-degree nodes are more likely to have tiny aggregate vectors, resulting in explosive or vanishing gradients later on (e.g., when using sigmoid). Furthermore, the scale of input data appears to be a factor in neural networks. As a result, we must normalise these vectors to eliminate any potential difficulties.

Problem 1 can be solved by adding an *Identity matrix I* to *A* to create a new adjacency matrix $\tilde{A}$.

$$\tilde{A} = A + \lambda I_N$$

Using lambda = 1 (the node's feature is just as essential as its neighbours), we get $\tilde{A} = A + I$. Note that we can treat lambda as a trainable parameter, but for now, just set it to 1, in fact, lambda is just set to 1 in the study.

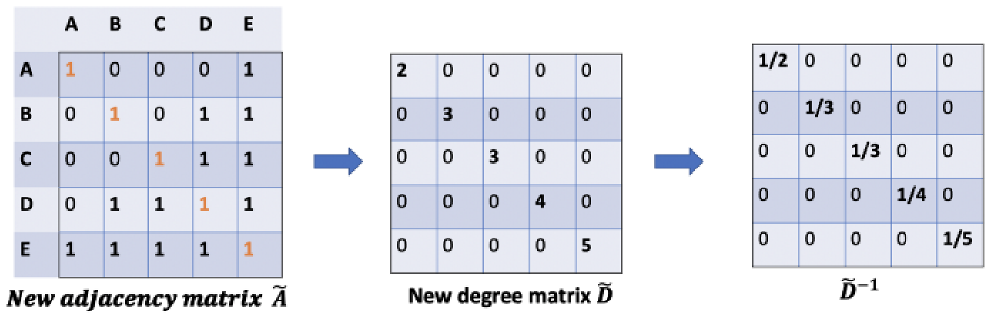


$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 0 & 0 & 0 & 1 \\ B & 0 & 0 & 0 & 1 & 1 \\ C & 0 & 0 & 0 & 1 & 1 \\ D & 0 & 1 & 1 & 0 & 1 \\ E & 1 & 1 & 1 & 1 & 0 \end{array} + \begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} = \begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 1 & 0 & 0 & 0 & 1 \\ B & 0 & 1 & 0 & 1 & 1 \\ C & 0 & 0 & 1 & 1 & 1 \\ D & 0 & 1 & 1 & 1 & 1 \\ E & 1 & 1 & 1 & 1 & 1 \end{array}$$

Fig 17 : The new adjacency matrix is created by adding a self-loop to each node. [27]

For the second problem, When scaling a matrix, we commonly multiply it by a diagonal matrix. In this scenario, we wish to average the sum feature, or scale the sum vector matrix $\tilde{A}X$ according to the node degrees mathematically. Our initial sense is that the diagonal matrix we're using to scale here is related to the Degree matrix $\tilde{D}$ (Why $\tilde{D}$, not D?). Because we're looking at the Degree matrix D of the new adjacency matrix $\tilde{A}$ now, not A).

Now, to normalise the sum vectors in order to pass information from neighbours to a specific node, the average function is again used. $\tilde{D}$ inverse ($\tilde{D}$^{-1}) plays a major role here in this case. Each of the elements in $\tilde{D}$ inverse is actually the reciprocal of its corresponding term of the diagonal matrix D


New adjacency matrix $\tilde{A}$
New degree matrix $\tilde{D}$
$\tilde{D}^{-1}$

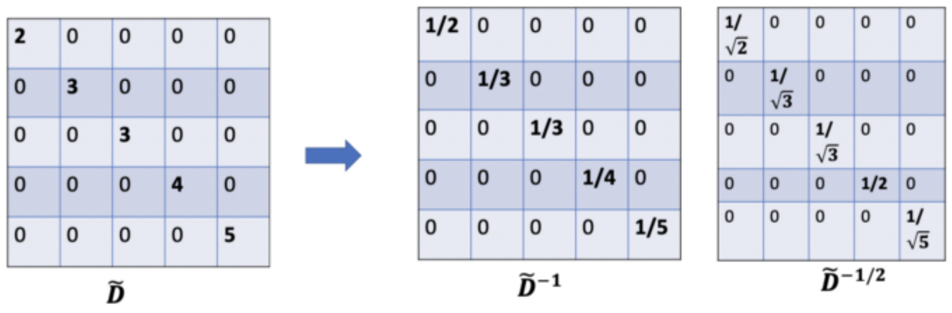
	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

2	0	0	0	0
0	3	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

Fig 18 : Generation of the matrix $\tilde{D}$ inverse[27]

As a result, we can calculate the average of all neighbours' feature vectors including itself by multiplying $\tilde{D}$ inverse and X. So, mathematically a new scaling strategy can be used instead of $\tilde{D}^{-1}\tilde{A}X$, which is $\tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}X$. This new scaler will give the weighted average. What we're doing here is putting more weight on low-degree nodes while reducing the influence of high-degree nodes. The concept behind this weighted average is that we think low-degree nodes have larger effects on their neighbours, whereas high-degree nodes have smaller impacts since their influence is spread out too thinly. In this case its actually being normalised twice, first for the row and second for the column. So, to rebalance the model, instead of using $\tilde{D}^{-1}$, $\tilde{D}^{-½}$ is used. So, the formula is further altered to $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}X$.


$\tilde{D}$
$\tilde{D}^{-1}$
$\tilde{D}^{-1/2}$

2	0	0	0	0
0	3	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

$1/\sqrt{2}$	0	0	0	0
0	$1/\sqrt{3}$	0	0	0
0	0	$1/\sqrt{3}$	0	0
0	0	0	1/2	0
0	0	0	0	$1/\sqrt{5}$

Fig 19 : Due to twice the normalisation "-1" is changed to "-1/2" [27]

Now, the forward model is to be formed using a 2-layer GCN

$$Z = f(X, A) = \mathrm{softmax}(\hat{A}\ \mathrm{ReLU}\ (\hat{A}XW^{(0)})\ W^{(1)}\ )$$

Where, ReLU ($\hat{A}XW^{(0)}$) is the first layer, $\hat{A}$ is the N✕N scaled adjacency matrix, X is the N✕C feature vector matrix, and $W^{(0)}$ and $W^{(1)}$ are the trainable weights of dimensions C✕H and H✕F respectively. N is the number of nodes, C is the number of dimensions of feature vectors. H is the number of nodes in the hidden layers, F is the dimensions of resulting vectors.

If we have a multi-classification problem with ten classes, for example, F will be set to ten. For the prediction, we send the 10-dimension vectors via a softmax function after they arrive at layer 2.The cross-entropy error over all labelled cases, where Y_{l} is the collection of node indices with labels, is used to calculate the Loss function.

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

2.3.3 The number of layers

The maximum distance that node characteristics can move is determined by the number of layers. With a single layer GCN, for example, each node can only get information from its neighbours. The information gathering procedure occurs independently and at the same time for all nodes. We repeat the information gathering procedure by stacking another layer on top of the first, but this time the neighbours already have information about their own neighbours (from the previous step). The maximum number of hops that each node can travel is determined by the number of layers. So, we may configure a reasonable amount for #layers based on how far we believe a node should get information from the networks. But, once again, we don't want to travel too far in the graph. We virtually get the whole graph with 6–7 hops, which makes the aggregation less useful. Experiments with shallow and deep GCNs revealed that a two- or three-layer model produces the greatest results. Furthermore, with a deep GCN (more than 7 layers), it is more likely to have poor performance. One option is to make use of the buried layers' residual connections.

Chapter 3

Multiview Networks

Multiple sorts of information on the same set of things are frequently provided by real-world data sets. Multi-view graphs, which consist of numerous separate sets of edges across the same nodes, are an excellent representation of this data. These can be used to investigate how creatures interact from various perspectives. The quality of conclusions made from the underlying data improves when numerous views are combined. Each view of the complicated system in a multi-view network is made up of one type of relationship, and all views share the same set of objects. For describing various relationships, a multi-view network can be constructed from a pair of single-view networks. It is more rational to collectively evaluate these numerous relationships because each view represents different relation kinds while sharing the same collection of nodes. We must investigate how different components within a system relate to one another in order to model and understand complex systems. Many relational data sets offer numerous perspectives on the same set of entities. A group of people's interactions with one another on a social networking platform, for example, can be used to classify them. We can only examine if a relationship exists between two people on this platform as a first estimate. However, we can investigate their interactions across other platforms, or across the platform's many modalities of communication, which give us several perspectives on the same group of people. Multi-omics measurements in single cell RNA sequencing data] and 2D projections of a single 3D object taken from several angles for 3D reconstruction are other instances of multi-view data sets.

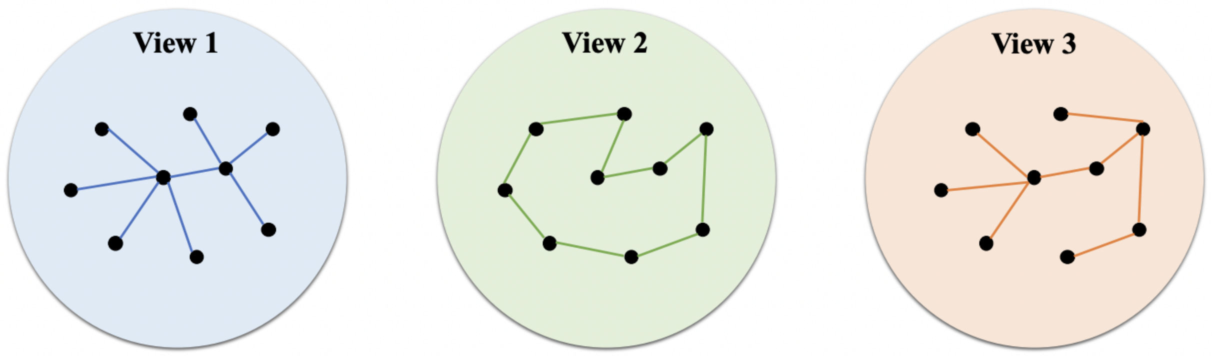


Fig 20 : An example of a multi-view network with three views. Each view corresponds to a type of proximity between nodes, which is characterised by a set of edges. Different views are complementary to each other. [23]

Mathematically, a multi-view network can be defined as following:

- A multi-view network is a graph **G = (V, E, X, φ)** whose edges are associated with more than one type.
- In such a multi-view graph, the mapping **φ : E → R, |R| > 1** associates an edge with an edge type.
- **V, E ∈ V × V, X ∈ R |V|×F** , and R represent the node set, the edge set, the node attribute matrix, and the set of edge types respectively.
- Alternatively, a multi-view network can be regarded as the union of a series of graph view ∪**r ∈ R{Gr}**,
  where **Gr = (V, Er), Er ∈ E** is the set of all edges of type r ∈ R.

