# JADAVPUR UNIVERSITY

MASTER DEGREE THESIS

---

# Classification of Multi-View Network Data using Quantum Walk Neural Network

---

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

*in*

Computer Science and Engineering

*by*

Subham Dawn
Exam Roll No.: M4CSE22041
Class Roll No.: 002010502041
Registration No.: 154165 of 2020-2021

*Under the Guidance of*

Dr. Debotosh Bhattacharjee
Department of Computer Science and Engineering
Jadavpur University

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

August, 2022

# Declaration of Authorship

I, Subham Dawn, declare that this thesis titled, “Classification of Multi-View Network Data using Quantum Walk Neural 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 “Classification of Multi-View Network Data using Quantum Walk Neural Network” is a bona-fide record of work carried out by Subham Dawn, Examination Roll No.: M4CSE22041, University Registration No.: 154165 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.

_____

Dr. Debotosh Bhattacharjee
Department of Computer Science and Engineering
Jadavpur University
Kolkata - 700032

_____

Dr. Nandini Mukherjee
Head of the Department
Department of Computer Science and Engineering
Jadavpur University
Kolkata - 700032

_____

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

# Certificate of Approval

(Only in case the thesis is approved)

The thesis 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 classification of Multi-view network data with the help of Quantum Walk inspired Neural Network

Approach by Subham Dawn

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 propose a graph-based Quantum walk inspired neural network for learning on multi-view networks by introducing preservation and collaboration parameters in subspace merging 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 Dr. Debotosh Bhattacharjee, 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 Dr. Nandini Mukherjee, Head of the Department of Computer Science and Engineering, Jadavpur University and Dr. 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,
Subham Dawn
Exam Roll No.: M4CSE22041
University Registration No.: 154165 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 |
| **QWNN** | Quantum Walk Neural Network |
| **GNN** | Graph Neural Network |
| **QRAM** | Quantum Random Access Memory |
| **CNN** | Convolutional Neural Network |
| **RNN** | Recurrent Neural Network |
| **MLP** | Multilayer Perceptron |
| **NLP** | Natural Language Processing |
| **W2V** | Word2Vec |
| **STGNN** | Spatial Temporal Graph Neural Network |
| **QWNN** | Quantum Walk Neural Network |
| **MOEA** | Multi Objective Evolutionary 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 Classification Problem for Graph Networks

A network is considered to have a community structure in the study of complex networks if its nodes can be quickly categorized into sets of nodes, with each set being densely connected to the others. In terms of graphs, a community is a subset of nodes closely related 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, we eventually connect with others from various social circles. These social circles could include 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 regularly. We might frequently communicate 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 analyzing them. Using community detection techniques, social media algorithms can find people who share interests and keep them connected. Machine learning may utilize community detection to find groups with similar characteristics and extract groupings for various 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. The machine learning clustering technique groups related data points based on shared characteristics. Clustering is a more general area of unsupervised machine learning that deals with various attribute types, even though it can be applied to networks. However, community detection was created specifically for network analysis 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 due to clustering techniques. However, different advantages and disadvantages of using clustering and community detection algorithms may arise depending on the problem and the area.

## 1.1 Node Classification in Multi-view Networks

A graph network can be visualized 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 and 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 complex networks. Since 2005, surveys of community detection in graphs and networks have been published regularly. 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. Similarly, Facebook users connect by liking, commenting, 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 2017, and up to 40 million small companies had public pages. It has been discovered that there is a large correlating increase in the difficulty of finding community structures within Facebook pages with increased 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.

## 1.2 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 proximity interaction 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 a single view.

Multi-view networks exist around us in almost every field. May it be Twitter, Facebook, LinkedIn, or any other real-time networks, all of which show some multi-view properties, and community detection in these kinds of networks is an area that still needs to be explored. It is necessary to devise an effective solution to this multiview network community detection problem.

## 1.3 Aim and Objective

This thesis aims to develop a technique to learn the multi-view network pattern efficiently. This requires processing and developing datasets consisting of multi-view graph networks and representing the network in the different views of the graph. To achieve this goal, a number of key points need to be achieved :

- Using subspace analysis to merge multiple views of the graph efficiently.
- Find the most informative subcomponents of the graph
- Using multi-objective optimization to adjust the preservation and collaboration of the information in different views.
- Quantum walk Neural Net generates the diffusion matrix, learns the data, and effectively makes predictions and class labeling.

## 1.4 Contribution

Learning multi-view graph network data has always been challenging for conflicting view types. We have tried to propose an efficient learning mechanism for multi-view graph data in the project, making the model capable of node classification and prediction. We have used the methods from subspace analysis to 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 optimizing the preservation and collaboration factor using NSGA III. Finally, we apply a neural network based on Quantum walk, adapted to graph-structured data, to allow for supervised or semi-supervised node classification.

## 1.5 Thesis Organisation

The rest of the thesis is organized as follows: Chapter 2 describes the concept of Quantum Computing and deep learning, focusing mainly on the theory of Graph Neural networks, 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 4 Contains a brief description and the mathematical representation of Multi-view information networks. Chapter 4 briefly discusses the theory of optimization techniques focusing mainly on Multi-Objective Optimisation and Genetic Algorithms for optimization. Chapter 5 consists of a literature survey related to Quantum Networks. Our proposed method is discussed in Chapter 6, and the data collection and preparation process is discussed in Chapter 7. The experimental results are analyzed in Chapter 8. Finally, we conclude and put some light on future work in Chapter 9.

# Chapter 2

# An Introduction to Quantum Computing

Quantum computing is a fast-developing technology that uses the principles of quantum physics to address issues that are too complicated for classical computers. The field of computing known as quantum computing is devoted to creating computer technology based on the ideas of quantum theory (which explains the behavior of energy and material on the atomic and subatomic levels). Today's computers can only encode data in bits with a value of 1 or 0, limiting their capacity. On the other hand, quantum computing uses quantum bits or qubits. It takes advantage of subatomic particles' remarkable capacity to existing in several states (i.e., a 1 and a 0 simultaneously).
The world might transform thanks to quantum computing. It can revolutionize artificial intelligence, health, and communications while breaking encryption.

## 2.1 Basics of Quantum Computing

Bits are used in a typical computer chip. These resemble small switches that can only be in one of two states: off, denoted by a zero, or on, denoted by a one. Millions of these bits in various combinations of ones and zeroes make up every program we use, website we visit, and photo we snap.

Although it works well for most situations, this is not how the universe truly operates. Things aren't only on or off in nature. They are unsure. Even the finest of our supercomputers struggle to deal with ambiguity. That is a difficulty.

That's because physicists have learned over the past century that strange things start to happen when you scale down to small objects. They have created an entirely new branch of science to try to understand them. It is known as quantum mechanics. Physics, chemistry, and biology are all based on quantum mechanics, which is the basis of physics. Therefore, scientists need a better method of performing computations that can handle uncertainty if they mimic any of those things properly along come quantum computers.

Superposition and entanglement are two features of quantum physics based on these supercomputers. This empowers quantum computers to handle operations at speeds exponentially higher than conventional computers and much lesser energy consumption.

The field of quantum computing started in the 1980s. It was then discovered that certain computational problems could be tackled more efficiently with quantum algorithms than with their classical counterparts. Finance, military affairs, intelligence, drug research, aircraft design, utilities (nuclear fusion), polymer design, machine learning, artificial intelligence (AI), big data search, and digital manufacturing might benefit tremendously from quantum computing. Due to its potential and anticipated market size, some of the most well-known technology companies, including IBM, Microsoft, Google, D-Waves Systems, Alibaba, Nokia, Intel, Airbus, HP, Toshiba, Mitsubishi, SK Telecom, NEC, Raytheon, Lockheed Martin, Rigetti, Biogen, Volkswagen, and Amgen, are working in the field of quantum computing.

### 2.1.1 Quantum Computer v/s Classical Computer

Information is processed in quantum computers differently. Traditional computers employ transistors, which can only be either 1 or 0. Qubits, which may be 1 or 0 simultaneously, are used in quantum computers. The quantum computer capacity grows exponentially as more qubits are connected. In contrast, connecting additional transistors together just linearly boosts power.

Traditional computers work best for routine operations that a computer must carry out. Quantum computers, on the other hand, are fantastic at doing simulations and data analysis, like those used in medicine or chemical studies. But you must keep these computers at absolute zero. They are also far more expensive and challenging to construct.

Memory expansion is a traditional method of accelerating computers. Quantum computers, meanwhile, assist in resolving more challenging issues. Quantum computers can solve complicated problems more quickly, even though they might not perform better or quicker than Microsoft Word.

For instance, Google's quantum computer, which is now under development, may be able to speed up machine-learning training or contribute to the creation of batteries that use less energy. There are several more uses for quantum computing, such as safely transferring data. Fighting cancer and other health issues, such as generating new medications, are different strategies. Quantum computers can also aid in enhancing radars' capacity to identify objects like missiles

and airplanes. Other fields include the environment, chemical sensors, and quantum computing to keep the water pure.
Unlike supercomputers and conventional computers, quantum computers are several times quicker. Sycamore, Google's quantum computer, is alleged to have completed a computation in 200 seconds that would have taken IBM's Summit, one of the fastest computers in the world, 10,000 years to complete. The time required, according to IBM, would be 2.5 days, which is still more than 1,000 times slower than Google's quantum computer.

## 2.1.2 Working of a Quantum Computer

Quantum computers employ qubits as opposed to bits. Qubits can be in "superposition," which is when they are both on and off at the same time or anywhere along a spectrum between the two, as opposed to merely being on or off. If we flip a coin, it will come out heads or tails. However, if we spin it, there is a probability that it will rest on either heads or tails. It can be either until we quantify it by halting the coin. One aspect of superposition, which resembles a spinning coin, is what gives quantum computers their incredible capability. A qubit allows uncertainty.
If we give a typical computer a maze to escape, it will test each branch in turn and eliminate each one until it finds the best path. A quantum computer can traverse the entire maze simultaneously. It can contain uncertainty. It's similar to turning the pages of a choose your adventure novel constantly. Instead of going back to the book's beginning if our character dies, we may pick a new route right away.
Entanglement is the other function that qubits are capable of. In a normal coin toss, the outcome of one coin does not influence the outcome of the other. They are autonomous. Even if two particles are physically distinct, they are connected through entanglement. Both bets will result in heads if one is head.
Physics experts still don't completely get how or why it operates, and it sounds like magic. However, it means that we can transport information around the world of quantum computing even if it involves uncertainty. That rotating coin may be used to conduct intricate computations. We can solve issues that would take our finest computers millions of years to solve if we can connect many qubits.