Chapter 4

Optimization Techniques

In the field of Machine learning the model that performs well is mainly defined as a model that will give the most accurate predictions for a particular set of data. To achieve that principal goal of getting accurate predictions is done by using machine learning optimization techniques.
One of the core components of machine learning is optimization. Most of the machine learning algorithms focus on building an optimization model by learning the parameters in the objective function which is provided in the data. While performing optimization of a design, the design objective could be simply to minimise the cost of production or to maximise the efficiency of production. So, it can be termed that an optimization algorithm is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found. Hence, it can be said that optimization helps in fitting the learning algorithm on the training dataset. From fitting logistic regression models to training artificial neural networks, optimization is a challenging problem that is underlied by many machine learning algorithms. Optimization is the problem of finding a set of inputs to an objective function that results in a maximum or minimum function evaluation. It is the challenging problem that underlies many machine learning algorithms, from fitting logistic regression models to training artificial neural networks.

4.1 Multi objective optimisation

Multi-objective optimization is a branch of multiple criteria decision making that deals with mathematical optimization issues that require the optimization of numerous objective functions at the same time. Multi-objective optimization has been used in a variety of sectors of science, including engineering, economics, and logistics, where optimal decisions must be made in the face of trade-offs between two or more competing goals. Multi-objective optimization challenges involving two and three objectives include lowering cost while optimising comfort while purchasing an automobile and maximising performance while minimising fuel consumption and pollution emissions from a vehicle. Problems having only one objective function, often known as single-objective optimization, have been generally studied. Many engineering design issues, in practice, have many objective functions. These goal functions may be at odds with one another in numerous instances. For example, a design team may want to reduce weight while increasing the strength of a structural component, or increase vehicle performance while reducing fuel consumption and pollution emissions. Both problems are multiobjective optimization (MOO)

problems with two or more goals.Multi-objective optimization problems deal with objectives that are in conflict, i.e., one increases while the other drops. There is no single global solution, only a collection of them.
Mathematically, a multiobjective optimization problem can be formulated as follows:

$$\text{Minimise } \mathbf{F(x)} = (f_1(x), f_2(x), ..., f_k(x))^T$$
$$\text{Subject to } \mathbf{x} \in \mathbf{\Omega}$$

Here, the search space could be continuous or discrete and is represented by $\mathbf{\Omega}$, the decision variable is represented by x, where x = (x1,...,xn) $\in \Omega$ is. $\mathbf{F} : \mathbf{\Omega} \rightarrow R^k$ consists of k real-valued objective functions.

A vector x = $(x_1,\cdots,x_k)$ is considered to dominate another vector y = $(y_1,\cdots,y_k)$ in the multi objective problem taxonomy if and only if there is at least one m such that $x_j \leq y_j$, $\forall\ j \in \{1,\cdots,k\}$ and $x_m < y_m$, and the vector y dominated by vector x is denoted by $\boldsymbol{x} < \boldsymbol{y}$ **.** A solution $x^* \in \Omega$ is called a Pareto optimal solution if it is not dominated by any other solution. There can be numerous optimal solutions that are independent of one another. The Pareto Set (PS) is the collection of all these optimal solutions, whereas the Pareto Front (PF) is its image.

4.2 Genetic Algorithm

A genetic algorithm is a search heuristic based on Charles Darwin's natural selection hypothesis. This algorithm mimics natural selection, in which the fittest individuals are chosen for reproduction in order to create the following generation's children. Here, in this experiment the concept of genetic algorithm is being used as it is a randomised search and optimization technique guided by the principles of natural genetic systems. The notion of natural selection is taken in consideration of genetic algorithms. The selection of the fittest individuals from a population begins the natural selection process. They generate offspring who inherit the parents' qualities and are passed down to the next generation. If parents are physically active, their children will be fitter than they are and have a better chance of surviving. This procedure will continue to iterate until a generation of the fittest individuals is discovered. This concept can be used to solve a search challenge. A set of possible solutions to a problem is applied and the best ones are chosen.

A genetic algorithm consists of five steps, namely :

- **Initial population**

The procedure starts with a set of individuals known as a Population. Each individual is a potential solution to the problem being tried to solve.
Genes are a set of factors (variables) that characterise an individual and a Chromosome is made up of a string of genes (solution). The set of genes of an individual is represented by a string in terms of an alphabet in a genetic algorithm where binary values are commonly used (string of 1s and 0s). The genes are encoded in a chromosome.

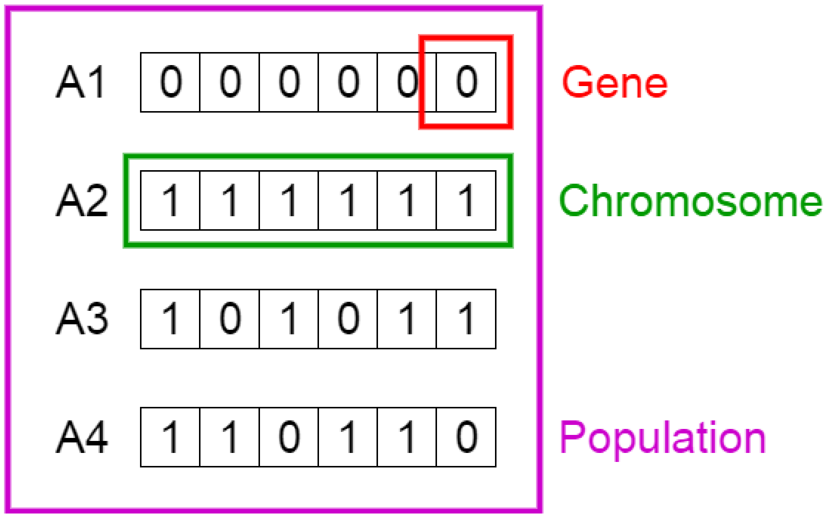


Fig 21: Population, Chromosomes and Genes [50]

- **Fitness function**

The fitness function determines an individual's level of fitness which is the ability of an individual to compete with other individuals. It assigns each individual a fitness score. The fitness score determines the likelihood of an individual being chosen for reproduction.

- **Selection**

The goal of the selection phase is to find the fittest individuals and let them pass on their genes to future generations.

Based on their fitness levels, two pairs of individuals (parents) are chosen. Individuals who are physically fit have a better probability of being chosen for reproduction.

- **Crossover**

The most significant part of a genetic algorithm is crossover. A crossover point is picked at random from within the DNA for each pair of parents to be mated.

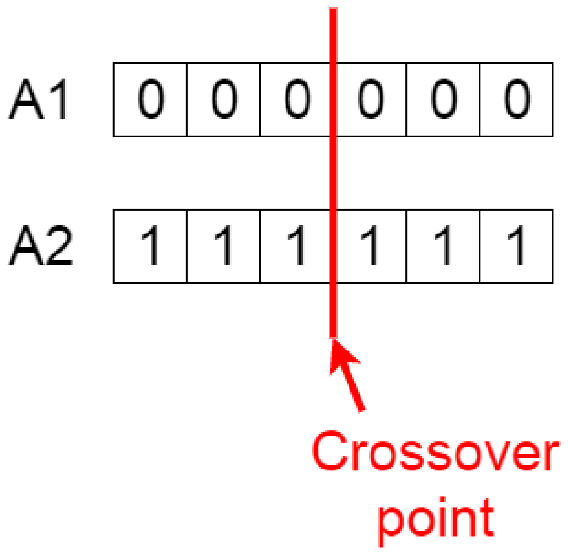


Fig 22 : Crossover point [50]

Parents' genes are exchanged among themselves to make offspring until the crossover point is achieved.

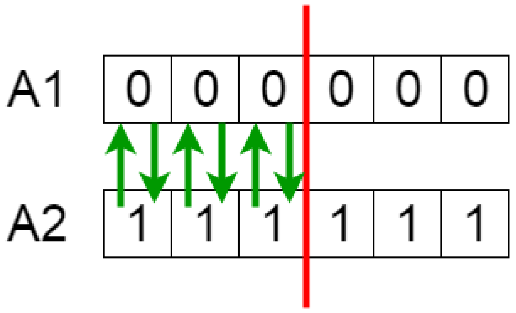


Fig 23 : Exchanging genes between parents [50]

The new offsprings are added to the population

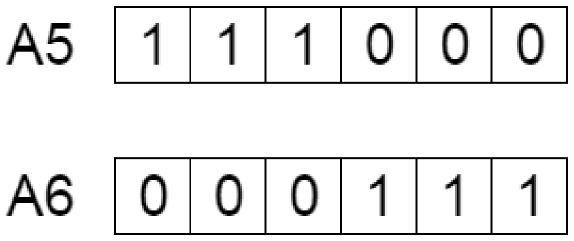


Fig 24 : New offsprings [50]

- **Mutation**

Some of the genes in the new offspring can be subjected to mutation with a low random probability. As a result, some of the bits in the bit string can be flipped.

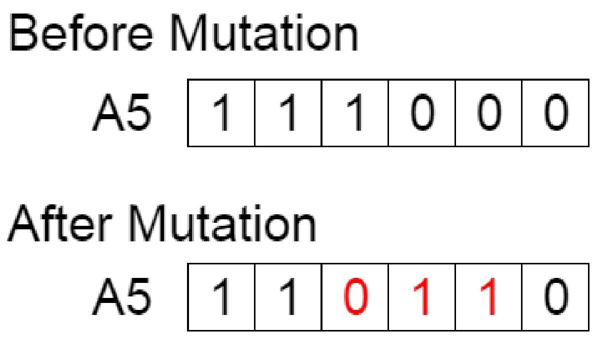


Fig 25 : Before and after mutation [50]

Mutation happens to maintain population heterogeneity and avoid premature convergence.

If the population does not produce offspring which are significantly different from the previous generation then it is termed as the population has converged, and in that scenario the genetic algorithm will get terminated. The genetic algorithm is thus considered to have generated a set of solutions to the given problem. Throughout the operations, the population size remains fixed. The individuals with the least fitness die when new generations emerge, thus making room for the future progeny. The above mentioned steps are repeated in order to develop individuals who are better than the previous generation in each new generation.

4.3 Non-dominated Sorting

Non-dominated sorting plays a significant part in determining the relative quality of solutions in a population; in many such algorithms, evolutionary computation has demonstrated tremendous performance in addressing many multi-objective optimization problems. In multi-objective evolutionary algorithms (MOEAs), non-dominated sorting is one of the critical steps to locate efficient solutions. A large percentage of computational cost of MOEAs is on non-dominated sorting for it involves numerous comparisons. Based on the dominance relationship of the solutions set, using non-dominated sorting, the solutions are separated into numerous non-dominated layers. Non-dominated sorting is primarily used to sort population solutions according to the Pareto dominance principle. This is crucial in the selection operation of many multi-objective evolutionary algorithms. Although there are various non-dominated sorting algorithms that have been presented based on various concepts, the majority of them adhere to the same structure (as illustrated in fig 26) and all of them achieve the same result (as illustrated in fig 27)

Individual A is said to dominate another individual B in non-dominated sorting if and only if there is no objective of A such that it is worse than that of B and if there is at least one objective of A that is better than that of B. If it is assumed that the solutions of a population P can be assigned to K Pareto fronts F_i, $i = 1, 2, \dots, K$ then a minimization issue can be solved without losing generality.

Non-dominated sorting usually starts by choosing all non-dominated solutions from a population P and assigning them to F1, which is the Front having the rank 1; then all the non-dominated solutions from the remaining solutions are selected and they are assigned to F2, which is the Front having rank 2); and this whole procedure is repeated until all the individuals have been assigned to a front. These fronts are relative to the given population, with no regard for the optimum of any process that may be responsible for their generation.

The non-dominated sorting algorithm was originally used in the multi-objective evolutionary algorithm's selection operation. Since then, numerous better variants of the original approach have been developed, all of which attempt to reduce the amount of redundant objective comparisons needed to achieve the correct dominance relationships among solutions. Fast non-dominated sorting is one of the few existing non-dominated sorting algorithms that is considered and used in this procedure of performing this experiment.

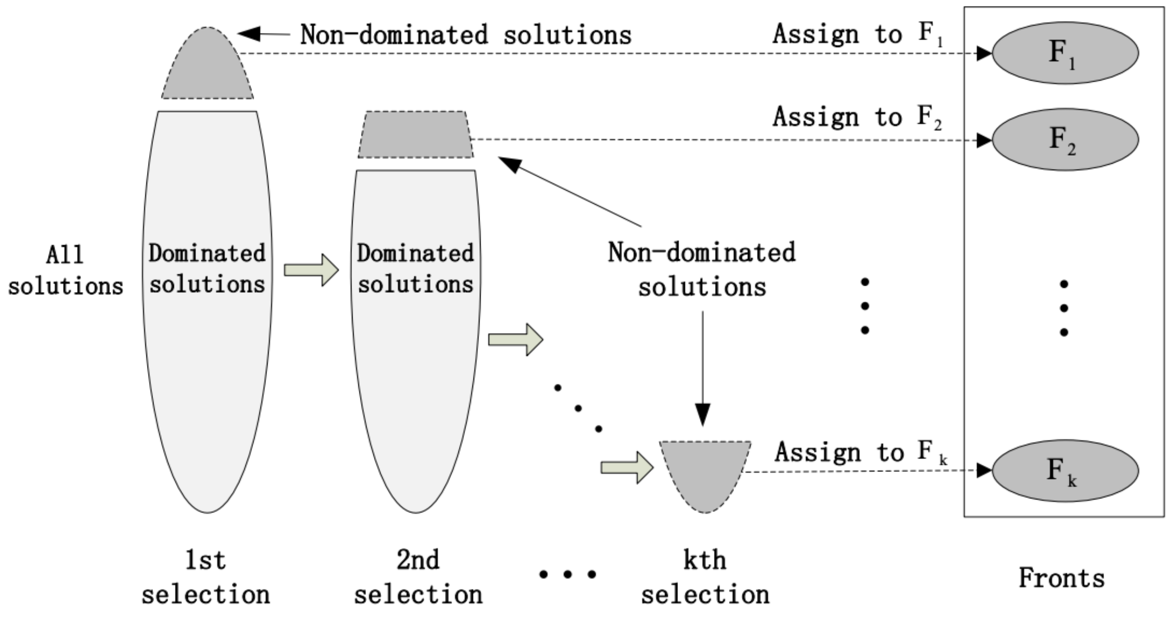


Fig 26 : Non-dominated sorting process of the solutions in a population [45]

4.3.1 Fast Non-dominated Sorting

The evolution of the population in the evolutionary multiobjective optimization algorithm is to eliminate the inferior solution and select the best solution by a certain strategy, such that the inferior solution is eliminated and the best solution is retained in the next generation of the evolutionary population so that the final excellent solution set can be found through step-by-step evolution.

The first updated version of the original non-dominated sorting algorithm is fast non-dominated sorting. It compares every solution to the others and saves the results to prevent duplicate comparisons between each pair of solutions. The first step in the fast nondominated sorting strategy is to determine each individual's corresponding dominating solutions number s_i and the individual number n_i in the evolutionary population who dominate the individual i. When an individual's corresponding n_i equals 0, they are transferred to the non-dominating individuals' first level. When the non-dominating individuals' dominating grade in a level is already decided, the level of all individuals under the influence of other individuals is no longer evaluated, according to the fast non-dominated sorting approach. Taking this into consideration, each of the individual's (let us say individual j) matching n_j in the individual i's dominating individual set s_i should deduct 1 for an individual's dominating individual set s_i.When n_j becomes 0 after subtraction, the individual j will be placed in the second non-dominant level. The rest of the evolutionary population follows the same procedure to determine which non dominating level they belong to. The lowest level's corresponding individuals are near to the actual Pareto front of the multiobjective optimization problem to be addressed after each individual in the evolutionary population finds their own non dominating level. As a result, the corresponding individuals at the lowest level are superior to the rest of the evolutionary population. So, it can be said that for a population which is having N solutions,it is required to perform $MN(N-1)$ objective comparisons which will eventually give a time complexity of $O(MN^2)$. This approach has a space complexity of $O(N^2)$ as for each solution i, it requires a set s_i to store the individuals that are dominated by i and a counter n_i to record the number of individuals that dominate i. The time and space complexity of a problem rapidly increases as the number of solutions grows.

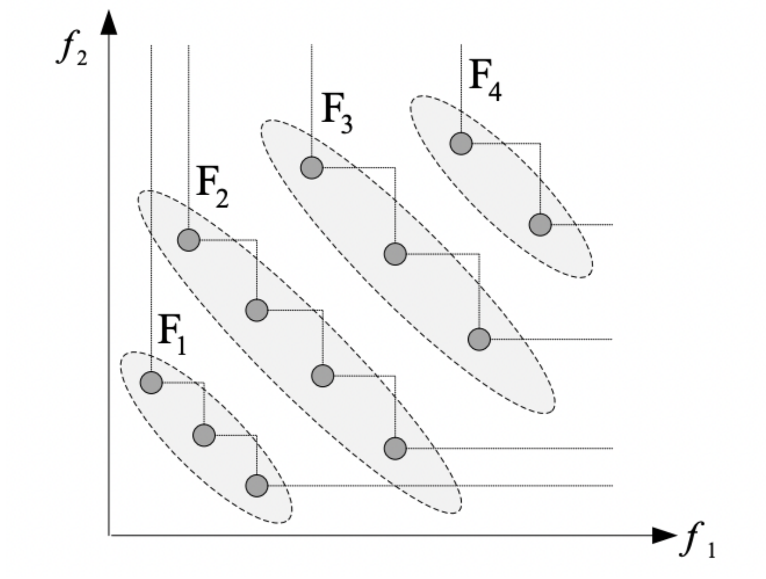


Fig 27 : An example of a Population with 12 solutions and 4 Pareto fronts [45]

4.4 Non-dominated Sorting Genetic Algorithm II (NSGA II)

NSGA-II is an evolutionary algorithm. Evolutionary algorithms were created as a solution to the issues that the traditional direct and gradient-based techniques have when dealing with non-linearities and complicated interactions:

- The initial choice of solution determines whether convergence to an ideal solution occurs.
- Most algorithms have a tendency to remain in a non-optimal state.

NSGA-II is one evolutionary algorithm that has the following three features:

1. It operates under the elitist principle that the elites of a population are given the chance to pass on their status to the following generation.
2. It uses a system that explicitly preserves diversity (Crowding distance)
3. It places particular emphasis on non-dominated solutions.