So far, it has been established that quantum computers are sophisticated devices that are smaller and use less energy than supercomputers. A laptop's wafer is comparable in size to an IBM Quantum processor. A quantum hardware system, which is roughly the size of a vehicle and consists primarily of cooling devices to maintain the superconducting processor at its extremely

low working temperature is also of this size. A traditional processor uses bits to carry out its functions. Qubits (CUE-bits) are used by a quantum computer to execute multidimensional quantum algorithms. The detailed working of quantum computers can be discussed as follows:

**Superfluids:** It's conceivable that our desktop computer utilizes a fan to become chilly enough to operate. However, quantum processors require very low temperatures—just one-tenth of a degree above absolute zero. Superconductors are made using super-cooled superfluids to accomplish this.

**Superconductors:** Certain components in our processors demonstrate another crucial quantum mechanical property at extremely low temperatures: electrons pass through them without resistance. They are "superconductors" as a result. Cooper pairs are formed when electrons move through superconductors. Through a process known as quantum tunneling, these pairs can transport a charge beyond insulators or barriers. A Josephson junction is created when two superconductors are positioned on opposite sides of an insulator.

**Control:** Josephson junctions serve as superconducting qubits in quantum computers. Microwave photons may influence these qubits' behavior, allowing them to store, modify, and read out discrete quantum bits of information.

**Superposition:** A qubit isn't particularly helpful by itself. It can, however, pull off a crucial ruse by putting the quantum data it contains into a state of superposition that combines all qubit configurations that could be feasible. The superposition of qubit groups can provide intricate, multidimensional computational landscapes. In these settings, complex issues may be expressed in novel ways.

**Entanglement:** A quantum mechanical phenomenon known as entanglement correlates the actions of two distinct entities. Changes to one qubit immediately affect the other when two qubits are entangled. Quantum algorithms make use of such connections to solve challenging issues.

## 2.1.3 Abilities of a Quantum Computer

It's not simply about working quicker or more effectively with quantum computers. We'll be able to do things with their help that we never even imagined. Things that not even the finest supercomputer can do. They might significantly hasten the advancement of artificial intelligence. Google is already using them to enhance the software of self-driving cars. They are essential for modeling chemical processes as well.

Supercomputers can only now analyze the most fundamental substances. The quantum characteristics that quantum computers use to function are identical to those of the molecules they seek to imitate. They ought to have no trouble managing even the trickiest reactions. This might result in more productive goods, such as significantly enhanced solar panels or novel battery materials for electric automobiles. Using quantum simulations by researchers might lead to the development of an Alzheimer's cure.

Anywhere there is a vast, complex, unpredictable system that has to be mimicked, quantum computers will be useful. Anything from forecasting the financial markets to enhancing weather predictions to simulating the behavior of individual electrons might fall under this category: understanding quantum physics using quantum computation anywhere there is a vast, complex, unpredictable system that has to be mimicked, quantum computers will be useful. Anything from forecasting the financial markets to enhancing weather predictions to simulating the behavior of individual electrons might fall under this category: understanding quantum physics using quantum computation.

The application of cryptography will also be important. Many encryption techniques now rely on how hard it is to divide big integers into prime numbers. This process, known as factoring, is cumbersome, costly, and unfeasible for older systems. But it can be done quickly by quantum computers. And that may jeopardize our data.
In anticipation of eventually having access to a quantum computer that can decrypt information, there are rumors that spy agencies worldwide are already collecting enormous volumes of encrypted data. The sole means of defense is quantum encryption. This is based on the uncertainty principle, which holds that it is impossible to measure anything without affecting the outcome. Keys for quantum encryption could not be duplicated or compromised. They would be indestructible.

## 2.2 Quantum Machine Learning

In recent years, quantum computing has grown quickly in both theories and practice, raising hopes that it may influence actual applications. How quantum computers could affect machine learning is one important topic of investigation. It's still very early for the theoretical science of quantum machine learning to mature. It sits at the nexus of machine learning and quantum computing. Quantum Machine Learning's major objective is to accelerate processes by integrating what we know about quantum computing with machine learning. Aspects of classical machine learning theory are included in the quantum machine learning theory, which approaches quantum computing through that framework.

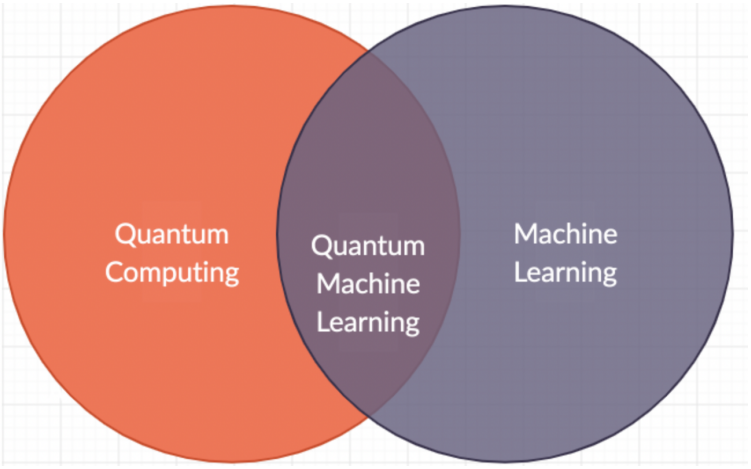


**Fig 1: Quantum Machine Learning: Quantum Computing and Machine Learning intersection.**

The hardware we use to run our algorithms has always dictated the bounds of what computers can learn; for instance, parallel GPU clusters enable the success of contemporary deep learning using neural networks. The quantum computer adds a completely new category of computing hardware to the machine learning hardware pool: quantum computers. The quantum theory underlies the fundamentally distinct physical principles that underpin information processing on quantum computers. Some research focuses on future, years away, fault-tolerant quantum

processors (or "fault-tolerant QPUs"). However, interest in quantum machine learning for upcoming quantum devices is developing quickly. These components can be considered specialized hardware with more constrained functionality than Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs).

Quantum computers may be utilized and trained similarly to neural networks from a contemporary perspective. To address an issue, we can methodically modify the physical control parameters, such as the intensity of an electromagnetic field or the frequency of a laser pulse. By measuring the physical state of the device and encoding the picture into it, a trained circuit, for instance, may be used to categorize the content of photographs. But the story is bigger than just using quantum computers to tackle machine learning problems. Quantum circuits are differentiable, and a quantum computer itself can compute the change in control parameters needed to become better at a given task.

Differentiable programming is the basis of deep learning, implemented in software libraries such as TensorFlow and PyTorch. Differentiable programming is more than deep learning: it is a programming paradigm where the algorithms are not hand-coded but learned. Similar to quantum machine learning, the concept of training quantum computers is more expansive. Other disciplines, like quantum chemistry or quantum optimization, can benefit from trainable quantum circuits. Applications include the creation of quantum algorithms, the identification of quantum error correction techniques, and the comprehension of physical systems.

Now let us go through some in-depth basics of quantum computing and then to quantum machine learning.

## 2.2.1 Bra-ket Notation

Equations are written using the "Bra-ket" or "Dirac" notation in quantum mechanics and quantum physics. Because of this, readers (particularly beginners) should be aware of it while reading research articles on quantum computing.

Angle brackets, 〈 〉, and a vertical bar, |, are used in the notation to create "bras" and "kets."

A “ket” looks like this: |v〉. Mathematically it denotes a vector, v, in a complex vector space V. Physically, it represents the state of a quantum system.

A “bra” looks like this: ⟨f|. Mathematically, it denotes a linear function f: V → C, i.e., a linear map that maps each vector in V to a number in the complex plane C.

Letting a linear function ⟨f| act on a vector |v⟩ is written as ⟨f | v⟩ $\in$ C

The Bra-ket notation may be used to describe wave functions and other quantum states as vectors in a complex state. This notation can also be used to represent quantum superposition. Other uses include measurements related to linear operators and wave function normalization.

## 2.2.2 The Concept of “Qubits” and the Superposition States

The "qubits" utilized in quantum computing differ from the "bits" used in traditional computers. A bit is a binary digit, the fundamental unit of classical computing. Quantum Binary Digit is what is meant by the word "qubit." Qubits can exist in numerous states simultaneously, unlike bits, which can only exist in two states—0 and 1. There is a value range of 0 to 1.
If we use the coin toss example, this idea will be easier to comprehend. One side of a coin is Heads, and the other is Tails (0). We don't know which side the coin is on when it's being tossed until we stop it or it hits the ground. Take a look at the coin toss below. Can we identify its side? Depending on your point of view, it displays either 0 or 1; only when we pause it to look does it display just one side. Similar circumstances apply to qubits. The superposition of two states is what is meant by this. Accordingly, the likelihood of measuring a 0 or 1 is often neither 0.0 nor 1.0. In other words, there is a chance that the quibit is in more than one state simultaneously. The superposition collapses when we learn the outcome of the coin flip (heads or tails).

## 2.2.3 The Bloch Sphere

A qubit is represented mathematically by the Bloch Sphere. It uses a two-dimensional vector with a typical length of one to represent the state of a qubit. Two numbers make up this vector: a real number α and a complex number β

A qubit is defined as a superposition of two states and may be identified by the following definition, which is illustrated in the below image:

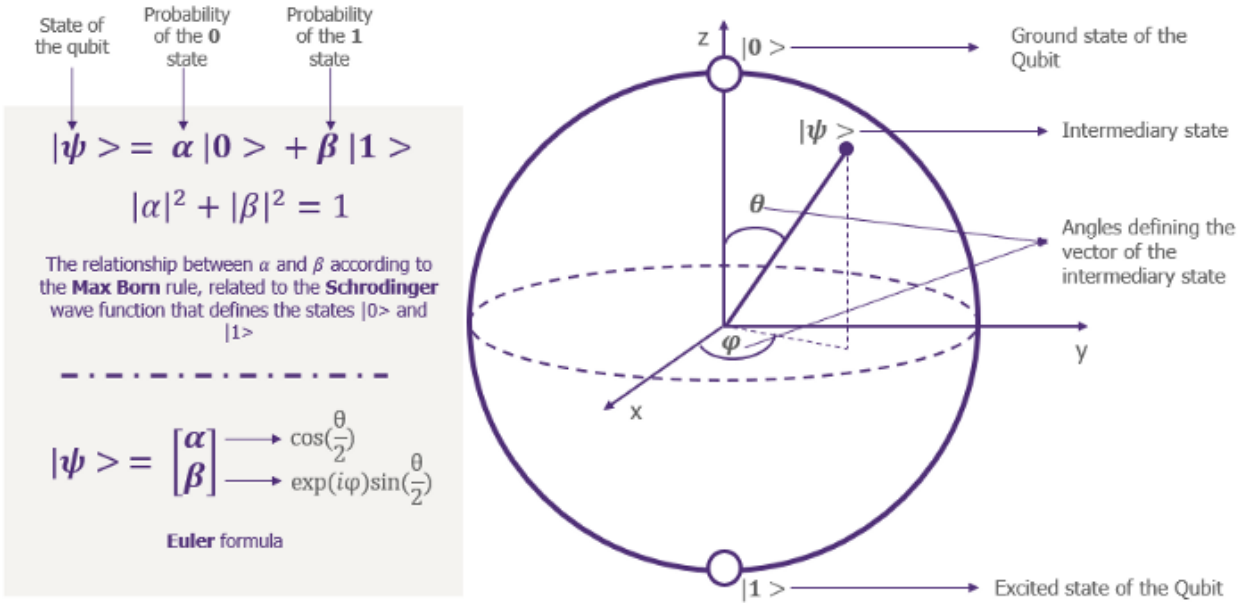


**Fig 2: The Bloch Sphere**

The Bloch sphere, which is used to depict various polarisation types in optics, is sometimes referred to as the Poincaré sphere for historical reasons. There are six typical polarisation types, which are referred to as Jones vectors. At the end of the 19th century, Henri Poincaré was the one who originally proposed using this type of geometrical representation to describe Stokes parameters in three dimensions.