Below is mentioned the procedure of NSGA II :

Step 1: Population initialization Initialize the population based on the problem range and constraint.
Step 2: Non dominated sorting Sorting process based on non domination criteria of the population that has been initialised.
Step 3: Crowding distance Once the sorting is complete, the crowding distance value is assigned front wise. The individuals in the population are selected based on rank and crowding distance.
Step 4: Selection The selection of individuals is carried out using a binary tournament selection with a crowded-comparison operator .
Step 5: Genetic Operators Real coded GA using simulated binary crossover and polynomial mutation.
Step 6: Recombination and selection Offspring population and current generation population are combined and the individuals of the next generation are set by selection. The new generation is filled by each front subsequently until the population size exceeds the current population size.

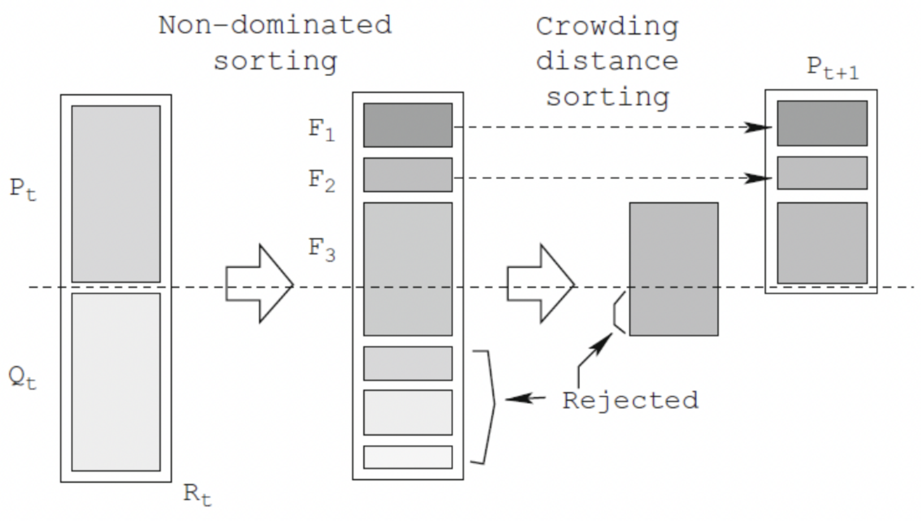


Fig 28 : Illustration of the process of NSGA II [38]

4.4 Non-dominated Sorting Genetic Algorithm III (NSGA III)

NSGA III is just a modified version of NSGA II with some slight changes in the selection mechanism. NSGA-III is based on Reference Directions instead of crowding distance as in NSGA II which is required to be provided when the algorithm is initialised. For problems with multiple objectives, the crowding distance measure does not always perform well. As a result, the NSGA-III selection mechanism is altered by doing a more systematic study of the members while adhering to the provided reference points. To guarantee diversity in the resulting solutions, NSGA-III makes use of a predetermined set of reference points. However, the selected reference points can either be provided preferentially by the user or predefined in a structured manner.

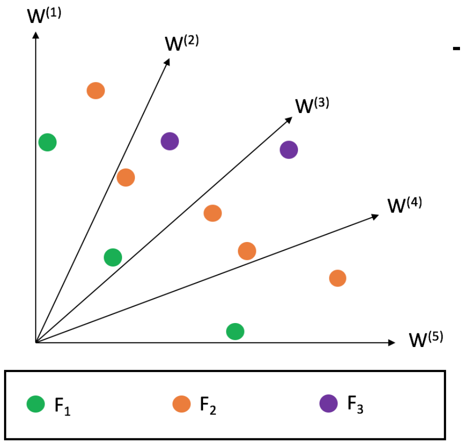


Fig 29 : Solutions selected from the splitting front after non-dominated sorting [61]

In NSGA III, at first the non-dominated sorting is done in the same way as in NSGA II.

Second, a few alternatives must be chosen from the splitting front. The underrepresented reference direction is first filled by NSGA-III. The solution with the lowest perpendicular distance in the normalised goal space is the one that survives if the reference direction has no solutions assigned. In the event that this reference line receives a second solution, it is given at random.

As a result, each reference line searches for a good representative non-dominated answer as the algorithm converges.

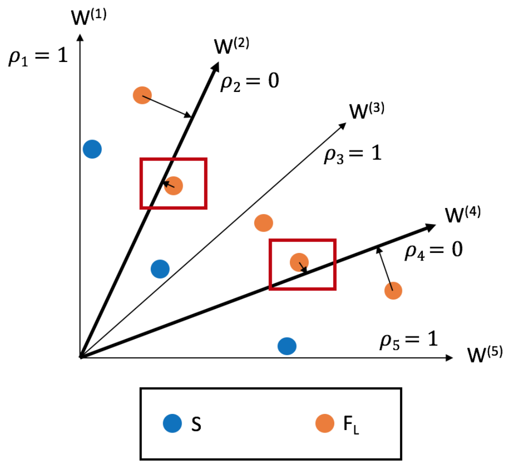


Fig 30 : Getting solution with respect to reference direction [61]

Determination of reference points in a HyperPlane : To guarantee the diversity of the answers found, a set of reference points must be established. On a normalised hyper-plan, various points are positioned with the same orientation along all axes. The formula for the number of reference points (H) is

$$H = \binom{C+g-1}{g}$$

Where g is the number of divisions to take into account for each objective axis, and C is the number of objective functions (for 3 objectives and 4 divisions, there will be 15 reference points). The produced reference points will be connected to the solutions when the reference points are placed on the hyper-plane and the solutions are defined using a Pareto front.

Normalisation of the population members : An ideal point of the current population must be determined, so the minimal value of each objective function must be identified (OF_i^{min} , i = 1, 2, ..., C) Then, each objective function will be moved subtracting $z_i^{min} = (OF_1^{min}, OF_2^{min}, ..., OF_C^{min})$ to the objective f_i . Then it is continued with the steps proposed by Deb and Jain (2014) in order to generate a hyper-plane (solving the Pareto fronts where the objective solutions are different scale).

Association among reference points and solutions : Each population member must be connected to a reference once each objective function has been normalised. For each point, a reference line connecting it to the origin point is defined. The perpendicular distance between each reference line and each population member is then calculated. Finally, a population member is paired with the reference point that is closest to them on the reference line.

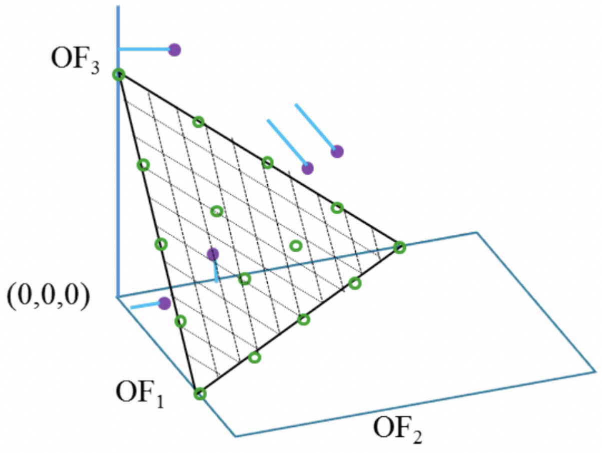


Fig 31 : Association of solutions with reference points [62]

Niche preservation operation : One or more solution members may be connected with a reference point, but the solution that is closest to the reference point must be maintained (perpendicular to the reference line, Deb and Jain 2014).

Genetic operations : The NSGA II algorithm's genetic operators will be utilised to create the next generation of children. Deb and Jain's (2014) recommendations led us to set the population size near to the number of reference points (H) in order to give each population member equal weight.

Chapter 5

Survey of Literature

5.1 Multi View Graph Attention Networks

In a multi-view network, where each view represents a different kind of interaction between nodes, multi-view graph embedding aims to learn low-dimensional representations of nodes that capture diverse relationships. Numerous graph embedding techniques currently in use focus on single-view networks, which can only describe one straightforward sort of proximity interactions between objects. However, the majority of complex systems in real life have a variety of interactions between the entities. This paper proposes Multi-view Graph Attention Networks, a novel method of graph embedding for multi-view networks (MGAT). An attention-based architecture, the network parameters of which are restricted by a novel regularisation term, for learning node representations from each individual view is being investigated. A view-focused attention strategy is investigated to aggregate the view-wise node representations in order to cooperatively integrate various sorts of relationships in various views. The suggested approach outperforms current state-of-the-art baselines when the algorithm is tested on a variety of real-world datasets.

In this paper, a set of view-specific node representations are learned once each individual view is first embedded. A regularisation term is being developed to promote the similarity of the network parameters of various viewpoints since there are mutual information exchanges among them. Then, an attention-based approach is investigated to determine the weights of views by allowing nodes to focus on the view that contains the most useful information. Finally, the learnt weights further combine the node representations for each distinct perspective into a global node representation. The global representation is then used for all graph analysis operations after that.

This MGAT model has two primary pieces: an attention-based aggregation part for multi-view sections and a graph embedding part for individual views. A limited GAT model is used to effectively encode each individual view during the graph embedding component's mining of the node proximities. Different views are integrated for more thorough node representations by the attention-based aggregation component.

In this study, MGAT, a brand-new multi-view graph embedding method has been presented. More specifically, a novel attention-based mechanism for learning the weights and encouraging collaboration across various views as well as an unique regularisation term to promote similarity among the network parameters of each individual view has been investigated. This method can more successfully encode multi-view networks because it considers the unique characteristics of each view and bases node treatment on the amount of information in different views.

5.2 MVN2VEC: Preservation and Collaboration in Multi-View Network Embedding

Multi-view networks are widely used in applications in the real world. Network embedding has since become a successful representation learning strategy for networked data. Two such aims, referred to as preservation and collaboration, are expressly articulated in the practice of embedding real-world multi-view networks. This paper's discussion covers the in-depth consideration of these two goals. Additionally, the MVN2VEC methods are suggested to investigate how different levels of preservation and collaboration can affect embedding learning and investigate whether modelling them simultaneously could improve embedding quality. The validity and significance of preservation and collaboration as two aims for multi-view network embedding are proven by experiments on a number of synthetic datasets, a sizable internal Snapchat dataset, and two open datasets. The tests conducted further show that improved embedding can be attained without overly complicating the model or necessitating more supervision by modelling the two objectives simultaneously. To design embedding methods for multi-view networks, the primary challenge lies in how to use the type information on edges from different views. For that reason two such objectives are identified, preservation and collaboration, from the practice of embedding real-world networks. Need for preserving unique information carried by different views is termed as preservation; need for collaboration as well as preservation may coexist in the same network. If a user pair in a social network shows an edge in either the message exchange view or the post viewing view, it is likely that these two people are content to be connected. In this case, the ideas might be complementary, and combining them might result in better outcomes than combining them separately. Collaboration is what we refer to as having such a synergistic impact in jointly embedding many perspectives, which is also the main idea underpinning the majority of existing multi-view network techniques.

The goals that are particular and significant to multi-view network embedding, namely preservation and collaboration, are recognised and examined in this research. The viability of simultaneously modelling both aims for improved embedding is then investigated, and two multi-view network embedding techniques are suggested. Three real-world multi-view networks from different sources, including two public datasets and a sizable internal Snapchat dataset, were used in experiments with a variety of downstream objectives. The findings supported the validity and significance of cooperation and preservation as two optimization goals and illustrated the potency of the suggested MVN2VEC approaches. Future study will simulate various levels of preservation and collaboration for various pairings of views in multi-view embedding in light of the existence of the defined objectives. Exploring supervised techniques for task-specific multi-view network embedding that combine model preservation and cooperation is also rewarding.

5.3 Multi view network representation learning algorithm research

A significant area of study in network data mining is network representation learning. The multi-scale relationships of network vertices are embedded into the low dimensional representation space using a novel multi-view network representation algorithm (MVNR) that is proposed in this research. In contrast to current methods, MVNR explicitly uses k-step networks to encapsulate higher level information. The matrix forest index is also introduced as a type of network feature that may be used to balance the representation weights of various network views. Research is being done on the relationship between MVNR and a number of outstanding scientific discoveries, such as DeepWalk, node2vec, GraRep, and so on. The experiment is run on a variety of real-world citation datasets, and the results show that MVNR performs better than certain fresh neural matrix factorization methods. In particular, the effectiveness of MVNR is demonstrated for network classification, visualisation, and link prediction tasks.

This research introduces a novel network representation learning (MVNR) method that directly incorporates global characteristics and higher order adjacent relations. When it comes to network classification tasks, MVNR performs better than network representation techniques that rely on single view features, such DeepWalk, Line, and node2vec. Additionally, MVNR beats the current network representation learning approach based on higher order features like GraRep and

NEU. To compare the weights of various k-step networks, the Matrix Forest Index (MFI) is presented. This method yields various weights for the representation vectors of various k-step networks in a common representation vector space. This procedure corrects a significant flaw in the GraRep algorithm. The network structure features are also used to calculate MFI features. In order to solve the sparse problem of the structure feature matrix in sparse networks, it can supply sufficient structure features. The MVNR algorithm produces superior visualisation results than the node2vec and DeepWalk algorithms. It exhibits greater coherence and a more distinct classification border. As a result, MVNR can learn the vectors of discriminative representation. It consequently performs superbly in link prediction tasks, outperforming the baseline methods utilised in this research in terms of link prediction performance.

So, basically, in this paper, a unified network representation learning framework (MVNR) that can incorporate global information and nearby relationships in a lower dimension's representation space is being offered. Future research would examine how this system may be applied to different representation learning problems. The experimental findings demonstrate that the suggested MVNR can simultaneously capture global features and higher order relationships between vertices. In the meantime, MVNR beats the well-known global network representation-learning algorithms (Line), GraRep, and NEU in terms of classification performance.

5.4 Multi-view Collaborative Network Embedding

Multiple views of real-world networks are frequently present, with each view describing a particular sort of interaction among a common set of nodes. On a network for sharing videos, for instance, two user nodes can be linked in one view if they have similar favourite videos, but they can also be linked in another view if they have similar subscribers. Multi-view networks, as opposed to conventional single-view networks, maintain distinct semantics to support one another. In this paper what is being suggested is MANE, a multi-view network embedding method to learn low-dimensional representations. MANE is dependent on diversity and collaboration, much like earlier research; diversity allows views to preserve their own semantics, while collaboration allows views to cooperate. To get better node representations, a fresh type of second-order collaboration that had not been studied before was also found and further incorporated into the framework. Furthermore, MANE+ is suggested as an attention-based extension of MANE to describe node-wise view importance because each view frequently has varied relevance w.r.t. different nodes. Finally, thorough tests are carried out on three open, real-world multi-view networks, and the outcomes show that the models regularly outperform cutting-edge methods.

This research presents a novel approach to second-order collaboration on multi-view networks. A brand-new multi-view network embedding approach called MANE has been suggested, incorporating several properties into a unified framework. Through an attention mechanism, MANE+ is further enhanced to capture node-wise view relevance. Three publicly available datasets have been the subject of in-depth investigations, and the superiority of the methodologies has been demonstrated empirically.

Three categories of node pairs can be used to unify the three properties of a multi-view network as a whole. Intra-view pairs, which can be produced using random walks in each view, are taken into consideration in the first category. It is able to capture the variety of various viewpoints by modelling intra-view pairs in each view. Cross-view, intra-node pairs, which are formed when occurrences of the same node (i.e., intra-node) across two views (i.e., cross-view), are taken into consideration in the second category. The first-order collaboration in such a pair can be observed by aligning the node representations. In the third category, cross-view and cross-node pairings are taken into account to capture second-order cooperation. Here, a node in one view forms pairs with distinct nodes (i.e., cross-node) in another view (i.e., cross-view) depending on their affiliations in each view. The final embeddings are created by combining the three categories of node pairs' collective training results to create a single set of embeddings for each view. In particular, MANE+ learns a different weight for each view while MANE assumes equal weight across views during aggregation.

5.5 Multi view Knowledge graph embedding for entity Alignment

Research is being done on the issue of embedding-based entity alignment amongst knowledge graphs (KGs). The relational structure of entities has largely been the subject of earlier investigations. Some go beyond and include different aspects, including qualities, for refining. However, a sizable number of entity properties remain undiscovered or are not handled equally when combined, which reduces the accuracy and sturdiness of embedding-based entity alignment. This research proposes a novel framework for learning entity alignment embeddings by integrating various viewpoints of entities. Specifically the entities are embedded using various combination tactics based on views of entity names, relations, and characteristics. Additionally, a few cross-KG inference techniques to improve alignment between two KGs is developed. The proposed framework greatly outperforms the most advanced embedding-based entity alignment algorithms, according to studies on real-world datasets. The chosen viewpoints, cross-KG inference, and combination procedures all help to increase performance.

An entity alignment framework using multi-view KG embedding that learns entity embeddings from three representative KG views is proposed in this paper. Here, two cross-KG training techniques for alignment inference are shown. Additionally, three other types of embedding combinations are being developed. The framework's usefulness was demonstrated by tests on two real-world datasets. Future research will look into additional practical viewpoints (such entity types) and examine cross-lingual entity alignment.

5.6 Community detection in node-attributed social networks: How structure-attributes correlation affects clustering quality

The majority of parametric community detection (CD) methods focus on proposing new techniques and rarely pay much attention to the general analysis of how node-attributed social networks (ASNs) properties affect the corresponding CD. To fulfil the gap, in this paper which investigated CD quality dynamics for ASNs with different structure-attributes correlation. The model under consideration is a weight-based one that interpolates between the so-called fixed and non-fixed topology cases and generalises a wide class of known models. The calculations indicate that the presence of correlation noticeably affects CD quality. This makes the common suggestion that "adding attributes to structure leads to better CD results" questionable in certain cases.

An explicit control of the components, interpolation between fixed and non-fixed topology, and generalisation of a large class of weight-based models for CD in ASNs are all features of the weight-based model under study. In this model, it is theoretically investigated how an ASN's structure-attributes correlation and the topology concept chosen have an impact on CD quality. Furthermore, experiments using synthetic ASNs and original/modified real-world ASNs are used to test the theoretical conclusions. Also presented is the feature selection paradigm as it relates to

ASN CD quality.

The generalised weight-based model, which offers explicit component control and interpolates between the Fixed and Non-Fixed topology scenarios, is taken into consideration in this work. We first considered theoretically how ASN structure-attributes correlation and fixed and non-fixed topology affect CD quality dynamics in the setting of the model. In particular, it is demonstrated that, compared to the Fixed topology scenario, the Non Fixed topology typically widens the value range of structural and attributive measures, allowing one to attain lower and higher values for the former and the latter, respectively. Furthermore, it is argued that in an ASN, a strong connection between structure and attributes results in a weak dependence of the quality measures on. This suggests that the common suggestions that "adding attributes to structure leads to better CD results" are not universal recipes. Future research is needed to study how ASN properties affect CD quality within other fusion models. It was observed that a simultaneous usage of structure and attributes is hardly reasonable for ASNs with strong structure-attributes correlation.

5.7 A Multi-Objective Genetic Algorithm for Community detection in networks

A multi-objective genetic algorithm to uncover community structure in complex networks is proposed. The algorithm optimises two objective functions able to identify densely connected groups of nodes having sparse interconnections. It generates a set of network divisions at different hierarchical levels in which solutions at deeper levels, consisting of a higher number of modules, are contained in solutions with lower number of communities. Experiments on synthetic and real life networks show the capability of the method to successfully detect the network structure. MOGA-Net is a multi-objective approach to discover communities in networks by employing genetic algorithms.

The method optimises two objective functions that are both efficacious in detecting modules in complex networks. Each of these solutions corresponds to a different tradeoff between the two objectives and thus to diverse partitioning of the network. This gives a great chance to analyse several clusterings at different hierarchical levels.