## 2.2.4 Quantum Decoherence

Qubit superposition leads to problems like Quantum Decoherence. These unwelcome collapses occur erratically and organically as a result of the system disturbance. This ultimately results in calculation mistakes. When we operate on a qubit that you believe is in a superposition but isn't, the result won't be what you would have anticipated. This is why we repeatedly execute the same program, like training a machine learning model.

Since interaction with the environment leads to quantum decoherence, quantum systems must be kept apart from it. Qubits are cooled almost to zero degrees. Information from the environment seeps into qubits as they interact with the environment, and information from the qubits themselves leaks out. The information that seeps in is random noise, but the information that leaks out is most likely required for a current or future calculation. This idea is precisely like the Second Law of Thermodynamics, which states, "The total entropy of an isolated system can never decrease over time, and is constant if and only if all processes are reversible. Isolated systems spontaneously evolve towards thermodynamic equilibrium, the state with maximum entropy."

Thus, coherence is a necessary condition for quantum systems. When compared to larger items like a book or a table, such as a particle, quantum decoherence is more apparent in smaller particles. Every substance has a certain wavelength linked with it; however, the smaller the item's wavelength, the more material it contains.

## 2.2.5 Quantum Entanglement

The concept of quantum entanglement asserts that two qubits are constantly superposed in two different states. Here is one instance. Imagine there is a package with a pair of gloves inside of it. One glove is selected at random from the box. The package is then brought to another room. We may infer that the glove still in the box is left-handed because the glove taken out was discovered to be right-handed.
With qubits, the situation is the same. If one is spinning up, the other is put into a spin-down position by default. A situation in which both qubits are in the identical state is not possible. In other words, they are continuously intertwined. Quantum entanglement is what is happening here.

## 2.2.6 Dual Principle

Qubits have characteristics that are similar to both waves and particles. All items do, but atomic-scale objects like the qubit make them easier to see. Due to the wave-particle duality, qubits may communicate with one another through interference.

### 2.2.7 Quantum Speedup

The quantum computer can process information in a way that is impossible for classical computers, thanks to quantum coherence. A quantum algorithm uses a step-by-step process to address issues like database searches. It can perform better than the most popular classical algorithms. The Quantum Speedup is the name given to this occurrence.

## 2.3 Quantum Walk Neural Network

Numerous graph neural networks transmit data between two nodes based on their distance. Both diffusion and graph convolution networks have this characteristic. The quantity of information transferred between two nodes in quantum walk neural networks, like graph attention networks, also depends on the nodes' properties. This is accomplished in graph attention networks by computing an attention coefficient for each node's neighbors. In quantum walk neural networks, the coin operator modifies the quantum walk's spin states to give particular neighbors priority.

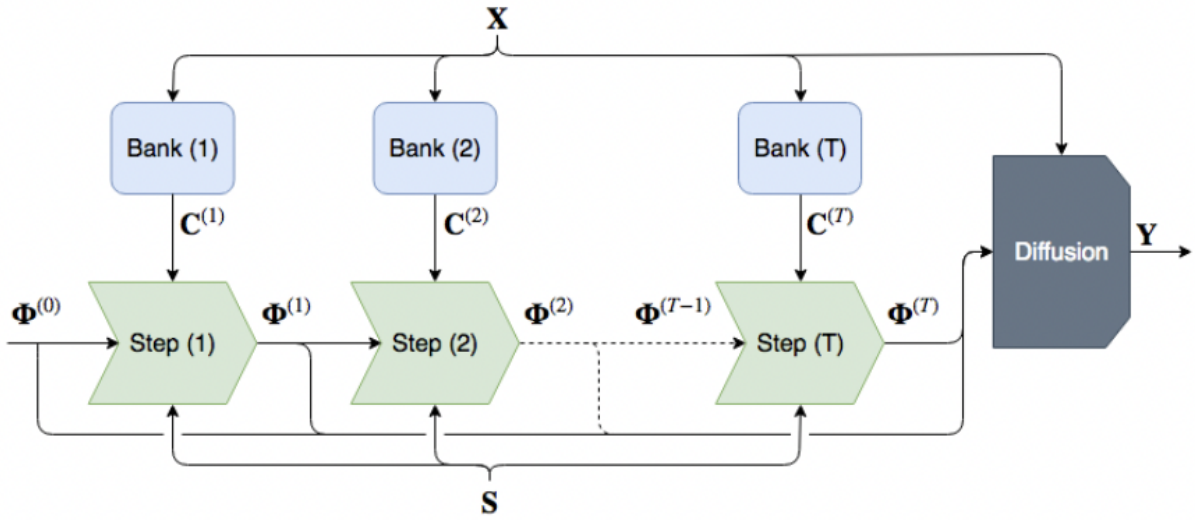


**Fig 3: Quantum Walk Neural Network Diagram**

A quantum walk on a graph is learned via a quantum walk neural network (QWNN), as seen in the above figure, by backpropagating gradient updates to the coin operators used in the walk. The signal is then spread throughout the graph using the learned walk.

## 2.4 Quantum Computing to Improve Classical Machine Learning

Various approaches that quantum computers utilize to resolve machine learning issues can be discussed now as we are aware of the fundamental principles of quantum computing.

### 2.4.1 Quantum Machine Learning to Solve Linear Algebraic Problems

Matrix operations on vectors in a high dimensional vector space are used to handle a wide range of Data Analysis and Machine Learning challenges.
The quantum state of a qubit in a quantum computer is a vector in a $2^a$-dimensional complex vector space. In this area, numerous matrix transformations take place. The Fourier Transform, identifying eigenvectors and eigenvalues, and resolving linear sets of equations over $2^a$-dimensional vector spaces can all be solved by quantum computers in polynomial-time (and exponentially faster than classical computers due to the Quantum Speedup). The Harrow, Hassidim, and Lloyd (HHL) algorithm is one illustration.

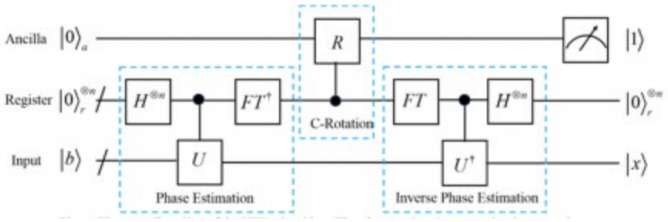


**Fig 4: Circuit Diagram of HHL algorithm**

### 2.4.1 Quantum Principal Component Analysis

Large datasets can have their dimensionality reduced using the dimensionality reduction approach known as principal component analysis. Accuracy must be sacrificed to reduce the dimensions since we must choose which variables to remove without losing crucial data. If done correctly, dealing with a smaller dataset is significantly more convenient, which greatly eases the machine learning work. For example, a traditional computer can efficiently analyze the principal component on a dataset with ten input attributes. However, the traditional methods of principal component analysis will not work if the input dataset has a million features since it will

be difficult to show the relative value of each feature.
The computation of eigenvectors and eigenvalues is another problem with traditional computers. The set of related eigenvectors and eigenvalues increases proportionally to the input's higher dimensionality. Quantum Random Access Memory (QRAM), which selects a data vector randomly, enables quantum computers to handle this problem quickly and efficiently. It employs qubits to translate the vector into a quantum state. Logarithmic qubits are included in the summarised vector that results from the quantum principal component analysis. The specified random vector creates a dense matrix. This matrix is the covariance matrix.
We can take the quantum version of any data vector and decompose it into its principal components by repeatedly sampling the data, employing a trick known as density matrix exponentiation, and using the quantum phase estimation algorithm (which determines the eigenvectors and eigenvalues of the matrices). Thus, there is an exponential reduction in both computational and time complexities.

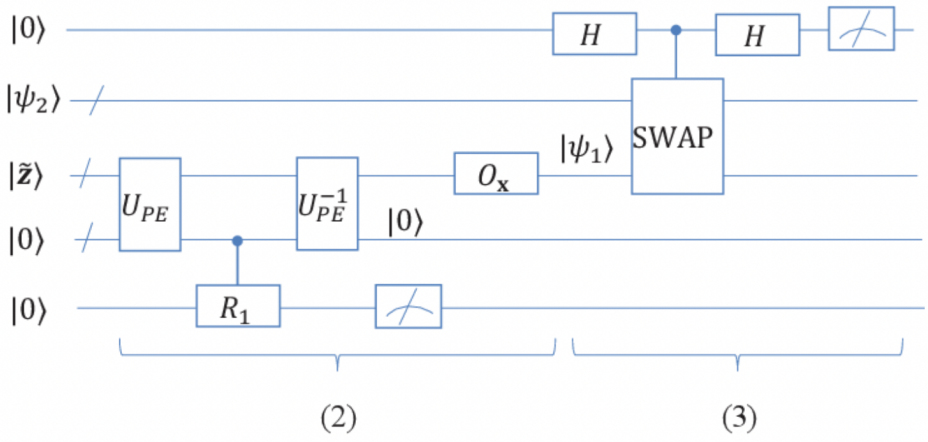


**Fig 5: Quantum Circuit to perform Principal Component Analysis**

## 2.4.1 Quantum Support Vector Machines

A traditional machine learning approach known as the Support Vector Machine is used for classification and regression. It categorizes linearly separable datasets into the appropriate classes for classification tasks. If the data cannot be separated linearly, let's say its dimensions are expanded till they can.

SVM can only be used in classical computers up to several dimensions. It will get difficult after a certain point because such computers do not have enough computing power.
However, quantum computers can execute the Support Vector Algorithm exponentially more quickly. It operates effectively and produces results more quickly thanks to the superposition and entanglement principles.