The Non-dominated Sorting Genetic Algorithm (NSGA-II), introduced by Srinivas and Deb and implemented in the Genetic Algorithm and Direct Search Toolbox of MAT- LAB, is the Multi-Objective Genetic Algorithm (MOGA) being employed here. The NSGA-II creates a population of competitors and ranks them according to non-dominance. For the purpose of using NSGA-II, MOGA-Net has been modified with a special population type that accurately depicts a network's partitioning and is equipped with two complementing aims.

A multiobjective evolutionary method for locating communities in intricate networks was given in the paper. It has been demonstrated that the method accurately detects communities and is competitive with cutting-edge techniques. In comparison to single objective techniques, the algorithm has the benefit of offering a variety of solutions at various hierarchical levels by allowing for the analysis of the network structure at various resolution levels.

5.8 Graph Neural Network Encoding for Community detection in Multi Attribute Networks

In order to address the community detection problem in complex attribute networks, a graph neural network encoding approach for a multiobjective evolutionary algorithm is first described in this research. Each edge in an attribute network is linked to a continuous variable in the graph neural network encoding method. A continuous valued variable is, a concatenation of the continuous variables linked to the edges is converted into a discrete valued community grouping solution using non-linear transformation. The attribute homogeneity of the nodes in communities is also evaluated using two objective functions for single- and multi-attribute networks, respectively. For the modified community detection issue with continuous decision variables, a multiobjective evolutionary algorithm (MOEA) based on NSGA-II, referred to as continuous encoding MOEA, is created based on the novel encoding approach and the two objectives. The created algorithm outperforms several well-known evolutionary and non-evolutionary based algorithms significantly, according to experimental results on single- and multi-attribute networks of various sorts. The suggested graph neural network encoding method is proven to be effective because the fitness landscape analysis confirms that the modified community detection issues have smoother landscapes than the original problems.

Two factors in particular drove the development of this change. First off, unlike continuous problems, discrete problems do not have enough relevant local information to aid in searching. Second, the fitness landscape is not smooth because the local structure of a discrete problem is typically highly rough. In other words, a little change in genotype may cause a significant difference in phenotype. As a result, the search process may oscillate as a result. Searching over the discrete search space is exceedingly challenging as a result of these factors. On the other hand, the fitness landscape of the modified problem can be made smoother than the original landscape using the suggested continuous encoding method. This will fix the problems with the locus and label-based encoding methods in addition to making the search simpler. This research does a fitness landscape analysis to support these claims. The research proves that the landscape of the modified continuous problem is really smoother than the one of the original community detection problem.

The goal of the attribute complex network detection problem in this paper is to identify a partition of communities that satisfies two requirements: 1) the edges between communities are sparse while those within the community are dense; and 2) the node attributes within the same community should be as similar as possible while the similarity of node attributes in different communities should be dissimilar. As a result, it is simple to represent the community detection problem as a two-objective optimization problem. This makes using MOEA to fix the issue simple and perhaps even promising.

5.9 Multi-View Community Detection in Facebook Public Pages

Due to its significance in revealing how individuals connect and interact, community detection in social networks is a topic of intense study. However, community organisation on Facebook's public pages has received little consideration. This study is looking into the issue of community detection on Facebook newsgroup sites. In particular, multi-view clustering is being used to integrate various views, such as likes on articles and likes on comments, in order to deal with the diversity of user actions.

In this study, the community structure is examined on numerous pages in addition to a single page. The outcomes demonstrate that this approach can successfully lessen isolating behaviours and enhance community structure. This study suggests modelling a Facebook page as a weighted graph produced by two views (posts and comments). The Facebook communities of CBS News and The New Times are then analysed for the final week of 2012. Additionally, the organisation of the general user community across several pages is researched. The results demonstrate that integrating diverse views can significantly minimise the number of isolates in a single-view and increase the cohesiveness of the community structure in networks since both views can complement one another.

In this study, the community structure was not only examined on a single page but also on two or even three pages. The outcomes show that the technique has three benefits: 1) It can help with the sparse network/isolates view's problem. For instance, the isolates are down from 3, 321 and 6, 647 in the final week of 2012 on the CBS News website to 4. 2) During the process, a more unified community structure can be found; for instance, on the New York Times page from the final week of 2012, there are 31 communities, down from 3, 395 and 36, 340. 3. It unearths latent communities on numerous pages. The common users on the ABC, CBS, and NBC pages exhibit high coherence in the post graph, but in this weighted graph, two additional communities are identified that could offer useful data to recommend systems.

The aforementioned technique still has two drawbacks, though. First, directed graphs cannot be handled by it. One edge should point from the user who has the like to the user who submits the comment, for instance, since like actions are one-way. It is unclear whether the community structure could change in such a circumstance. Second, it is impossible to identify communities that overlap, but this is noteworthy since it makes the notion that a user can be a member of numerous communities more logical. More crucially, in actual recommendation systems, such overlapping structure might support making more accurate product recommendations to users.

5.10 A Multiobjective Evolutionary Algorithm Based on Structural and Attribute Similarities for Community Detection in Attributed Networks

Vertex connection is the foundation of the majority of the current community detection techniques. In contrast, each vertex in most real networks typically includes one or more attributes that describe its qualities, which are frequently homogeneous in a cluster. Such networks can be represented as attributed graphs, whose attributes sometimes play an equivalent role in graph clustering as topological structure. Finding communities while simultaneously taking into account vertex attributes and topological structure is a significant problem. In order to address this, a multiobjective evolutionary algorithm (MOEA-SA) based on structural and attribute similarities is initially suggested to address the challenges of attributed graph clustering in this study. A new aim called "attribute similarity" is presented for MOEA-SA, and another objective used is "modularity." The erroneously assigned genes are corrected using a hybrid representation and a neighbourhood correction technique that balances structural and attribute information. Additionally, a powerful multi-individual-based mutation operator is created to control evolution in a positive direction. Several actual Facebook attributed graphs and numerous ego-networks with multiple attributes are used to validate MOEA-SA’s performance. Density and entropy are considered as two metrics to assess the effectiveness of the communities created. The outcomes of the experiments suggest that MOEA-SA is effective, and systematic comparisons with other approaches reveal that MOEA-SA can find more communities with practical applications and obtain better values of and in each graph. In order to aid decision-makers in making practical decisions, knee points that correlate to the best compromise options are determined.

Chapter 6

The Proposed Method

Three steps are integrated into our method for semi-supervised learning on multi-view graphs. First, we effectively merge multiple views of the graph using techniques from subspace analysis. Second, to find the most informative results, we used a multiple ranking technique to trim the graph based on its sub-components. We then use a convolutional neural network that has been modified for graph-structured input to enable classification of nodes.

6.1 Merging Subspace Representations

We first determine the graph Laplacian for each layer in an undirected multilayer graph with *M* layers $G = G_i$, $M_i = 1$, where each layer G_i has the same vertex set *V* but distinct or the same edges set Ei. The normalised graph Laplacian is defined as the product of the degree matrix Di and the adjacency matrix Wi for the *i*th view of the graph, respectively.

$$L_i = D_i^{-1/2} \; (D_i - W_i) \, D_i^{-1/2}$$

We determine the spectral embedding matrix U_i by trace minimization given the graph Laplacian L_i for each layer of the graph:

$$min_{U_i \in \mathrm{R}^{n*k}} \; tr(U_i' \, L_i \, U_i) \;\; , \;\; s.t. U_i' U_i \; = \; 1$$

The Rayleigh-Ritz theorem can be used to resolve this trace minimization problem. The first k eigenvectors, which correspond to the k smallest eigenvalues of Li, are contained in the solution U_i. The original graph's nodes are embedded in a low-dimensional spectrum domain by the spectral embedding technique.

A Grassman manifold *G(k, n)* can be thought of as a collection of linear subspaces in R^n of dimension k, where each distinct subspace corresponds to a distinct point on the manifold. The distance between the subspaces can be determined as a collection of primary angles between these subspaces. Each point on the manifold can be represented by an orthonormal matrix whose columns span the corresponding k-dimensional subspace in R^{n*k}. demonstrate how the trace minimization problem for the projection distance between the two subspaces Y_1 and Y_2 may be separated apart.

$$d^2_{proj}(Y_1, Y_2) = \sum_{i=1}^{k} \sin^2 \theta_i = k - tr(Y_1 Y_1' Y_2 Y_2')$$

where the projection distance between the target representative subspace U and the individual subspaces $U_{i\,i=1}^{\,M}$ can be calculated as:

$$\begin{aligned} d^2_{proj}(U, \{U_i\}_{i=1}^{M}) = & \sum_{i=1}^{M} d^2_{proj}(U, U_i) \\ = & \; kM - \sum_{i=1}^{M} tr(UU'U_iU_i') \end{aligned}$$

Minimization of the above equation ensures that individual subspaces are close to the final representative subspace U.

Finally, a second term is introduced to minimise the quadratic-form Laplacian (evaluated on the columns of U) in order to guarantee that the original vertex connectivity in each graph layer is preserved:

$$\min_{U \in \mathbb{R}^{n*k}} \sum_{i=1}^{M} tr(U'L_iU) + \alpha_i[kM - tr(UU'U_iU_i')],$$
$$\text{s.t. } U_i'U = 1$$

α is the regularisation parameter that balances the trade-off between the two terms in the objective function
Ignoring the constant yield terms:

$$\min_{U \in \mathbb{R}^{n*k}} tr[U'(\sum_{i=1}^{M} L_i - \sum_{i=1}^{M} \alpha_i U_i U_i')U].$$

To regularise the embedding across different views, message passing and hence preservation and collaboration of properties, the following optimization problem is solved:

$$\min_{\{\mathbf{f}_u^v, \tilde{\mathbf{f}}_u^v\}} \sum_{v \in \mathcal{V}} l(\{\mathbf{f}_u^v, \tilde{\mathbf{f}}_u^v\}_{u \in \mathcal{U}} | G^{(v)}) + \gamma \cdot [\mathcal{R}^v + \tilde{\mathcal{R}}^v].$$

Which is the MVN2VEC-REG algorithm where γ is the hyper parameter that ensures the message passing of the two networks.