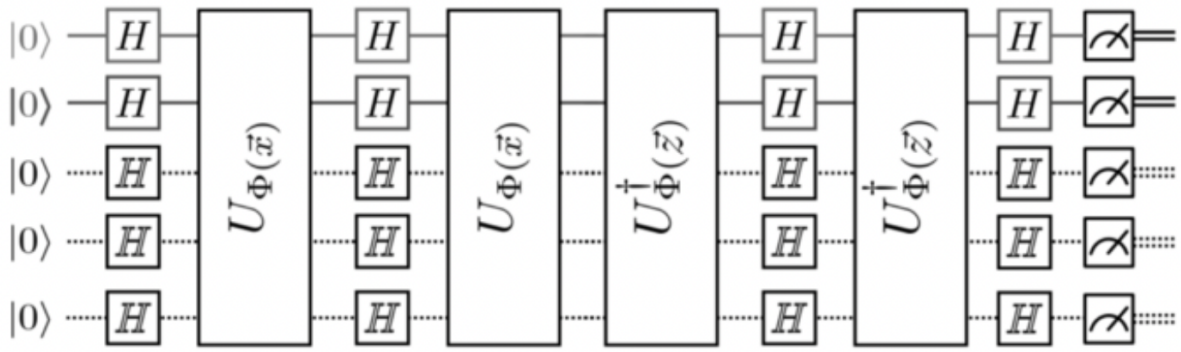


**Fig 6: Quantum Circuit to perform SVM**

## 2.4.1 Quantum Optimization

Optimization is attempting to create the greatest possible result while utilizing the fewest resources possible. A machine learning model uses optimization to enhance the learning process and produce the best and most accurate estimations.
Loss function minimization is the primary goal of optimization. An increased loss function will result in more inaccurate and unreliable outputs, which can be expensive and result in inaccurate estimates.
Iterative performance optimization is required for the majority of machine learning techniques. Quantum algorithms for optimization point to improvements in machine learning optimization problems. The ability to make multiple copies of the current solution, encoded in a quantum state, results from the quantum entanglement feature. Each stage of the machine learning algorithm uses them to enhance that solution.

### 2.4.5 Deep Quantum Learning

Deep learning and quantum computing can be used together to speed up neural network training. Using this technique, we may accomplish underlying optimization and create a new framework for deep learning. We can reproduce classical deep learning methods on a real, physical quantum computer.

As more neurons are added, the computational complexity rises when multi-layer perceptron topologies are used. Performance can be enhanced by using specialized GPU clusters, which also considerably cuts down on training time. Even this, though, will rise in comparison to quantum computers.

The hardware in quantum computers is built to imitate neural networks rather than the software found in traditional computers. Here, a qubit takes on the role of a neuron, the fundamental building block of a neural network. As a result, a quantum system with qubits can perform the function of a neural network and be used for deep learning applications at a rate that is faster than any traditional machine learning technique.

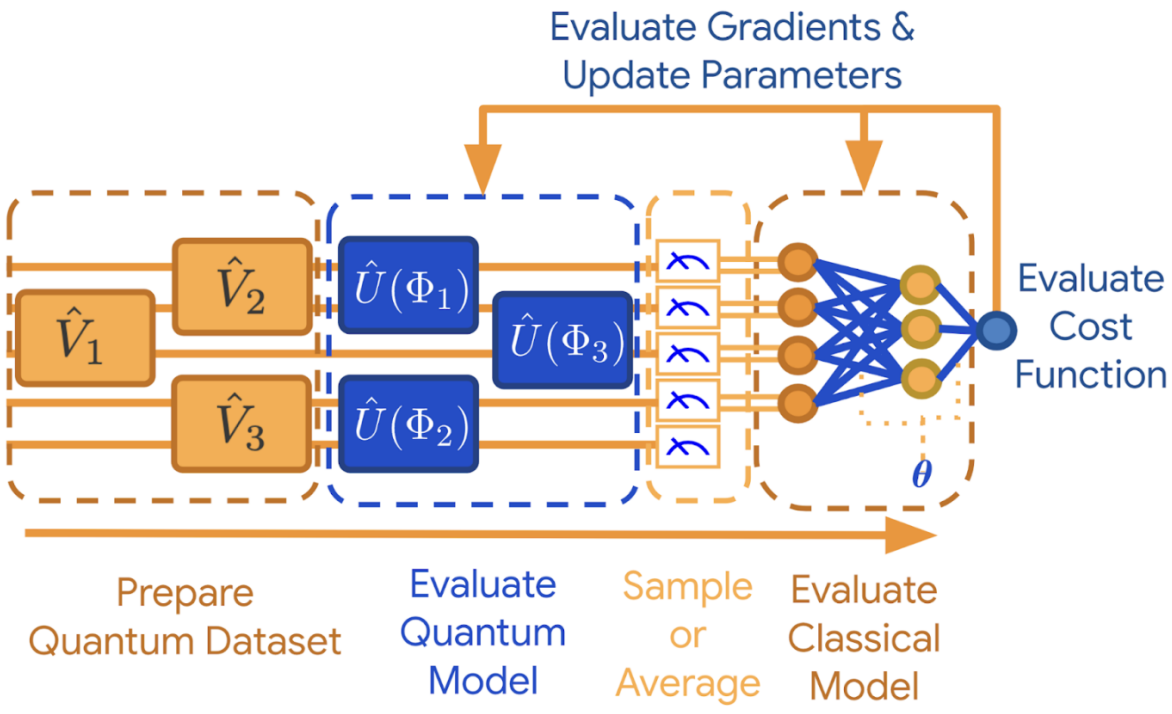


**Fig 7: Deep Quantum Learning**

## 2.5 Grover's Search Algorithm

Grover's algorithm, a quantum algorithm, handles one of the most challenging computing problems. It is the second important quantum computing algorithm that has been proposed. It addresses the issue of finding information in unstructured databases.
Let's now take a look at a scenario where we are trying to extract a value from a list of unsorted numbers N (shown in the red box below) that has some specific characteristics. Now, how do we find w?


**Fig 8: List in N numbers**

We would have to check on average N/2 of these things in the list with traditional computing. In plain English, we would have to check each of them individually, taking N steps or O (N). Binary search on ordered data requires log(N) steps or O(log(N)) if random access and pre-sort are incorporated. However, a quantum method will only take approximately √N steps, or O(N), to get w. For example, let's take a list of 25 elements and search for a value from it. A classical approach will require N steps, while a quantum algorithm will only require √N steps. i.e., classically, it will take 25 steps, and quadratically it will take √25=5 steps.
Superposition and phase interference is used to enhance the search. The amplitude amplification contributes most spatially to our search for the desired element. Let's see how the amplitude is amplified.
The amplitude amplification process lowers the other probability amplitudes while increasing the probability amplitude of the value to be searched.
If we think of the following Kets (k1, k2, k3,.....) in a balanced superposition, we seek kw. To distinguish the *kw* ket from other kets, the first oracle flips it upside down to a negative phase from the probability of .0 to -.5, which is accomplished by executing amplitude amplification.

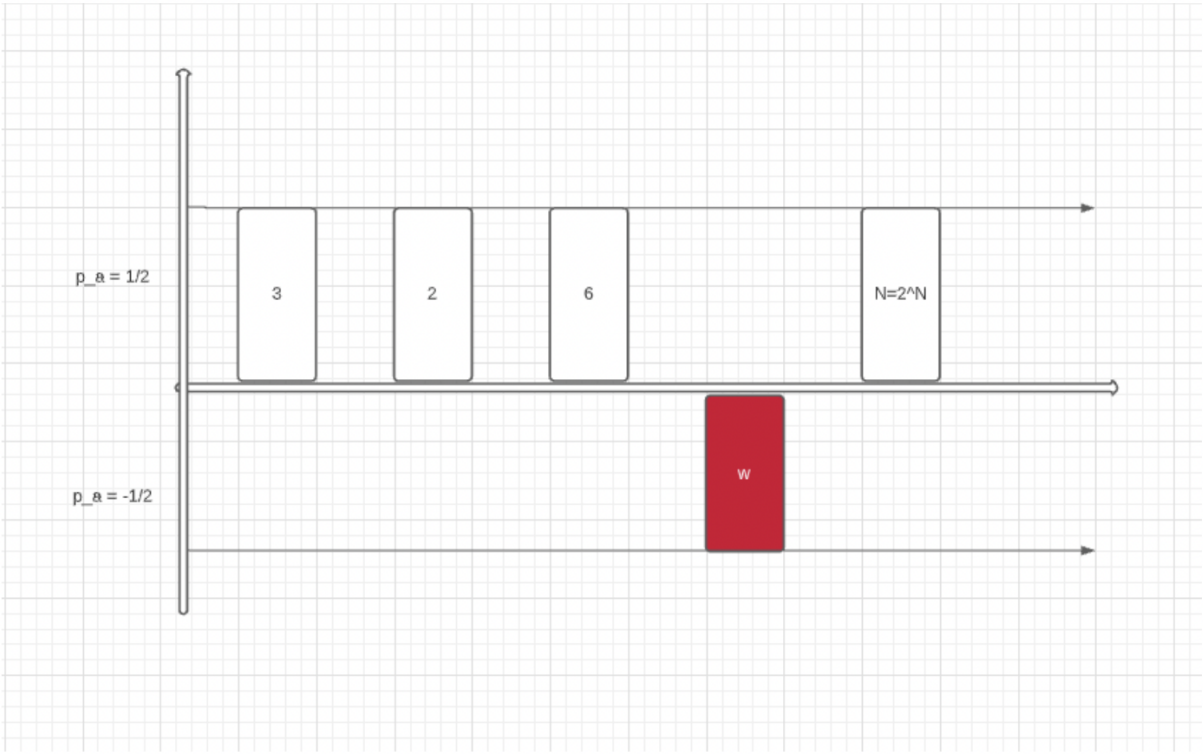


**Fig 9: Oracle**

The second oracle calculates each amplitude's mean, or average, and then inverts it. The result is a reduction in the amplitude A ∉ kw. Additionally, make the kw ket larger, so it becomes 1, and the other kets become 0.