The two objective functions are optimised using the multi objective Non Dominated Sorting Genetic Algorithm III (NSGA-III) to get the final sub space.

After the final optimization is achieved the Rayleigh-Ritz theorem can be used to get the final solution which is given by the first k eigenvectors of the modified Laplacian:

$$L_{mod} = \sum_{i=1}^{M} L_i - \sum_{i=1}^{M} \alpha_i U_i U_i'$$

6.2 Graph based Manifold ranking

By ranking the nodes in the manifold according to their saliency with regard to select essential nodes, model performance can be improved. The updated Laplacian obtained above can be supplied straight to the downstream graph convolutional networks. On the manifold, we apply the closed form function to rank points.

$$f^* = (I - \beta * L_{mod})^{-1} q$$

L_{mod} is the normalised Laplacian. *I* stands for the identity matrix, and *β* is the regularisation parameter.

Algorithm:

Input : $\{A_i\}$ $i = 1$ to M: n × n adjacency matrices of individual graph layers $\{G_i\}M$ $i=1$, with G_1 being the most informative layer
$\{\alpha_i\}$ $i = 1$ to M, regularisation parameters per subspace to be merged.
K: salient query points
β: manifold ranking regularizer
γ : Regularisation parameter for collaboration of multiple views.

Output: L_{mod} : Merged Laplacian,
A_{mod} : Merged Adjacency matrix,
E_s : Salient Edges,
E_{ns} : Non salient edges

Steps:

1. Compute normalised Laplacian matrix L_i for each layer of the graph.
2. Compute subspace representation U_i for each layer of the graph.
3. Compute the modified Laplacian matrix $L_{mod} = \sum_{i=1}^{M} L_i - \sum_{i=1}^{M} \alpha_i U_i U_i'$ using the two objective functions and optimising them using the NSGAIII algorithm.
4. Perform clustering on the modified Laplacian to identify K salient points i.e. centroids $\{q_i\}_{i=1}^{K}$
5. For each of the centroid rank other edges on the manifold $f^* = (\mathcal{I} - \beta * L_{mod})^{-1} q$
6. For each centroid q_i add Y salient edges to the E_s and Z non-salient edges to the E_{ns}
7. Add E_s to A_1 to form A_{mod}
8. Remove E_{ns} from A_{mod}

Chapter 7

Data Collection and preparation

Network	Network Type	#Nodes	#Edges (View 1)	#Edges (View 2)	#Classes	#Views	#Features
CoRA	Citation	2708	5429	2846	7	2	1433
CiteSeer	Citation	3312	4732	3492	6	2	3703

Table 1 : Statistics of the experimental datasets

Cora and Citeseer are two widely used citation networks, are chosen for trials for the learning task. Each node in these data sets corresponds to a piece of writing, and the citation is used as the edge. For every data set, we create two views: a text similarity network and a citation network. Citation linkages in paper citation records serve as the foundation for citation networks. Cosine similarity between publications is used to build text similarity networks.

Cora: The 2708 machine learning papers that make up the network's nodes in this dataset's citation network. The dataset contains 2 views, 7 different types of papers, and 5429 edges.

Citeseer: There are 4732 edges in all across 3312 publications in the data set. There are 2 views and 6 categories on this network.

For each set of data, we employ a nonlinear activation function (RELU), and 200 training cycles are used. We change different parameters based on various data sets.

Chapter 8

Experimental results and discussion

In a network where only a relatively tiny percentage of nodes are labelled, our overall objective is to correctly classify nodes.In the tests, we begin with a small sample of labelled nodes and test Multi-prowess GCN's at correctly classifying unlabeled nodes in the validation and testing sets, as well as that of many other cutting-edge algorithms. As a first set of benchmarks, we use the three well-known node embedding techniques Node2vec, Deepwalk, and LINE. We have also used normal GCN for result comparison.
We use accuracy and F1-score to measure the performance of the model.

Tools Used:

1. **Tensorflow :**

TensorFlow is a deep learning library. TensorFlow allows us to perform specific machine learning number-crunching operations like derivatives on huge matrices with large efficiency. We can also easily distribute this processing across our CPU cores, GPU cores, or even multiple devices like multiple GPUs. We can even distribute computations across a distributed network of computers with TensorFlow. So, while TensorFlow is mainly being used with machine learning right now, it actually stands to have uses in other fields, since really it is just a massive array manipulation library.

2. **Scikit Learn:**

Scikit-learn is a free software machine learning library for the Python programming language. Simple and efficient tools for predictive data analysis and built on NumPy, SciPy, and matplotlib. We use for data preprocessing like feature extraction, normalisation and we use for machine learning algorithm for classification and improving parameter via parameter tuning and comparing models with metric function.

3. **Spektral:**

It is built on top of keras and tensorflow api and is specifically meant to perform deep learning on graph data structures. Having useful classes like Dataset helps in handling Graph Data.

4. **Networkx:**

It is a utility that is built on top of matplotlib and allows us to visualise Graph Data Structures.

Dataset :
The citation datasets are present in the spektral.datasets.citation module. We have extended that class to create two separate views for the citation data.

Procedure:
The modified graph A_{mod} that is produced when Laplacians are combined using subspace analysis, message passing through preservation and collaboration, and manifold ranking can be directly input into the previously mentioned graph convolution networks. The following is a representation of the forward propagation model for a two layer network:

$$Z = f(X, A) = \mathrm{softmax}(\hat{A}\ \mathrm{ReLU}\ (\hat{A}XW^{(0)})\ W^{(1)}\ )$$

Here, $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is calculated as a preprocessing step before giving the input to the neural network. W_0 and W_1 represent the input-to-hidden-layer and hidden-layer-to output weight matrices for a two layer neural network, and can be trained using gradient descent. ReLU and Softmax represent the activation functions in the hidden and output layers.

Table 2 : Mean Node classification accuracy in percentage and standard error over 10 randomly selected dataset splits of equal size

Methods	Cora	Citeseer
DeepWalk [22]	70.7 ± 0.6	51.4 ± 0.5
Node2Vec [12]	69.11 ± 0.01	50.15 ± 0.02
LINE [63]	65.23 ± 0.13	51.12 ± 0.5
GCN [26]	81.7 ± 0.5	72.5 ± 0.7
Our Method	83.0 ± 0.7	73.0 ± 0.5

Table 3 : Mean F1 measure over 10 randomly selected dataset splits of equal size

Methods	**Cora**	**Citeseer**
DeepWalk [22]	0.581	0.45
Node2Vec [12]	0.47	0.250
LINE [63]	0.42	0.247
GCN [26]	0.873	0.768
Our Method	0.883	0.772

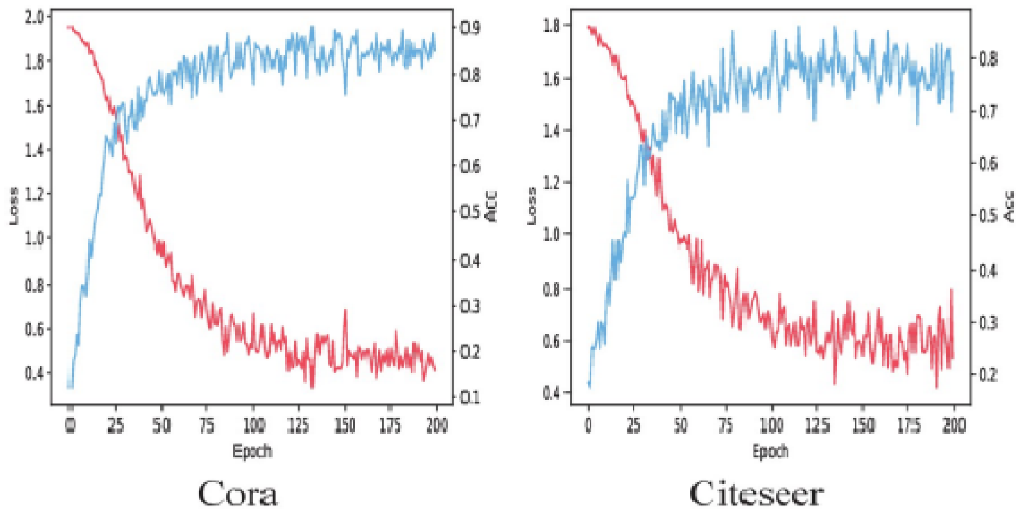


Fig 32: Iteration influence on loss (red) and accuracy (blue)

Chapter 9

Conclusion and Scope for Future Works

Representation learning for graph data has achieved considerable success in fields of machine learning. With the advent of social media data, relational data, the ability to learn patterns from network data is very necessary nowadays. In many cases network data might consists of multiple views of the same node representation. In our project we have tried to achieve the feat of learning patterns in a multiview network with the aid of Graph Convolutional Networks. Hence we are able to perform node level classification with the help of our proposed model with a better accuracy than the benchmark tests. Hence we will also be able to detect communities within the network based on node labelling. In the project we have done the tests on the benchmark citation networks and have successfully performed node labelling and community detection with significant accuracy.

Future scope of our project involves link prediction and subgraph prediction. Also we would like to extend our training with different multiview datasets like Twitter, Youtube, Snapchat. In the project the regularisation and message passing parameters are taken to be fixed. We would like to make them adaptive in the future and include them as part of optimization.

Bibliography

[1] Graph neural networks: A review of methods and applications Jie Zhou , Ganqu Cui , Shengding Hu , Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun (https://arxiv.org/pdf/1812.08434.pdf)

[2] https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

[3]https://towardsdatascience.com/a-gentle-introduction-to-graph-neural-network-basics-deepwalk-and-graphsage-db5d540d50b3

[4] https://distill.pub/2021/gnn-intro/

[5] Understanding Convolutions on Graphs. Daigavane, A., Ravindran, B. and Aggarwal, G., 2021. Distill. DOI: 10.23915/distill.00032

[6] The Graph Neural Network Model. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M. and Monfardini, G., 2009. IEEE Transactions on Neural Networks, Vol 20(1), pp. 61--80.