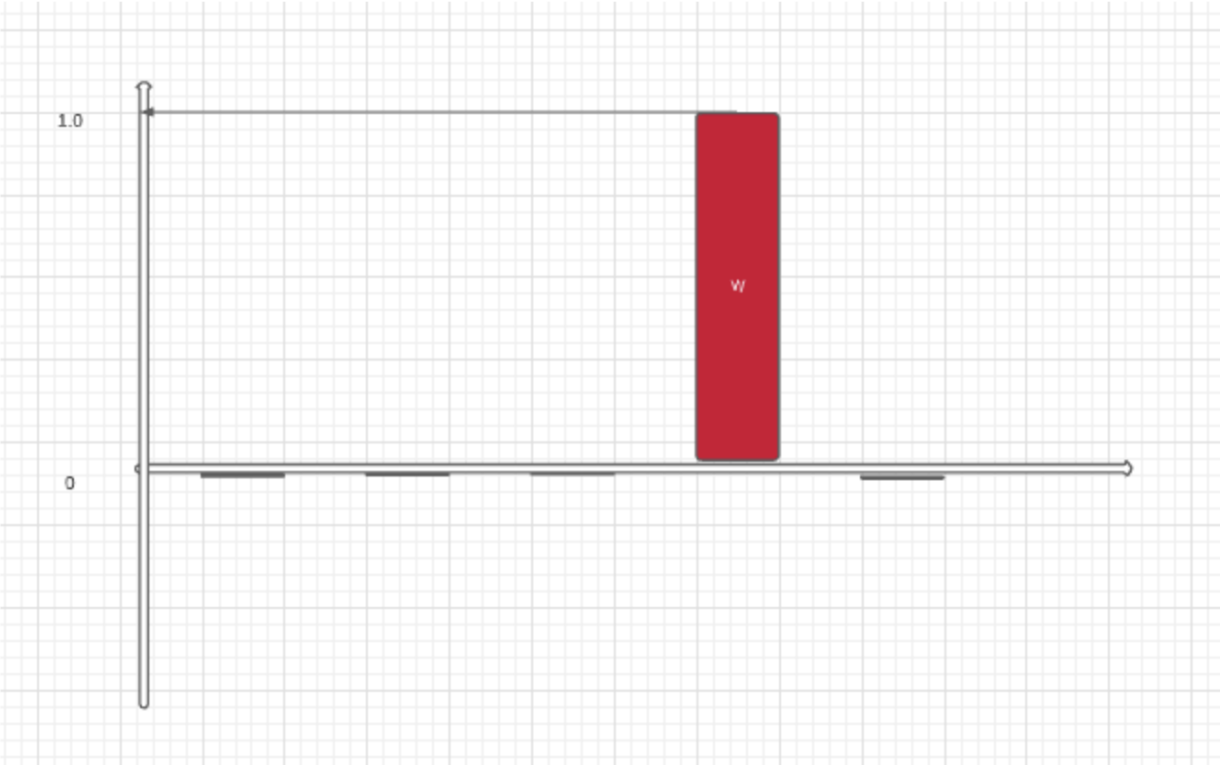


**Fig 10: Diffuser**

The states from the second oracle are measured at the very end using encoded values associated with the item on the list. The state that results is then mapped to the database. The stages are repeated if the response is incorrect. The likelihood of an error is O(1/N).

Here, we can suppose that diffuser and oracle are just two abstract entities manipulating qubits. Firstly, oracle is denoted as Uf and the second one as Uφ, which is Grover's diffusion operator. These oracles are repeated √N times. If the matches are multiple m, the oracles are repeated √(N/t) times. Grover search can also be used to search multiple elements.

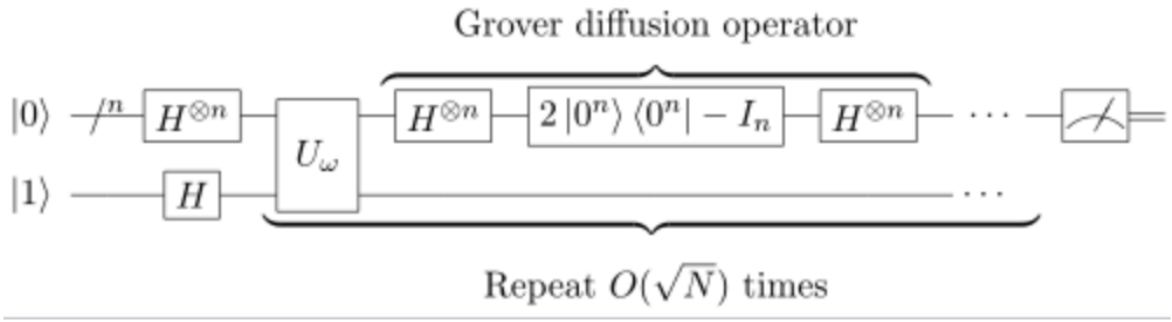


**Fig 11: Grover's Algorithm Circuit**

The above figure is Grover's algorithm circuit that follows the below algorithm :

1. Choose a random value from the qubits for the search.
2. Passing the qubits through Hadamard gate H will superposition all of them.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

3. Develop an oracle called f that flips the amplitude of the object being looked for.
4. Develop the Diffuser that reverts to the mean.

$$U_s = 2\,|s\rangle\,\langle s| - I.$$

5. Repeat the oracle √N times for a single target and √(N/t) for multiple targets.
6. Finally, validate the states with the values in the database.
7. Done.

# Chapter 3

## A Brief Introduction to Graph Neural Networks

Pattern recognition and data mining research have gained a boost thanks to the recent success of neural networks. End-to-end deep learning paradigms like CNN, RNN, and autoencoders have introduced 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 effectively detects hidden patterns in Euclidean data (images, text, videos). Graph Neural Networks (GNN) are useful in applications where data is generated from non-Euclidean domains, representing 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 various 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 a graph symmetry-preserving transformation that optimizes all graph properties (nodes, edges, and global context) (permutation invariances).

### 3.1 Graph

Graphs are a data structure representing a collection of items (nodes) and their connections (edges). Because of the great expressive power of graphs, i.e., graphs can be used as a 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 a Graph before moving on to GNN. A graph is a computer science data structure that comprises two parts: vertices and edges. The set of vertices V and edges E that makeup 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

$$G = (V, E)$$

Graphs can also represent social media networks or molecules if the nodes are considered users and the edges as connections.

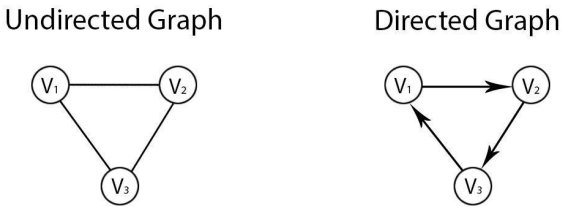


**Fig 12: Representation of an undirected and directed graph**

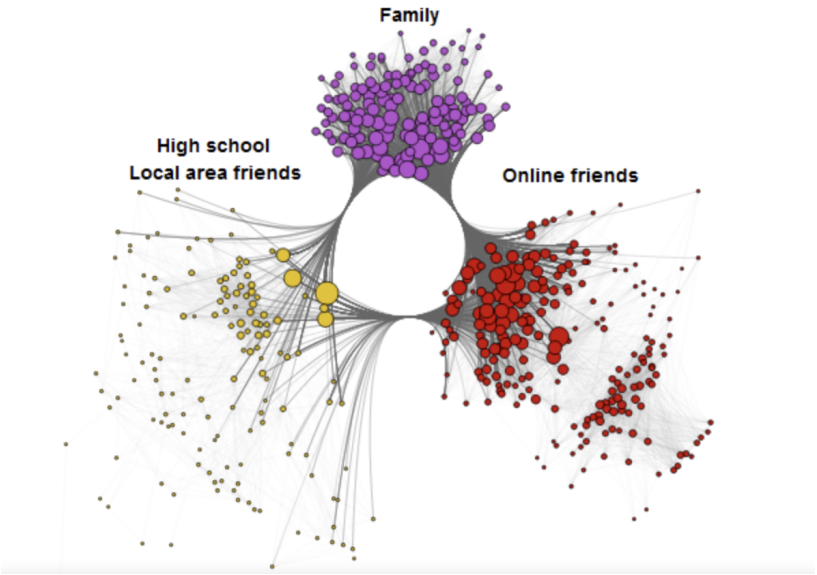


**Fig 13: Representation of a social media graph**

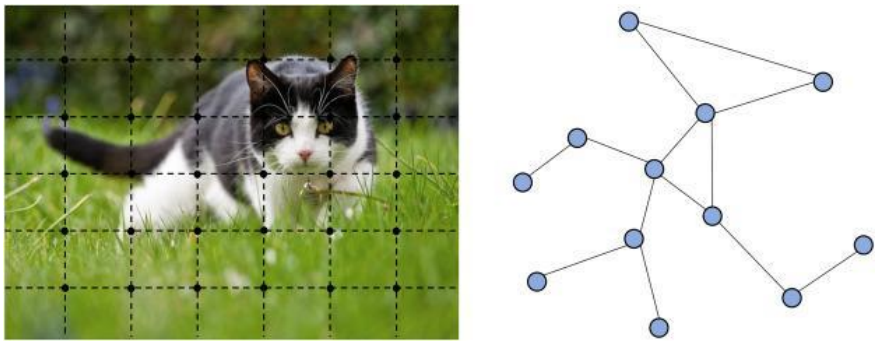

**Fig. 14. Left: image in Euclidean space. Right: graph in non-Euclidean space**

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 number of features.

Real-world objects are frequently characterized 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. Researchers have been developing neural networks that operate on graph data (known as graph neural networks, or GNNs) for more than a decade. 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.

## 3.1.1 Images as Graph

Images are rectangular grids with image channels, representing them as arrays (e.g., 244x244x3 floats). Images can also be considered 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.

An adjacency matrix is a tool for visualizing 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.

### 3.1.2 Text as Graph

Text can be digitized 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. In most cases, this is not how text and images are encoded in practice. This kind of graph representation is 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 following words, the adjacency matrix for text is merely a diagonal line.

### 3.1.3 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 that share a covalent link when a stable distance separates them. 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.

### 3.1.4 Social Networks as Graph

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

### 3.1.5 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.

### 3.1.6 Few other miscellaneous Graph representations

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

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

## 3.2 Functions of GNN

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

### 3.2.1 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 categorizing 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.

### 3.2.2 Node Classification

The node-level tasks focus on predicting the identity or role of each node in a graph. Basically,

this task includes estimating the node embedding for each node in a network. In such cases, only a piece of the graph is labeled, resulting in a semi-supervised graph. Examples are YouTube videos, Facebook friend suggestions, and other applications.

### 3.2.3 Link Prediction

The main goal of edge-level tasks is to determine how two things in a graph are related and predict whether or not they are connected. A recommender system in which a model is given a set of user reviews for various products. The goal is to predict user preferences and optimize the recommender system to offer relevant things 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 and identify them. We may think of this as edge-level classification: given a set of nodes representing the objects in the image, we want to predict which 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.

## 3.3 Embedding techniques

Until now, the above discussion concludes that we can not deny that Graph Neural Networks are currently a hot topic. GNNs are all about latent graph representation 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 intrigues GNN because while there are numerous ways to embed a phrase or image as a vector, none 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. A few of the methods are briefly discussed below.

### 3.3.1 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 vectorized text representations. Word2Vec, a popular approach for producing word embeddings, has many uses, including text similarity, recommendation systems, sentiment analysis, and more.Word2Vec was introduced by Google in 2013 and allowed words to be embedded in n-dimensional space, with similar words clustered together locally. This indicates that cosine distances between words frequently used together or in similar situations are less. Word2Vec compares the target words to their context using the Skip-Gram technique. Skip-Gram employs a sliding window technique at its basic level, which attempts to predict the surrounding words based on the target word in the center. This means we are effectively trying to estimate the neighbors around the target node inside our network in our use-case of encoding comparable nodes within the graph to be close to one another in n-dimensional space.

- **Word Embedding:** Before diving into word2vec, it's crucial first to 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. Using word embeddings, individual words are turned into a numerical representation of the word (a vector). Each word is mapped to a single vector, which is then trained like that of a neural network. The vectors attempt to capture various aspects of that word about 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 similarities or dissimilarities between words. These are important as inputs to many areas of machine learning. A machine cannot process text 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 several drawbacks, including 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 similar words' vectors. 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. For example, words like "King" and "Queen" are quite similar. 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 that is very close to the embedding vector of "queen."

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

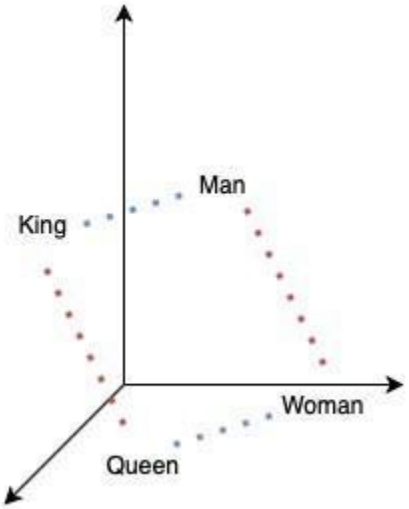


**Fig 15: 3-Dim Example of Word Embedding Behaviour**

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 similar to this architecture. This model architecture aims 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 the target word and ["have," "great," "day"] as context words. This model will use the distributed representations of the context words to predict the target word.

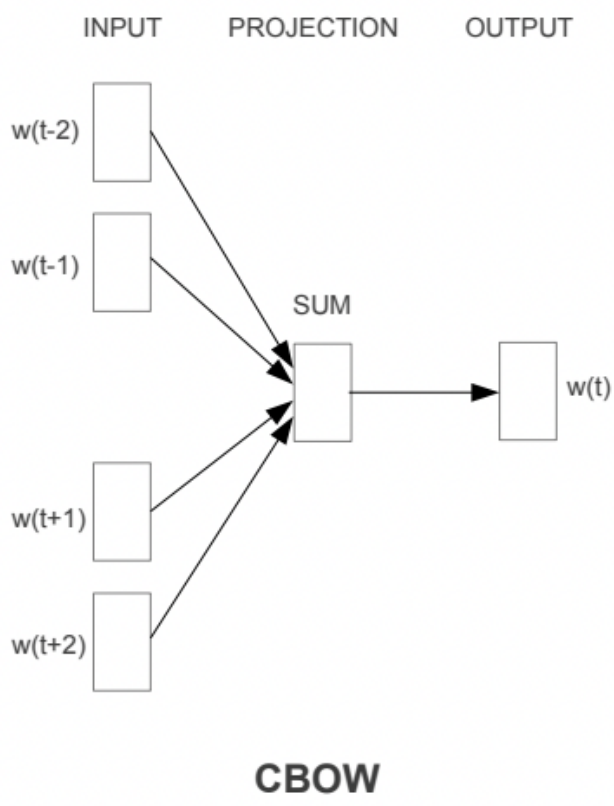


**Fig 16: CBOW Architecture.**

- **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 categorized by humans. One of the unsupervised learning strategies for finding the most similar words for a given the 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 because there are multiple context words to anticipate.

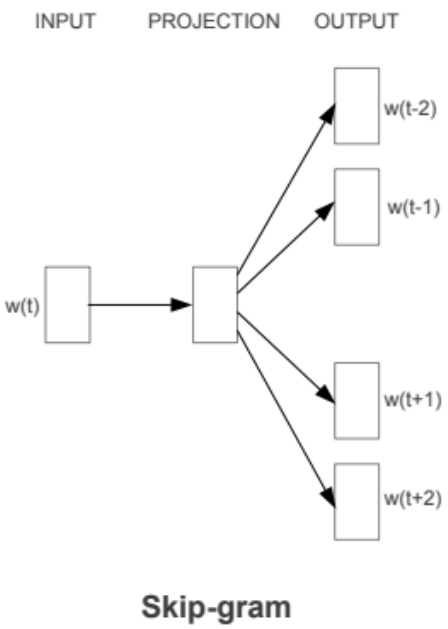


**Fig 17: The Skip-gram model architecture**

Here, the input is 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. The output layer computes the dot product between the hidden layer's output vector and the output layer's weight matrix. Then we apply the softmax activation function to compute the probability of words appearing in the context of w(t) at a given context location.

### 3.3.2 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 fewer 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. A vector in these embedding spaces represents each node in the network. Graph embeddings are frequently utilized 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. 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 traveling a certain distance in a certain direction. Random walks are a Markov process in which future behavior is unaffected by previous behavior. In general, scientists find it challenging to graphically communicate the data they're working with when dealing with huge graphs. One typical technique to examine how a graph look is constructing node embeddings and then visualizing them 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. From a high-level perspective, the simplest way to compare a random walk is to walk. Assuming that each step we take is determined probabilistically, 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 wonder how these odds of going from one node to the next are determined. Node2Vec introduces the following formula for estimating the likelihood of traveling to node x if we were previously at node v. Here, $\pi_{vx}$ is the unnormalized transition probability between nodes x and v, and z is the normalization

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

(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 normalized probability of travelling from v to x is found. If each edge had 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 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:

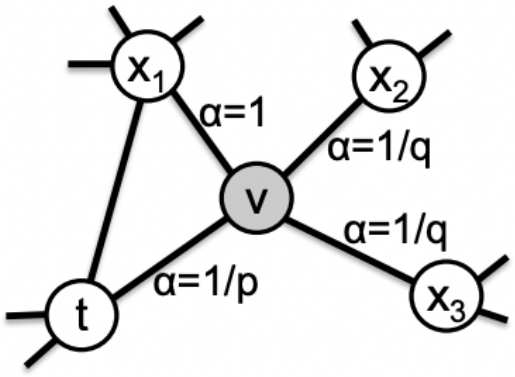


**Fig 18: Illustration of the random walk procedure in node2vec. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.**

- **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 the word being present. As seen in the diagram below, the process can be visually illustrated as follows:

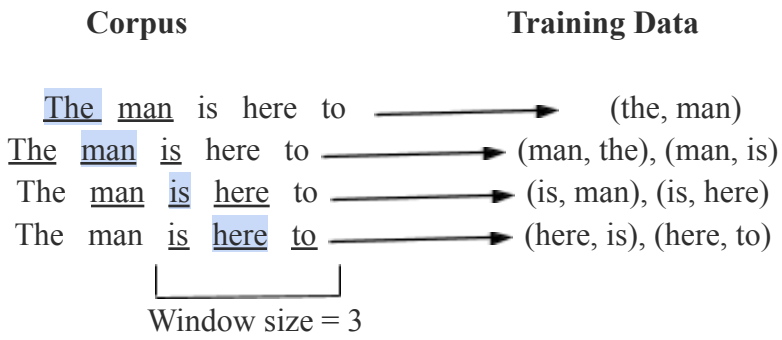


**Fig 19: Example of generating training data for the skip-gram model. The window size is 3**

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. The hidden layer will learn each word's embedding representation, 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 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).

### 3.3.3 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 effectively identify diverse structures within communities, they cannot 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 some papers provide easy introductions to the DeepWalk algorithm, few include code and describe implementation specifics for these systems that could be discovered. Model parametrization, deployment considerations, and handling unseen data are covered in detail. DeepWalk can transform a graph from force layout to vector space visualisation while maintaining some 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 very straightforwardly: Starting at the target root, a random neighbor of that node is chosen and added to the path, followed by another random neighbor 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.

## 3.4 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 neighbors to determine the labeling of samples (shown as nodes). Problems of this nature are usually taught semi-supervised, with only a portion of the graph labeled.

2. **Graph Classification:** The goal is to categorize 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 categorizing articles in natural language processing or social network analysis.

3. **Graph Visualisation:** At the intersection of geometric graph theory and information visualization, this is a mathematics and computer science field. It is focused on 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 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 based on their similarity.

6. **GNNs in computer vision:** Machines can differentiate and identify objects in pictures and videos using standard CNNs. However, more work must be done before machines 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, discover and recognise things, and 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 characterized 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 in interest in using GNNs to solve various 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 generalized 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, nodes represent road sensors, edges represent the distance between pairs of nodes, and each node has 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, program reasoning, social influence prediction, recommender systems, electrical health records modeling, brain networks, and adversarial attack protection have all been attempted using GNNs.

# Chapter 4

## Multiview Networks

Real-world data sets frequently provide multiple sorts of information on the same set of things. 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 comprises one type of relationship, and all views share the same set of objects. A multi-view network can be constructed from a pair of single-view networks describing various relationships. 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 to model and understand complex systems. Many relational data sets offer numerous perspectives on the same set of entities. For example, a group of people's interactions with one another on a social networking platform can be used to classify them. As a first estimate, we can only examine if a relationship exists between two people on this platform. However, we can investigate their interactions across other platforms or 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.

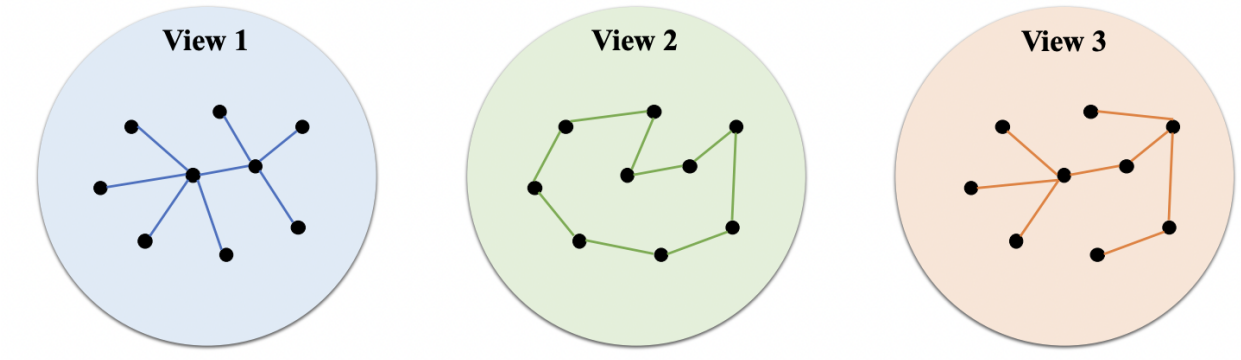


**Fig 20: An example of a multi-view network with three views. Each view corresponds to a type of proximity between nodes, characterized by a set of edges. Different views are complementary to each other.**

Mathematically, a multi-view network can be defined as the 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** $\in$ **V** × **V, X** $\in$ **R** $^{|V|\times F}$ , 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 views $\cup$**r** $\in$ **R{Gr}**,
  where **Gr = (V, Er), Er** $\in$ **E** is the set of all edges of type **r** $\in$ **R.**

# 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 proximity interaction 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, is being investigated for learning node representations from each individual view. A view-focused attention strategy is investigated to aggregate the view-wise node representations to integrate various sorts of relationships in various views cooperatively. The suggested approach outperforms current state-of-the-art baselines when the algorithm is tested on various real-world datasets.

This paper learns a set of view-specific node representations 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 learned weights 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 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. The attention-based aggregation component integrates different views for more thorough node representations.

MGAT, a brand-new multi-view graph embedding method, has been presented in this study. More specifically, a novel attention-based mechanism for learning the weights and encouraging collaboration across various views and a 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 whether modeling 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 further show that improved embedding can be attained without overly complicating the model or necessitating more supervision by simultaneously modeling the two objectives. The primary challenge lies in using the type of information on edges from different views to design embedding methods for multi-view networks. For that reason, two such objectives are identified, preservation and collaboration, from the practice of embedding real-world networks. The need for preserving unique information carried by different views is termed preservation; the need for collaboration and 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, these two people are likely content to be connected. In this case, the ideas might be complementary, and combining them might result in better outcomes than combining them separately. We refer to collaboration 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 recognized and examined in this research. The viability of simultaneously modeling 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 various 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 studies 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) 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 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 various 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, visualization, and link prediction tasks.

This research introduces a novel network representation learning (MVNR) method that directly incorporates global characteristics and higher order adjacent relations. Regarding 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 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. It can supply sufficient structure features to solve the sparse problem of the structure feature matrix in sparse networks. The MVNR algorithm produces superior visualization 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 utilized in this research regarding link prediction performance.

So, 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 offered. Future research will 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 regarding 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 favorite videos, but they can also be linked in another 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 depends on diversity and collaboration, much like earlier research; diversity allows views to preserve their semantics, while collaboration allows views to cooperate. A new type of second-order collaboration that had not been studied before was also found and further incorporated into the framework to get better node representations. 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. MANE+ is further enhanced through an attention mechanism 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. Intra-view pairs, which can be produced using random walks in each view, are considered in the first category. It can capture the variety of various viewpoints by modeling intra-view pairs in each view. Cross-view, intra-node pairs, formed when occurrences of the same node (i.e., intra-node) across two views (i.e., cross-view), are considered 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 considered to capture second-order cooperation. Here, a node in one view forms pairs with distinct nodes (i.e., cross-node) in another (i.e., cross-view) depending on their affiliations. 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 are developed to improve alignment between two KGs. According to studies on real-world datasets, the proposed framework greatly outperforms the most advanced embedding-based entity alignment algorithms. 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. Tests demonstrated the framework's usefulness on two real-world datasets. Future research will look into additional practical viewpoints (such as entity types) and examine cross-lingual entity alignment.

## 5.6 Quantum Neural networks for Face Recognition Classifier

This paper describes a face recognition method based on multi-level quantum neural networks (QNN) and multi-layer classifiers. First, image preprocessing helps detect the position of the eyes by removing irrelevant information from face photos. Second, to extract statistical features and minimize dimensions, feature extraction uses the eigenface approach based on the Karhunen-Loeve transform. Last but not least, quantum neural networks based on multi-level transform function are employed in the facial recognition component. The ORL faces digital database trains and tests the QNN. The results demonstrate that the identification approach in more complicated contexts with a certain degree of robustness is effective and practicable simultaneously with the classical BP neural network classifiers.

The major focus of the paper is the study and creation of a face identifier based on quantum neural networks. The human eye localization methods were attained through filtering, grey equalization, and binary operation. After locating the eyes in the images, a normalized processing technique based on the KL transformation and singular value decomposition (SVD) is used to further feature extraction. Finally, a method for quantum neural networks that relies on a multi-level excitation function is proposed. This experiment demonstrates the effectiveness and enormous potential of the quantum neural network for pattern recognition, as well as the method's collective and adaptive learning capabilities, robustness, and strong fault tolerance.

## 5.7 Facial Expression Recognition on a Quantum Computer

Using a quantum machine learning approach, the issue of face expression detection may be stated and illustrated as a potential solution. This method heavily utilizes quantum interference to define an effective classifier for a given dataset. A classifier is a quantum circuit that manipulates the graph adjacency matrices stored into the amplitudes of some adequately defined quantum states and represents facial expressions via graphs. The accuracy of the quantum classifier is then reviewed, analyzed, and compared with one of the finest classical classifiers using the quantum simulator accessible on the IBM Quantum Experience cloud platform.

The issue of categorizing graphs that depict human facial expressions has been addressed in this work. Graphs are created by choosing a few face landmark points and connecting them either through a complete graph construction or by triangulating the points. The creation of quantum states whose amplitudes encode graphs has been demonstrated, and a quantum circuit based on interference that performs distance-based classification for the encoded graphs has been developed. The Cohn-Kanade (CK) database is one of the most popular test-beds for algorithm development and evaluation in automatic facial expressions recognition. This quantum circuit is then implemented using the IBM open-source quantum computing framework Qiskit (2019) and tested on a collection of images. The tests were run on a qasm simulator, and the outcomes were compared to those of the traditional classifier. The simulated quantum classifier obtains equivalent results with both the meshed and complete strategies; however, the classical classifier performs better when a complete graph approach is used, according to an accuracy calculation.

## 5.8 Quantum Graph Neural Networks

Quantum Graph Neural Networks (QGNN), a new class of quantum neural network ansatz, have been developed and are specifically suited to be operated on distributed quantum systems across a quantum network. They are tailored to represent quantum processes that have a graph structure. Other specialized designs, such as Quantum Graph Recurrent Neural Networks (QGRNN) and Quantum Graph Convolutional Neural Networks (QGCNN), are now being introduced in addition to this general class of ansatz. The following four examples of QGNN applications are given: unsupervised learning for spectrum clustering supervised learning for

graph isomorphism classification, learning Hamiltonian dynamics of quantum systems, and learning how to construct multipartite entanglement in a quantum network.

The findings presented in this work represent a good set of initial investigations into the potential uses of QGNNs. We have demonstrated the application of these QGNN ansatzes to quantum dynamics learning, quantum sensor network optimization, unsupervised graph clustering, and supervised graph isomorphism classification through our numerical experiments. Future research should investigate hybrid approaches where one can learn a graph-based hidden quantum representation (via a QGNN) of a quantum chemical process. This is because there is a large body of literature on using graph neural networks and their variants to quantum chemistry. The QGNN may be a more accurate representation of the hidden processes that give rise to apparent emergent chemical features because the true underlying process is quantum in nature and has a natural molecular graph geometry. Future studies may include expanding the QSGCNN to support several features per node, generalizing the QGNN to incorporate quantum degrees of freedom on the edges, and training the graph parameters using quantum optimization techniques via quantum phase backpropagation.

## 5.9 Quantum Walk Neural Networks for Graph Structured Data

The state-of-the-art in the subject has recently been advanced by novel neural network designs created to operate on graph-structured data. Many of these structures employ a traditional random walk technique to spread information. A novel graph neural network design based on quantum random walks—the quantum counterpart to conventional random walks—is suggested as the quantum walk neural networks (QWNN). To create a diffusion operator that can be used with graph-structured data, a QWNN learns a quantum walk on a graph. We employ graphs using temperature, biological, and molecular datasets to show how QWNNs can be used for various prediction applications.

While traditional neural network methods for structuring data have received much study, there is considerable interest in expanding neural network architectures to handle graph-structured data. This work describes a neural network structure based on quantum walks that may be used with graph data. Exploring the possibilities of using quantum techniques to handle graph-structured data and the possible benefit that it might provide over classical algorithms is

one of the main objectives behind this work. A discrete quantum walk behaves differently from a classical random walk because it is controlled by extra operators that can be adjusted to influence the diffusion process depending on the learning goal at hand. One of the main potential implications of quantum-inspired machine learning algorithms is their implementation on a quantum computer and potential advantages in solving classical issues on classical computers. This research marks the first step to creating a quantum neural network technology that can work with graph-structured data. This work focuses on using simulated quantum walks in a classical setting, even though quantum walks have been realized utilizing various physical systems, including optical lattices and molecules.
On a variety of graph datasets, it has been demonstrated that this quantum walk-based neural network approach achieves better or comparable results compared to other state-of-the-art graph neural network approaches, indicating that graph-structured data should focus on further research into quantum techniques.

## 5.10 A fast and elitist multiobjective genetic algorithm: NSGA-II

The major reasons why non-elitist multi-objective evolutionary algorithms (MOEAs) have drawn criticism are (1) their computational cost, (2) their non-elitism approach, and (3) the requirement to define a sharing parameter. This work proposes NSGA-II (Non-dominated Sorting Genetic Algorithm II), a non-dominated sorting-based MOEA that resolves all three issues. In particular, a quick, computationally simple non-dominated sorting strategy is provided. The best N solutions are chosen by mixing the parent and offspring populations in a selection operator that generates a mating pool (concerning fitness and spread). The Pareto-archived evolution strategy and the strength-Pareto evolutionary algorithm are two other elitist MOEAs that pay particular attention to creating a diverse Pareto-optimal front, but simulation results on challenging test problems show that NSGA-II is able, for most problems, to find a much better spread of solutions and better convergence near the true Pareto-optimal front. Additionally, we change the notion of dominance to resolve limited multi-objective issues effectively. Compared to another restricted multi-objective optimizer, the constrained NSGA-II performs significantly better in simulation results on various test problems, including a five-objective, seven-constraint nonlinear problem.

# 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, we used a multiple ranking technique to trim the graph based on its sub-components to find the most informative results. We then use a quantum walk neural network modified for graph-structured input to enable the 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 $G = G_i$ , $i = 1$ to $M$, where each layer $G_i$ has the same vertex set $V$ but distinct or the same edges set Ei. The normalized 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.

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

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

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

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 $L_i$, 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 considered a collection of linear subspaces in Rn of dimension k, where each 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 minimize the quadratic-form Laplacian (evaluated on the columns of U) to guarantee that the original vertex connectivity in each graph layer is preserved:

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

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 hyperparameter that ensures the message passing of the two networks.

The two objective functions are optimized using the multi-objective Non-Dominated Sorting Genetic Algorithm II (NSGA-II) to get the final subspace.

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

Model performance can be improved by ranking the nodes in the manifold according to their saliency regarding selecting essential nodes. 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 $\beta$ is the regularisation parameter.

**Algorithm: Subspace merging**

**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,
$E_{ns}$: Nonsalient 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 optimize them using the NSGA II 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 centroids, 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 $A_{mod}$

The adjacency matrix $A_{mod}$ obtained from the algorithm represents the merged views of the multiview graph network. We have obtained the salient edges (relations) and omitted the non-salient edges from the graph data. The next step is feeding the merged graph to a quantum walk neural net for classification.

## 6.3 Quantum Walk Neural Network

Given an undirected graph G = (V, E), the position Hilbert space is $\mathcal{H}_{\mathcal{P}}$ spanned by basis vectors $\{\hat{\mathbf{e}}_v^{(p)},\ v \in V\}$ and the coin space $\mathcal{H}_{\mathcal{C}}$, as an auxiliary Hilbert space of dimension $d_{max}$ spanned by the basis vectors $\{\hat{\mathbf{e}}_i^{(c)},\ i \in 1,\ldots,d_{max}\}$, i corresponds to the edges incident on a vertex and $d_{max}$ is the maximum degree of the graph.

The tensor product of the position and coin Hilbert space defines the associated Hilbert space of the quantum walk $\mathcal{H}_{\mathcal{W}} = \mathcal{H}_{\mathcal{P}} \bigotimes \mathcal{H}_{\mathcal{C}}$.

The position vector of the quantum walker is defined as a linear combination of position state basis vectors, $\sum_{v\in V} \alpha_v \hat{\mathbf{e}}_v^{(p)}$, for some coefficients $\{\beta_{v,i},\ i \in 1,\ldots,d\}$ satisfying $\sum_i |\beta_{v,i}|^2 = 1$. The coefficients $|\alpha_v|^2$ and $|\beta_{v,i}|^2$ indicates the probability of finding the walker at vertex *v* and the walker having spin *i* at that vertex, respectively, when a measurement is done.

The coin operator $\mathcal{C}$ transforms the spin state of a vertex. The unitary shift operator $\mathcal{S}$ swaps the states of two vertices connected by an edge.

Let the quantum walker's superposition state at t be represented by the symbol $\Phi^{(t)}$. If $\Phi^{(0)}$ is the beginning state of the discrete quantum walk on graph G, then after t time steps, the state of the walker is defined by: $\Phi^{(t)}=U^t\Phi^{(0)}$, where $U = \mathcal{S}(\mathcal{C} \otimes I_{|V|})$ can be used to express one step of the discrete quantum walk on graph G.

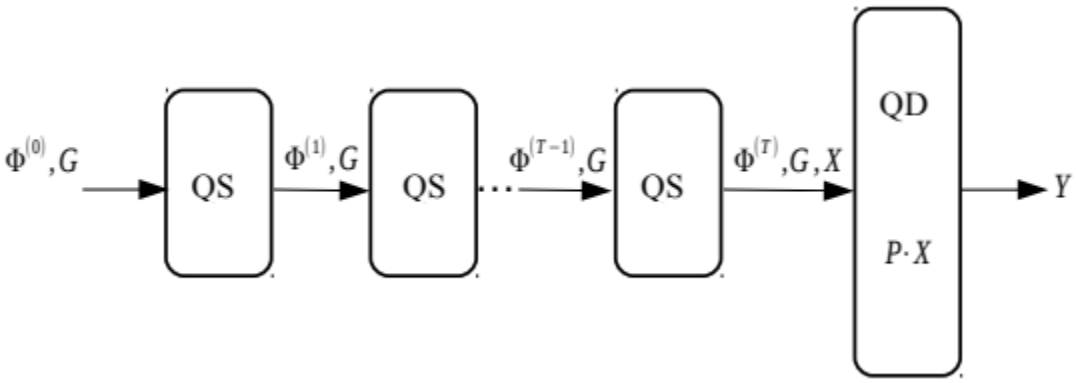


**Fig 21: Schematic Diagram of a QWNN architecture**

QWNN learns a quantum random walk on a graph by means of learning the coin operators

and/or the initial superposition of the walker. For each node, there is a separate coin operator. For the feature matrix $F$ as input along with the modified graph $A_{mod}$ , the QWNN will generate the diffused features $Y$. The final diffusion equation $Y \leftarrow PX$.

**Algorithm: QWNN**

**Input:** Initial Superpositions $\Phi^{(0)}$, Coins $C$, Shift $S$, Features $X$
**Output:** Diffused feature matrix **Y**
**Steps:**

1. for t = 1 to T do
2. for all nodes vi do
3. $\mathbf{\Phi}_{\cdot i \cdot}^{(t)} \leftarrow \mathbf{\Phi}_{\cdot i \cdot}^{(t-1)} \cdot \mathbf{C}_{i \cdot \cdot}$
4. end for
5. $\mathbf{\Phi}^{(t)} \leftarrow \mathbf{\Phi}^{(t)} : \mathbf{S} \quad \left(\text{i.e., } \mathbf{\Phi}_{wuj}^{(T)} = \sum_v \sum_i \mathbf{\Phi}_{wvi}^{(T)} S_{viuj}\right)$
6. end for
7. $\tilde{\mathbf{P}} \leftarrow \sum_k \mathbf{\Phi}_{\cdot \cdot k}^{(T)} \odot \mathbf{\Phi}_{\cdot \cdot k}^{(T)}$
8. $\mathbf{Y} \leftarrow h(\tilde{\mathbf{P}}\mathbf{X} + \mathbf{b})$
9. return **Y**

Finally, we will have the diffused feature matrix **Y** and the subspace merged adjacency matrix $\boldsymbol{A}_{mod.}$ For the classification, we have used a Graph Convolutional Neural Network.

# 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 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 across 3312 publications in the data set. There are 2 views and 6 categories on this network.

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

# Chapter 8

## Experimental results and discussion

Our objective is to correctly classify nodes in a network where only a relatively tiny percentage of nodes are labeled. In the tests, we begin with a small sample of labeled nodes and test Multi-prowess GCN's at correctly classifying unlabeled nodes in the validation and testing sets and many other cutting-edge algorithms. As the 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. **Qiskit :**

Qiskit is a python library for quantum computing development and simulations. It includes a comprehensive set of quantum gates and a variety of pre-built circuits so users at all levels can use Qiskit for research and application development. It allows one to design experiments and applications easily and run them on real quantum computers and/or classical simulators.

2. **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 stands to have uses in other fields since it is just a massive array manipulation library.

3. **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 it for data preprocessing like feature extraction and normalization and machine learning algorithms for classification and improving parameters via parameter tuning and comparing models with metric functions.

4. **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.

5. **Network:**

It is a utility that is built on top of matplotlib and allows us to visualize 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 convolutional networks. The following is a representation of the forward propagation model for a two-layer network:

$$Z = f(Y, A_{mod}) = \text{softmax}(\hat{A}\ \text{ReLU}\ (\hat{A}\ Y\ W^{(0)})\ W^{(1)})$$

Here, $\hat{A} = \tilde{D}^{-1/2}\tilde{A}_{mod}\tilde{D}^{-1/2}$ 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 | 70.9 ± 0.5 |
| Our Method | 83.0 ± 0.7 | 72.5 ± 0.7 |

**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 |

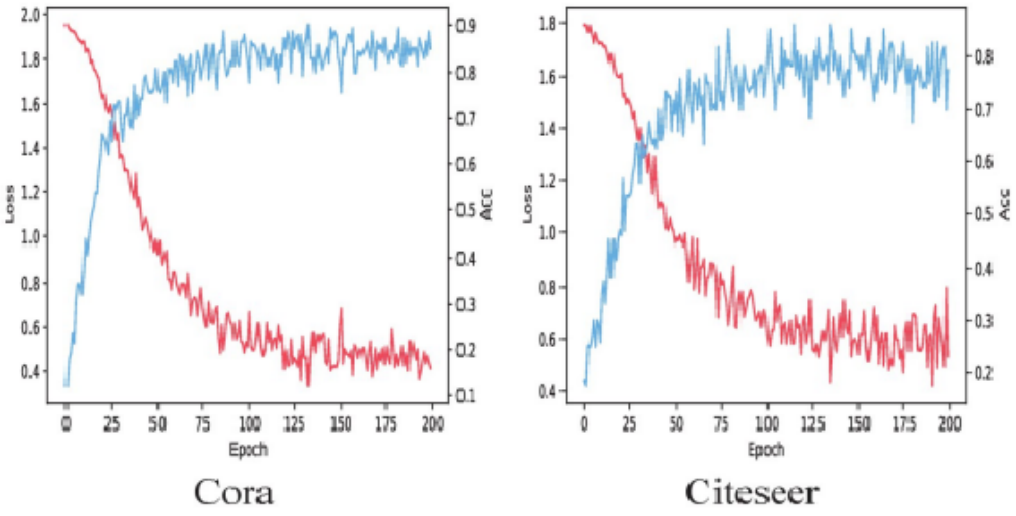


**Fig 22: Iteration influence on loss (red) and accuracy (blue)**

# Chapter 9

## Conclusion and Scope for Future Works

Representation learning for graph data has achieved considerable success in machine learning. With the advent of social media data and relational data, the ability to learn patterns from network data is very necessary nowadays. In many cases, network data might consist 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 Quantum Walk Neural Networks. Quantum walk generates faster spreading and can adapt both spatially and temporally. Hence we can perform node-level classification with the help of our proposed model with better accuracy than the benchmark tests. However, the constant coin and shift operations lead to a slowdown of the model compared to others. Also, it requires more space complexity.

Our project's future scope involves link prediction, subgraph prediction, and graph regression. Also, we would like to extend our training with different multiview datasets like Twitter, Youtube, and Snapchat. In the project, the regularisation and message passing parameters are fixed. We would like to make them adaptive in the future and include them as part of optimization. A study into learning multi-walker quantum walks is worth looking at as reducing the number of independent walkers will reduce the space complexity.

# 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://www.investopedia.com/terms/q/quantum-computing.asp#toc-understanding-quantum-computing

[3]https://www.wired.co.uk/article/quantum-computing-explained

[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]https://www.ibm.com/topics/quantum-computing#:~:text=Quantum%20computing%20is%20a%20rapidly,available%20to%20thousands%20of%20developers.

[8]CB Insights. "Quantum Computing vs. Classical Computing in One Graphic." https://www.cbinsights.com/research/quantum-computing-classical-computing-comparison-infographic/Accessed Sept. 2, 2021

[9] University of Waterloo. "The Future Is Quantum." Accessed Sept. 2, 2021. https://pennylane.ai/qml/whatisqml.html

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

[11] Quantum Walk Neural Networks with Feature Dependent Coins. Stefan Dernbach , Arman Mohseni-Kabir , Siddharth Pal , Miles Gepner and Don Towsley

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

[13] Dernbach, S., Mohseni-Kabir, A., Pal, S., Towsley, D.: Quantum walk neural networks. In: Seventh International Conference on Complex Networks and Their Applications (2018)

[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] Quantum Graph Neural Networks Guillaume Verdon, The Moonshot Factory Mountain View, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, Jack Hidary

[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]https://towardsdatascience.com/grovers-search-algorithm-simplified-4d4266bae29e#:~:text=Grover's%20algorithm%20i.e%20the%20quantum,of%20searching%20through%20unstructured%20data.

[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]https://upload.wikimedia.org/wikipedia/commons/thumb/a/ae/Grovers_algorithm.svg/500px-Grovers_algorithm.svg.png(Grover's algo ckt img source)

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