[7] A Deep Learning Approach to Antibiotic Discovery. Stokes, J.M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N.M., MacNair, C.R., French, S., Carfrae, L.A., Bloom-Ackermann, Z., Tran, V.M., Chiappino-Pepe, A., Badran, A.H., Andrews, I.W., Chory, E.J., Church, G.M., Brown, E.D., Jaakkola, T.S., Barzilay, R. and Collins, J.J., 2020. Cell, Vol 181(2), pp. 475--483.

[8] Learning to simulate complex physics with graph networks
Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J. and Battaglia, P.W., 2020.

[9] Fake News Detection on Social Media using Geometric Deep Learning
Monti, F., Frasca, F., Eynard, D., Mannion, D. and Bronstein, M.M., 2019.

[10] Traffic prediction with advanced Graph Neural Networks, O.L. and Perez, L.

[11] Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in {Real-Time}
Eksombatchai, C., Jindal, P., Liu, J.Z., Liu, Y., Sharma, R., Sugnet, C., Ulrich, M. and Leskovec, J., 2017

[12] node2vec: Scalable Feature Learning for Networks(https://arxiv.org/pdf/1607.00653.pdf)

[13] S. Doshi. Skip-Gram: NLP context words prediction algorithm

[14] Perozzi et Al. DeepWalk: Online Learning of Social Representations (https://arxiv.org/pdf/1403.6652.pdf)

[15]https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c?gi=8ff2aeada829

[16] Efficient Estimation of Word Representations in Vector Space. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean (https://arxiv.org/pdf/1301.3781.pdf)

[17] Understanding Graph Embedding methods and their applications. Mengia Xu (https://arxiv.org/pdf/2012.08019.pdf)

[18] Graph Machine Learning: Take Graph Data to the Next Level by Applying Machine Learning Techniques and Algorithms by Aldo Marzullo, Claudio Stamile, and Enrico Deusebio

[19] https://towardsdatascience.com/word2vec-explained-49c52b4ccb71[word2vec]

[20]https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

[21] https://towardsdatascience.com/deepwalk-its-behavior-and-how-to-implement-it-b5aac0290a15

[22] B. Perozzi, R. Al-Rfou, and S. Skiena, Deepwalk: Online learning of social representations, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[23] An attention-based Collaboration Framework for Multi-View Network Representation Learning Meng, Jian Tang , Jingbo Shang, Xiang Ren, Ming Zhang4, Jiawei Han

[24] A Comprehensive Survey on Graph Neural Networks Zonghan Wu, Shirui Pan, Member, IEEE, Fengwen Chen, Guodong Long, Chengqi Zhang, Senior Member, IEEE, Philip S. Yu, Fellow, IEEE(https://arxiv.org/pdf/1901.00596.pdf)

[25] Graph convolutional networks: a comprehensive review Si Zhang, Hanghang Tong, Jiejun Xu & Ross Maciejewski

[26] Semi-Supervised Classification with Graph Convolutional Networks Thomas N., Max Welling (https://arxiv.org/pdf/1609.02907v4.pdf)

[27] https://www.topbots.com/graph-convolutional-networks/

[28] Video Graph Convolutional Networks (GCNs) made simple: https://www.youtube.com/watch?v=2KRAOZIULzw

[29] Excellent slides on Graph Representation Learning by Jure Leskovec (Stanford): https://drive.google.com/file/d/1By3udbOt10moIcSEgUQ0TR9twQX9Aq0G/view?usp=sharing

[30] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," nature, vol. 393, no. 6684, p. 440, 1998.

[31] M. E. J. Newman, "The structure of scientific collaboration networks," Proceedings of the National Academy of Sciences, vol. 98, no. 2, pp. 404–409, 2001.

[32] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," Proceedings of the National Academy of Sciences, vol. 99, no. 12, pp. 7821–7826, 2002

[33] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Phys. Rev. E, vol. 76, p. 036106, Sep 2007.

[34] S. Fortunato, "Community detection in graphs," Physics Reports, vol. 486, no. 3, pp. 75–174, 2010.

[35] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in Proceedings of the eleventh annual conference on Computational learning theory, pp. 92–100

[36] ACM, 1998. [7] L. Tang, X. Wang, and H. Liu, "Uncovering groups via heterogeneous interaction analysis," in Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, pp. 503–512, IEEE, 2009.

[37] Overview of NSGA-II for Optimizing Machining Process Parameters. Yusliza Yusof, Mohd Salihin Ngadiman, Azlan Mohd Zain

[38] K. Deb, A. Pratap, S. Agarwal and T. A. M. T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 6 (2002), 182-197

[40] K. Deb and J. Himanshu, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints, IEEE Transactions Evolutionary Computation, 18 (2014), 577-601.

[40] D. W. Corne, N. R. Jerram, J. D. Knowles and M. J. Oates, PESA-II: Region-based selection in evolutionary multi-objective optimization, Genetic and Evolutionary Computation Conference, (2001), 283–290.

[41] D. W. Corne, J. D. Knowles and M. J. Oates, The pareto envelope-based selection algorithm for multiobjective optimization, Parallel Problem Solving from Nature PPSN VI. Springer, (2000), 839–848.

[42] A. Cheng and L. Cheng-Chew, Optimizing System-On-Chip verifications with multi-objective genetic evolutionary algorithms, Journal of Industrial and Management Optimization, 10 (2014), 383-396. doi: 10.3934/jimo.2014.10.383.

[43] H. B. Fang, Q. Wang, Y. C. Tu and M. F. Horstemeyer, An efficient non-dominated sorting method for evolutionary algorithms, Evolutionary Computation, 16 (2008), 355-384.

[44] M. Drozdik, A. Youhei, A. Hernan and T. Kiyoshi, Computational cost reduction of nondominated sorting using the M-front, IEEE Transactions on Evolutionary Computation, 19 (2015), 659-678.

[45] A novel non-dominated sorting algorithm for evolutionary multi-objective optimization. Chunteng Bao, Lihong Xu, Erik D.Goodman, Leilei Cao

[46] Holland J H GDE. Genetic Algorithms in Search, Optimization and Machine Learning[J]. 1989.

[47] An Improved Nondominated Sorting Genetic Algorithm for Multiobjective Problem. Ruihua Wang (https://doi.org/10.1155/2016/1519542)

[48] Chapter 5 - Multiobjective Optimization and Advanced Topics(Chang, K.-H. (2015). Multiobjective Optimization and Advanced Topics. Design Theory and Methods Using CAD/CAE, 325–406. doi:10.1016/b978-0-12-398512-5.00005-0 (https://doi.org/10.1016/B978-0-12-398512-5.00005-0)

[49] NSGA-II explained! By Paul Calle | October 24, 2017 (http://oklahomaanalytics.com/data-science-techniques/nsga-ii-explained/)

[50] Introduction to Genetic Algorithms (https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3)

[51] Z. Li, J. Liu, and K. Wu, “A multiobjective evolutionary algorithm based on structural and attribute similarities for community detection in attributed networks,” IEEE Transactions on Cybernetics, vol. 48, no. 7, pp. 1963–1976, July 2018.

[52] Multi-view Knowledge Graph Embedding for Entity Alignment Qingheng Zhang, Zequn Sun, Wei Hu , Muhao Chen , Lingbing Guo , Yuzhong Qu

[53] Multi-View Network Representation Learning Algorithm Research Zhonglin Ye, Haixing Zhao, Ke Zhang and Yu Zhu

[54] 9th International Young Scientist Conference on Computational Science (YSC 2020) Community detection in node-attributed social networks: How structure-attributes correlation affects clustering quality. Petr Chunaev, Timofey Gradov and Klavdiya Bochenina Petr Chunaev, Timofey Gradov and Klavdiya Bochenina. National Center for Cognitive Technologies, ITMO University, Saint Petersburg, 199034 Russia

[55] Graph Neural Network Encoding for Community Detection in Attribute Networks. Jianyong Sun Senior Member, IEEE, Wei Zheng, Qingfu Zhang Fellow, IEEE and Zongben Xu

[56] MGAT: Multi-view Graph Attention Networks. Yu Xie, Yuanqiao Zhang, Maoguo Gong, Zedong Tang, Chao Han

[57] A Multi-objective Genetic Algorithm for Community Detection in Networks. Clara Pizzuti Conference Paper · November 2009 DOI: 10.1109/ICTAI.2009.58 · Source: DBLP

[58] MVN2VEC: Preservation and Collaboration in Multi-View Network Embedding Yu Shi, Member, IEEE, Fangqiu Han, Xinwei He, Xinran He, Carl Yang, Member, IEEE, Roger Jie Luo, and Jiawei Han, Fellow, IEEE

[59] Multi-View Collaborative Network Embedding SEZIN KIRCALI ATA, Nanyang Technological University, Singapore YUAN FANG, Singapore Management University, Singapore, MIN WU, Institute for Infocomm Research, Singapore, JIAQI SHI, Singapore Management University, CHEE KEONG KWOH, Nanyang Technological University, XIAOLI LI, Institute for Infocomm Research and Nanyang Technological University

[60] Multi-View Community Detection in Facebook Public Pages. Zhige Xin, Chun-Ming Lai, Jon W. Chapman, George Barnett, S. Felix Wu

[61] https://pymoo.org/algorithms/moo/nsga3.html

[62] A NSGA-II and NSGA-III comparison for solving an open shop scheduling problem with resource constraints. Guillermo Campos Ciro, Frederic Dugardin, Farouk Yalaoui, Russell Kelly

[63] LINE: Large-scale Information Network Embedding Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei