THE UNIVERSITY OF SYDNEY

BUSINESS SCHOOL

DISCIPLINE OF BUSINESS ANALYTICS

MPHIL THESIS

DEEP LEARNING STRUCTURE FOR DIRECTED GRAPH DATA

Author:

Chunya Zou

SID: [redacted]

Supervisors:

Prof. Junbin Gao

A/Prof. Dmytro Matsypura

Dr. Stephen Tierney

A thesis submitted in fulfilment of the requirements for the degree of

Master of Philosophy

Discipline of Business Analytics, Business School

The University of Sydney

Abstract

Deep learning structures have achieved outstanding success in many different domains. Existing research works have proposed and presented many state-of-the-art deep neural networks to solve different learning tasks in various research fields such as speech processing and image recognition. Graph neural networks (GNNs) are considered as a type of deep neural network and their numerical representation from the graph does improve the performance of networks. In the real-world cases, data is not only in the form of simple graph, but also they could contain direction information in the graph resulting in the so-called directed graph data.

This thesis will introduce and explain the first attempt in this domain to apply Singular Value Decomposition (SVD) on adjacency matrix for graph convolutional neural networks and propose SVD-GCN. This thesis also utilizes the framelet decomposition to help better filter the graph signals, thus to improve novel structure's performance in node classification task and to enhance the robustness of the model when encountering high-level noise attack. The thesis also applies the new model on link prediction tasks. All the experimental results demonstrate SVD-GCN's outstanding performances in both node-level and edge-level learning tasks.

CERTIFICATE OF ORIGINALITY

I certify that this thesis work is my own work, and it has not been submitted or published for any degree or other purposes in other educational institutions. I declare that any contribution made to the research by other people, with whom I have worked at the University of Sydney or elsewhere, is explicitly acknowledged in the thesis.

Chunya ZOU

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I'd like to express my sincere gratitude towards my supervisors, Prof. Junbin Gao, A/Prof. Dmytro Matsypura and Dr. Stephen Tierney for their advice and continuous support and patience during my MPhil study. Their immense research experience and abundant professional knowledge in computer science and mathematics as well as other fields have helped me and guided me through my research study. Secondly, I would like to thank all my co-authors for sharing amazing ideas and opinions and making contributions in the paper. It is a delightful experience to work and research with these hardworking and intelligent scholars.

My sincere gratitude also goes to my friends and family for their tremendous support and belief in me throughout my research study. I really appreciate that they are always there when I am in low spirits and help me get through the difficult time.

Publications

This thesis is mainly based on the research works listed below:

- **Zou, Chunya**, Andi Han, Lequan Lin, and Junbin Gao. A simple yet effective SVD-GCN for directed graphs. Submitted to IEEE Transactions on Artificial Intelligence. *arXiv preprint arXiv: 2205.09335* (2022). (Under 2nd Round Review)

Authorship Attribution Statement

I participated in the broad discussion with all the co-authors in the process of model design and the research work design. I made contribution writing the relevant paper which is under review of IEEE Transactions on Artificial Intelligence (listed above). I also participated in doing the programme, conducted the experiments and run the results for the figures in the tables in this research work.

Chunya ZOU	Prof. Junbin GAO
Signature:	Signature:
Date: 23/03/2023	Date: 23/03/2023

Chapter 1

Introduction & Background

Since the 1950s, machine learning techniques have achieved outstanding success in many different domains over the last few decades. Neural Networks are a subsection under the umbrella of machine learning, while it is also the subfield which originated the Deep Learning at the very beginning. During recent decades, deep learning has become a prevalent field of machine learning, and deep learning techniques have garnered tremendous success in a variety of domains (Alom et al. 2018). Since its inception Deep Learning has been showing remarkable success in almost all the application domains such as image analysis, computer vision, natural language processing and speech recognition (Alom et al. 2018).

Deep learning techniques utilize the hierarchical structures to link the layers of nodes. The output of a lower layer will be feed forward as the input of a higher layer through certain calculations which could be linear or non-linear. These deep learning techniques could transform the features of raw data into the abstract features, and compared to machine learning architectures, deep learning techniques are much better on feature representation, especially on those complex datasets (Bronstein et al., 2017). For example, image and text are usually complex data and they could contain important

personal information, but machine learning technique is not sufficient and able to but to process these kinds of complex data.

Learning approaches based on the representations of the data are usually called representation learning (Alom et al. 2018). While recent research works also demonstrate that Deep Learning based representation learning contains a hierarchy of concepts and features, in which high-level features are determined from the low-level ones. While Deep Learning has been defined as a universal learning method instead of being task specific towards certain kinds of problems in certain fields (Alom et al. 2018).

Meanwhile, with the rapid progress of computer technology such as increasing power of chips processing abilities and then the drastic reduction of the computational cost, and these factors provide strong impetus for deep learning architectures to develop even faster (Charikar et al., 2017). While nowadays, more and more applications on structural data have appeared in this field, such as recommendation system and social network, the deep learning algorithms have progressed a lot to better fulfil the requirements of processing the graph data which is one type of structural data.

1.1 From Euclidean space to Non-Euclidean Geometry

A Euclidean space is a finite dimensional vector space over the reals R, in which the points are designated by coordinates (one for each dimension); while Euclidean space R^n := R*R*…*R (n times), in which the elements R are the vectors with n real components. Euclidean geometry is also known as “plane geometry”, in which the interior angles of a triangle should add up to 180 degree and the shortest distance between two points should always be the straight line between them. And all these

examples are in a two-dimensional flat world, and they are bound by the laws of plane Euclidean geometry, and the data exist in this domain is called Euclidean geometric data. Deep learning architectures have been very successful when it comes to deal with signals like images and speech where the underlying structure is Euclidean or grid-like (*Into the Wild: Machine Learning in Non-Euclidean Spaces* · Stanford DAWN, 2019).

Regarding the deep neural networks' success, one of the leading reasons is that they can leverage data's statistical properties – stationarity, locality and compositionality via local statistics, and these properties have formalized in the convolutional neural networks (CNNs) (Bronstein et al., 2017). For instance, in the task of image analysis we could regard pictures or images as functions in a plane Euclidean space, and sampled on a structure of grid, while Figure 1 clearly shows the procedure of CNN architecture in image classification task. In such condition, we can say that locality is the result of local connectivity, stationarity is because of the shift-invariance and compositionality is owed to the grids' multi-resolution structure property. Those properties mentioned above are all accomplished because of the convolutional structures that consist of alternating convolutional layer and pooling layer (Bronstein et al., 2017).

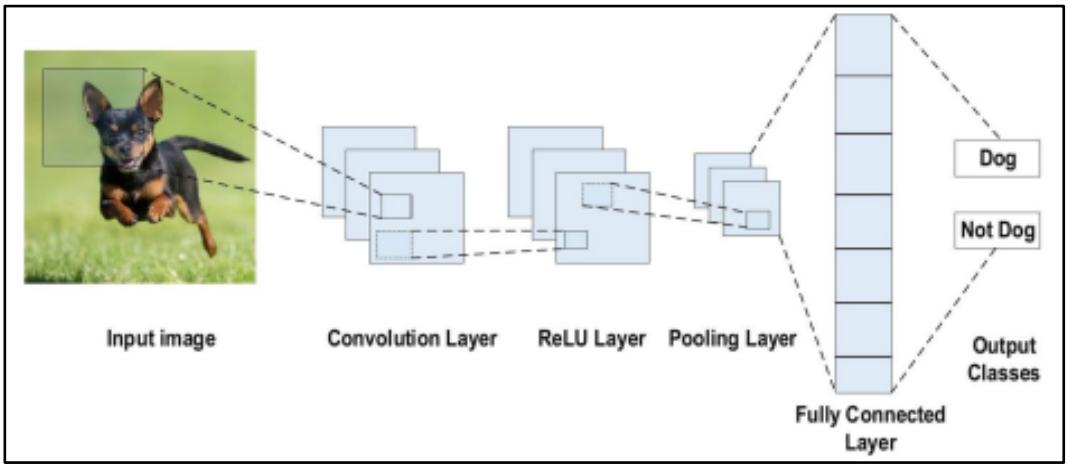


Figure 1 A typical example of CNN structure for the task of image classification (Alzubaidi et al, 2021)

The advantage of utilizing convolutions is that it could extract the local features which usually are shared across the image domain while it could also reduce the parameters' amount without sacrificing the network's capacity (Alzubaidi et al, 2021). However, not all the data could be presented in the format that deep neural networks required, and if we force some other complex data to be contorted into the grid, this will cause us to sacrifice some probably important relationship info in that complex data in favour of much more simple representation that neural networks can take as their input (Alzubaidi et al, 2021).

However, many scientific study data are within a non-Euclidean space, for example, social networks in social science, medical imaging showing brain's functional networks and genetics networks, and these are usually called non-Euclidean geometric data. In many real-world applications, such geometric data are usually very complex, and the scales are very large, for instance, the social network could be on the scale of millions or even billions (Bronstein et al., 2017). Thus, the non-Euclidean nature of complex data like this indicates that they do not have properties such as shift-invariance, global parameterization, coordinate system or vector space structure.

Therefore, convolutional operations which are taken for granted in the Euclidean geometry cannot even be defined correctly in the non-Euclidean cases, which causes the difficulty for deep learning techniques such as CNNs to process and deal with such complex data. Extending deep neural network architectures to the non-Euclidean domain could be referred to as geometric deep learning, recently more and more attention has been drawn into the application of deep learning techniques on non-Euclidean geometric data (Bronstein et al., 2017). For instance, the sensor networks are actually graph models of the distributed inter-connected sensors, and their reading are regarded as time-dependent signals on the sensors (nodes); while the social networks

can be considered as social graph and the characteristics of the users (nodes) in the social graph can be modelled as signals on these vertices.

1.2 Graph Data

Graphs has many different complex types according to their connections and information on nodes and edges, in this section, different categorizations will be illustrated in more details.

1.2.1 Directed and Undirected Graph

While the edges in between the nodes could be undirected or directed by associating directionality to additionally specialize graphs and the directionality is also an important information attached with the graph. In the case of undirected graphs, the adjacency matrix A is symmetric, but when it comes to directed graph, the adjacency matrix A is asymmetric as the matrix size is not square size of |v|*|v| anymore, instead the size might be |n|*|m| or |x|*|y|. More details will be illustrated in more details in the following sections.

Figure 2 Directed and Undirected Graph

There have been many types of graphs in the real world such as citation networks, while some kinds of data that we might not think could be graph-structured data in the first place. For example, text can be regarded as graphs, we actually can digitize the texts via associating indices to each token or word and representing the sentence or text as a sequence of these indices. This could generate a quite simple directed graph, in which

each index is a node and is connected by the edge to the following index, just like the Figure 3 shown below. And its adjacency matrix for such text is a diagonal line as each token could only connect to its prior token and the following one.

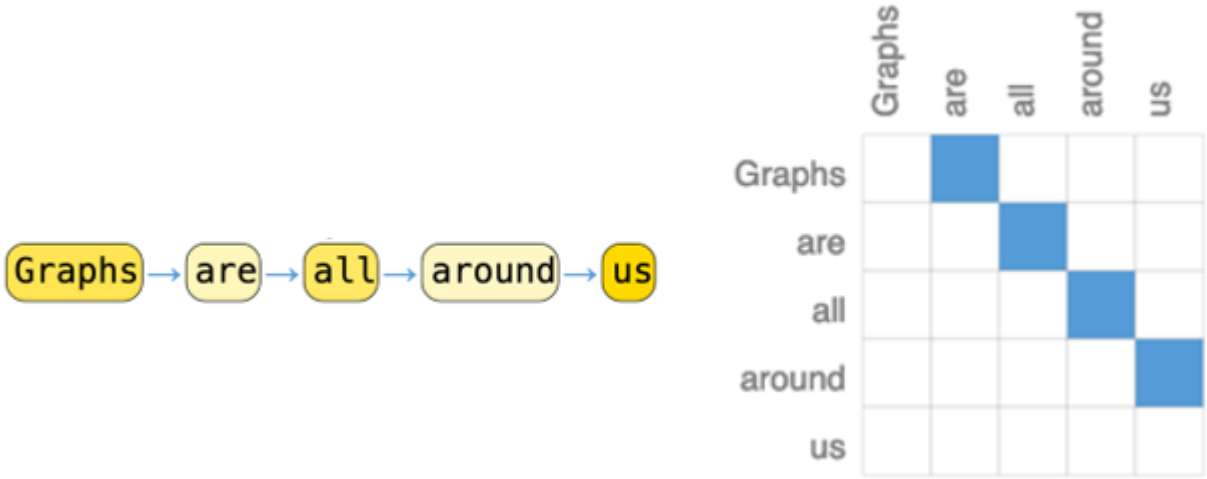


Figure 3 A sentence (text) and its diagonal adjacency matrix

1.2.2 Heterogenous and Homogeneous Graph

There are mainly two types of graphs under the umbrella of graph data, heterogeneous graphs and homogeneous graphs. A graph with just one single type of node and a single kind of edge is called homogeneous (Hu et al. 2019). An example of such homogeneous graph would be the Facebook social network, in which the nodes representing the individuals and edges representing the friendship between two individuals connected with each other. On the other hand, a heterogeneous graph can have nodes and edges that are different types. For instance, in the recommendation system, nodes could be products and customers, while edges could represent somebody buying something or someone returning something.

Meanwhile the nodes and edges in the graph also could incorporate properties which can be named as features or attributes. For example, in the case of Facebook social media, a person node could have many features and attributes such as this person's age,

location, university or high school they attended and hobbies, while an edge between two person nodes could have an attribute of date establishing the date that they added each other as friends.

1.2.3 Dynamic Graphs & Static Graphs

A graph is a kind of structure of data that comprise two parts: nodes and edges. A graph could be expressed as $G = (V, E)$, in which V represents nodes while E represents those links/edges in between nodes. Officially, a weighted and undirected graph G could be written as a triple $G = (V, E, A)$, in which the extra A is the adjacency matrix recording relationships between nodes. While the matrix A should be a square matrix size of |v|*|v|. Information could be stored in each node, or each edge or the entire graph, where the nodes symbolise entities in the data such as the members in the social network and the edges represent the relationships between these entities. But in static graph, time information is not considered carefully (Rossi et al. 2020).

Thus, when input features of the graph data vary with time, the graph is regarded as dynamic graph. There are two kinds of dynamic graphs, Discrerte-Time Dynamic Graph (DTDG) and Continuous-Time Dynamic Graph (CTDG). They both are observed on a timely manner and an update will occur to the existing graph, the difference is that DTDG is observed at regular intervals and generates a sequence of snapshots of the graph and the changes or event happened during the intervals are not recorded (Rossi et al. 2020). On the contrary, CTDG observes and records each change in the graph individually with its timestamp. The types of events or changes could be deletion of node or edge, creation of node and edge as well we the feature change of nodes and edges (Rossi et al. 2020). Thus, this consecutively recording activity allows the CTDG to track down the complete evolution of the graph, and it leads to the

minimal information loss during the whole evolution process. However, it is obvious that it is generally more complicated and difficult to develop models for CTDG as these models need to incrementally and efficiently incorporate new events and changes at test time and handle all different kinds of events (Rossi et al. 2020).

1.3 Types of Deep Learning Approaches

As deep learning methods could be classified into many different types: supervised, semi-supervised, and unsupervised, Additionally, there is also another section of learning method which is called Deep Reinforcement Learning (DRL) and this structure will also be introduced briefly later and this approach is sometimes under the scope of unsupervised or semi-supervised learning approach (Alom et al. 2018).

1.3.1 Supervised Learning

Supervised learning is one type of learning methods and it utilizes labelled data. When it comes to the case of supervised deep learning methods, the environment contains a group of inputs and corresponding outputs $(x_t, y_t) \sim \rho$. In particular, after processing the inputs x_t, the intelligent agent will get a prediction result $\hat{y_t} = f(x_t)$, then the agent will receive a loss value of $l(y_t, \hat{y_t})$ between the ground truth label and predicted label. Afterwards, the agent will train the model and iteratively modify the parameters of the network to get better approximation for the desired outputs (Alom et al. 2018). After appropriate modification of the parameters, then the agent will get the best answers which is the closest one to the true answer to the problem. There are many kinds of supervised learning methods under Deep Learning structures, including Deep Neural Networks (DNN), Recurrent Neural Networks (RNN) such as Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN) (Alom et al. 2018). While

LSTM will be further explained in Section 1.6.2.3 and CNN will be described with more detail in Section 1.6.2.1.

1.3.2 Semi-supervised Learning

Semi-supervised learning is a learning process that is based on the partially labelled datasets, and it is often called as reinforcement learning. In some cases, reinforcement learning (Deep Reinforcement Learning) and Generative Adversarial Networks (GAN) are applied as semi-supervised learning methods. While RNN structures such as LSTM and GRU could also be utilized for semi-supervised learning (Alom et al. 2018). Generative Adversarial Networks (GAN) will be further described in Section 1.6.2.2.

1.3.3 Unsupervised Learning

Unsupervised learning is a learning type that can do the learning without the true labels of data. Thus, the agent will need to learn the important features or internal representation to find out the hidden relationships or latent structures among the input data. In the normal cases, either clustering, dimensionality reduction or generative techniques could be applied as a unsupervised learning method (Alom et al. 2018). Under the scope of deep learning structures, there are many networks which have advantages in non-linear dimensionality reduction and clustering, for example, Auto Encoders (AE) and Generative Adversarial Networks (GAN).

1.3.4 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning is a relatively recently proposed but outstanding learning technique to be utilized in the unknown environments. Deep Reinforcement Learning began in 2013 and several advanced structures have been introduced and proposed based on the technique of Reinforcement Learning (RL) since then (Alom et al. 2018).

In Reinforcement Learning, there is no straight forward loss function which makes the learning process even more complicated compared to the common supervised methods. The basic differences between reinforcement learning and supervised learning are: first of all, there is no full access to the functions that need to be optimized during the training process, and these functions need to be queried via interaction; secondly, the environment is state-based, which means that input x_t depends on the previous actions and it updates its action as this process goes on (Alom et al. 2018).

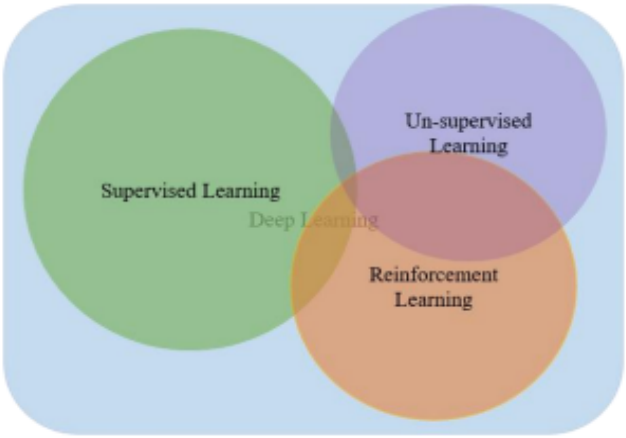


Figure 4 Category of Deep Learning Methods (Alom et al. 2018)

1.4 Learning Tasks for Graph Data

This section will provide a brief information about three most common and major representation learning tasks for the graph data.

1.4.1 Node Classification

This is a node-level learning task, inferring some nodes' incomplete attributes while given other neighbourhood nodes' features values and structures in that network. The main purpose here is to find out the best representation for each graph node so that they could be further processed for the labelling phase, then put into the neural network models for further learning and classification (Asif et al., 2021). And this is a graph based supervised or semi-supervised learning problem, as the model is trained on a

subset of nodes with true labels and then make prediction of those target nodes (Asif et al., 2021).

1.4.2 Link Prediction

This is an edge-level learning task, inferring missing relationship or finding out the hidden relationships in between the nodes in the network. Similar with the node classification problems, but the difference is that the subject is to assign labels to edges instead of nodes. The model will utilize the hidden representation of node pairs and then get the corresponding likelihood result of the link existence according to the nodes' similarity scores (Asif et al., 2021). While deep neural networks models usually integrate the representations of node pairs which are learned from the input data altogether and then regard the link prediction problem as a binary classification problem (Asif et al., 2021). Similar with the node classification, link prediction is also a semi-supervised learning.

1.4.3 Graph Classification

This is a graph-level learning task, discriminating problem between graphs of different classes. As a dataset could have multiple or hundreds of graphs, and each of these graphs could be considered as instances to be assigned labels, and main task here is to find out a low-dimensional representation of each graph from the graph data and the output embedding of the graph will be passed to the readout layers such as fully connected network after the graph pooling phase (Asif et al., 2021).

The node classification and link prediction learning tasks will be covered in more details with empirical studies in the later chapters.

1.5 Feature Learning

The main difference between traditional machine learning methods and deep learning methods is the techniques they apply to extract features from the input data. The traditional machine learning methods utilize manually made features through applying feature extraction algorithms such as Local Binary Pattern (LBP). Then the machine learning algorithms including Random Forest (RF), Principal Component Analysis (PCA), Support Vector Machine (SVM) and many other structures are considered to be utilized for the classification task using those extracted features (Alom et al. 2018). In addition, boosting techniques such as AdaBoost and XGBoost are frequently implemented where several machine learning algorithms are applied on the features of the input data and the final output and decision is computed based on several outcomes from all the algorithms (Alom et al. 2018).

While among the Deep Learning methods, features are usually learned automatically at the same time features' representations are hierarchical in multiple levels, which is a main advantage of the deep learning techniques compared to the traditional machine learning methods (Alom et al. 2018).

1.6 Deep Learning Structures

In this section, a brief introduction of the history of graph neural network will be illustrated. Since 2006 when graph theory has gradually come in close contact with the machine learning techniques and then a new concept of Graph Neural Network architecture emerged (Li et al., 2021). The very first proposal regarding GNN appeared in 2006 by Scarselli and Gori, and then in 2008 the paper called "The Graph Neural Network Model" was subsequently published, whose authors laid the basic

mathematical foundations for the modern graph neural network afterwards (Li et al., 2021).

The proposed graph neural network consists of recursive neural networks and Markov chains, which both are commonly applied to graph problems as the recursive neural networks are neural network architectures whose input domain are usually directed acyclic graphs and they map a graph to a vector of reals but the graph needs to go through a pre-processing phase to make sure the model could handle different types of cyclic graphs (Li et al., 2021). While recursive neural networks are also similar to support vector machines (SVM) as they both utilize special kernels to process graph structured data and encode the input graph as a representation.

Meanwhile, Markov chain model could imitate the processes in which the causal relationships among events are represented by the graphs, and the random walk theory also helps Markov chain models to be applied successfully to the web page ranking algorithm (Li et al., 2021). And this algorithm was exploited by internet search engines such as Google, to measure the relativity of the web pages. The paper "The Graph Neural Network Model" also extends these two techniques to make the model be able to directly deal with graph structured information and unifies them into a common framework (Li et al., 2021). And this proposed architecture could process a more general class of graph data such as undirected and cyclic graphs and can deal with node-focused applications without any pre-processing phase (Li et al., 2021). Since advent of this milestone paper, there has been many amazing spikes in the graph deep learning world.

1.6.1 Convolutional Neural Network in Euclidean Domain

In LeCun et al. (1989) paper, researchers developed a convolutional neural network which is designed for the handwritten zip code recognition and they utilize word "convolution" for the very first time. Convolutional neural network is a type of feed forward neural network which could extract multi-scale localized spatial features from the dataset with a convolutional structure (Zhou et al., 2020). The main advantages that CNN possesses are as follows,

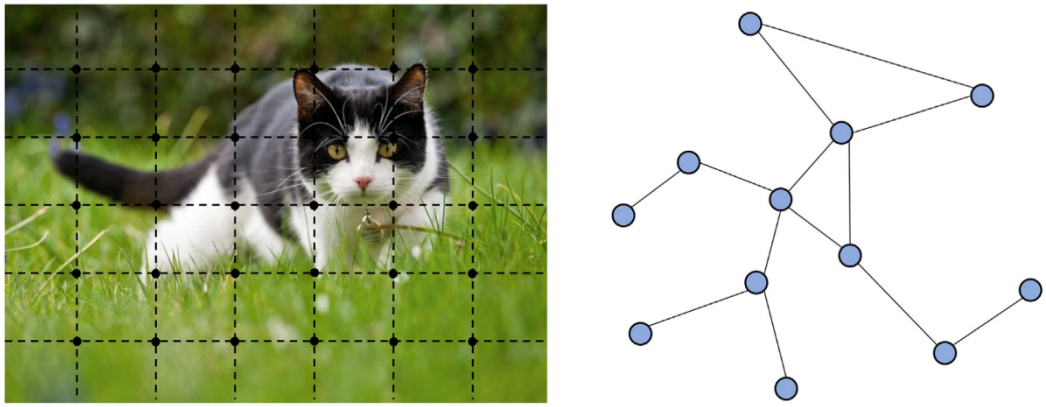
1) Shared weights. A same group of connected nodes could share the same weights, which could reduce the number of parameters.
2) Local connections. Every neuron is not linked to all the neurons from its previous layers anymore, while this could help reduce the parameters as well and also speed up the convergence.
3) Down-sampling dimensionality reduction (Zhou et al., 2020). This is a pooling layer that utilize the principle of an image's local correlation to down-sample this image, and this process could also reduce the data amount at the same time keeping the useful information and reduce the parameters via removing those very trivial features.

The three characteristics listed above make convolutional neural network (CNN) turn into one of the most influential and typical algorithms in the deep learning field for Euclidean data (Zhou et al., 2020).

However, CNNs is only able to operate on regular Euclidean data like texts and images while these data could also be considered as graphs. When the graph is in non-Euclidean space (Figure 5), it is quite difficult for CNN to define localized convolutional filters and the pooling operators, and this poses an obstacle in the

transformation of convolutional neural network from the Euclidean domain to non-Euclidean domain (Zhou et al., 2020).

Figure 5 Left: Image in Euclidean Space; Right: Graph in Non-Euclidean Space (Zhou et al., 2020)



1.6.2 Neural Network on Non-Euclidean Space

This section will present and explain several commonly studied neural networks utilized for graph exploitation. Generative models and discriminative models and the sequential model will all be discussed for better understanding of these common graph-based neural architectures.

Graph data is known to be relatively complicated compared to other data structures such as time series, that is because graph data can be irregular and have a different size of nodes in varying orders, thus, it hinders the application of machine learning architectures to the graph domain such as the operation of convolutions (Wu et al., 2020). Moreover, there is a significant assumption when applying many existing machine learning models and that is all the nodes are independent from each other; however, when it comes to graph data, this assumption cannot hold any more because the nodes are related to each other via the edges (Wu et al., 2020).

Therefore, to better address the complexity issue of graph data and catch the hidden information from the graph data, graph neural network (GNN) has gradually become an effective method for graph learning tasks as the underlying principles of GNN is to transform the complex graph-structured data into another space which is low dimensional, and at the same time retain the structural information (Wu et al., 2020).

1.6.2.1 Convolutional Neural Network

The convolutional operation is also utilized in the graph neural network and nowadays modern computer vision architectures use convolutional neural network or convolution operation to learn image patches' complicated features (Asif et al., 2021). Standard CNN uses fully connected layers via transforming the output into a simple single dimension. As for convolutional neural network, every pixel in the image is considered as the input unit in the input layer whose size is n_1*1 where n_1 represents the amount of input channels, and then trough the t kernels, the input vector with the size of k_1*1 is filtered via convolutional layer in the hidden layer (Asif et al., 2021). And the convolutional layer activation layer could be represented as follows:

$$y^i(p) = \max\left(0, b^{j(p)} + \sum_j k^{ij(p)} * x^{i(p)}\right)$$

In which $x^{i(p)}$ is defined as the ith input and $y^{j(p)}$ is defined as the j^{th} output activation, while $b^{j(p)}$ represents the the jth output's bias and the * denotes the convolution, at the same time $k^{ij(p)}$ represents the convolution kernel between the input and output layer (Asif et al., 2021).

1.6.2.2 Generative Models

In the machine learning domain, there are two highly appreciated approaches – generative learning and discriminating learning. There are many structures have been

proposed so far such as generative adversarial network, auto-regressive networks, variational autoencoder and Markov models (Asif et al., 2021). While these architectures are also very applied in many real-life situations. Generative architectures are utilized to better process graph structure and among the existing generative models, Generative Adversarial Networks (GAN) has become very popular because of its adversarial training process, and the Figure 6 below illustrates the vanilla adversarial network process (Asif et al., 2021).

At the beginning, Goodfellow et al. (2020) designed and proposed F_c layers for discriminator P_D as well as Generator G. While the generator G is designed to fool the discriminator via developing the fake input samples and generates a distribution P_d on the true data which is denoted as X, which mean that G will generate the synthetic data through an adversarial training process and make the fake samples as real as the distribution of the original real data (Asif et al., 2021). The objective function of G is expressed as follows: $^{min}_{G}E_{Z\sim P_z}[\log\left(1-P_d\left(G(z)\right)\right)]$, in which $P_d(x)$ represents the probability which the data's possible distribution is from the true data instead of the generated fake sample data. The equation is maximized when P_d is correct and it is minimized when P_d is wrong (Asif et al., 2021). The purpose of P_d is to enhance the accuracy rate of the classification to better distinguish the fake synthetic data from the real data (Asif et al., 2021). The objective function for P_d is as follows (Asif et al., 2021):

$$^{max}_{D}E_{X\sim P_{data}}[\log P_d(\mathrm{x})] + E_{Z\sim P_z}[\log\left(1-P_d\left(G(z)\right)\right)],$$

Therefore, the combined function for the whole generative architecture follows that min-max theory which could be defined as follow (Asif et al., 2021):

$$\max_D E_{X\sim P_{data}}[\log P_d(\mathrm{x})] + E_{Z\sim P_z}[\log\left(1 - P_d\left(G(z)\right)\right)]$$

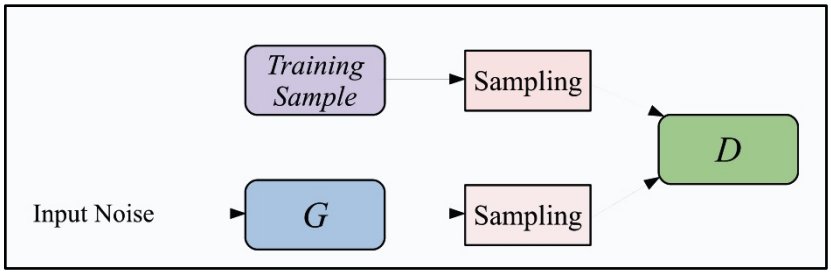


Figure 6 Demonstration of GAN structure. G represents generator which helps sample fake data to D, where D defines the discriminator which calculates the probability regarding if this sample data is fake or real.

1.6.2.3 Long Short-Term Memory Architecture

The paper Hochreiter & Schmidhuberet (2017) first proposed the Long Short-Term Memory (LSTM) architecture which could reduce gradient vanishing issue from the normal Recurrent Neural Network (RNN). The Long Short-Term Memory (LSTM) model is comprised of recurrent networks in which each neuron/node of the hidden layers will be interchanged through those memory cells (Asif et al., 2021). While LSTM's each memory cell architecture consists of self-connected recurrent edge which normally has a fixed weight. And the property mentioned above helps gradients flow through the networks successfully instead of vanishing during the process because of the little weights they have (Asif et al., 2021). Long Short-Term Memory (LSTM) model leverages long-term memory storage for a certain period in terms of short-term activations (Asif et al., 2021).

The input sequence $x^{(t)}$ sends the input activation into input node when it is at present time step, then present time step will be further prepared according to its previous time hidden state which is denoted as $h^{(t-1)}$. Usually, the weighted sum is determined utilizing that *tanh* activation function in hidden layer, while the paper Hochreiter &

Schmidhuber (1997) uses the sigmoid activation function. Every memory cell is lined by the linear activation function which could represented by $s^{(t)}$ (Asif et al., 2021). While there are self-connected recurrent cells which are mentioned above, and they hold the internal state, whose edge circulates based on the time steps with the fixed value which is also weighted, and it could help avoid the explosion of gradient (Asif et al., 2021). While in Gers & Schmidhuber (2000) paper, the forget gate is introduced to help illustrate the method to remove the unimportant internal state part, which could allow the gradient to pass through the network more smoothly.

During the phase of forward gradient propagation, the internal state is in control of the gradient via the activation function. If both output cell and internal cell are lcoked and closed, the activation will be detained completely in the memory cell and there will be no changes towards the intermediate time steps (Asif et al., 2021). While in the backward gradient propagation phase, the frequent error issue leads the structure to backpropagate based on time steps. Multiple memory cells could help enhance the Long Short-Term Memory (LSTM) model's ability to learn more dependency information from the input sequence (Asif et al., 2021).

It is obvious that LSTM network solves the problem of vanishing gradients in the existing recurrent neural networks, and it is also proved to be a robust sequential architecture. LSTM has been utilized and applied in many different industrial domains such as speech recognition, natural language processing and medical imaging (Asif et al., 2021). Among the modern deep learning techniques, LSTM is utilized to capture the long-term dependency information or the spatio-temporal information (Asif et al., 2021). Therefore, LSTM model could better leverage the long-term information for the following learning processes which could eventually increase the accuracy of the learning tasks. However, it required much more computational capacity and higher

memory capability to do the task comparing to a normal recurrent neural network (Asif et al., 2021).

1.7 Thesis Outline and Contribution

There have been large number of literatures in graph-based deep learning methods, most of them only can perform on homogeneous graphs with only one type of entity or one kind of relationship, and there are some pioneer research works on dealing with directed graph data and architectures proposed in those works have become benchmarks.

At the beginning, inspired by the paper (Song et al., 2019), in which researchers proved that the proposed tensor singular value decomposition (SVD) is very effective to help achieve better performances of recovery regarding robust tensor completion issue, we think that it could be worth to try to apply SVD into graph convolutional network and see whether the new model could increase the accuracy rate when dealing with directed graph's learning tasks, though there have been some outstanding benchmark architectures which already achieved relatively high accuracy rate in both node classification and link prediction tasks.

After trying to conduct convolutions on the spectral domain which was generated by the singular value decomposition (SVD) of the directed adjacency matrix, we implemented it in coding and propose and denote this model as SVD-GCN, and the preliminary results from the experiments showed that SVD-GCN performs better on directed graph on both node classification task and link prediction task compared to most of the state-of-the-art models' results. We also leverage the graph framelets (Zheng et al., 2021) and quasi-framelets (Yang et al., 2022) for the multi-resolution analysis on the directed graph data because they could generate better model for high-

pass as well as low-pass information. Thus, this proposed SVD-GCN architecture is the main contribution of this thesis, its application on link prediction task will also be presented and explain in the Chapter 3.

To help better understand the contents, Chapter 2 will provide a more comprehensive background introduction for the research topic and will help readers have a better overview in this domain. Then Chapter 3 will present and explain in detail about the proposed SVD-GCN architecture and its performance in node classification and link prediction with all the experiments' results and theorems. To close the thesis, Chapter 4 will also summarize the contributions and discuss some potential follow-up works and potential directions for future research.

Chapter 2

Literature Review

This section will present a more comprehensive background in graph-cased deep learning on graph-structured data with more literatures.

2.1 Graph Neural Network on Directed Graph Data

Neural Network was first time applied on direction graph data in the paper Sperduti & Starita (1997), while this work handles the structured patterns that are represented as directed labelled graphs and proposed a generalization of recurrent neurons which could better represent the structured patterns than other statistical architectures. This research work constructed the foundation for graph neural network (GNN) structures application on the digraph data, and it motivated some early studies in this direction (Wu et al., 2020).

The early research works focused a lot on the application of Recurrent Graph Neural Network (RecGNN) on digraph data, addressing issues such as the correlated maximum outdegree limit and the positional constraint in an image classification task (Sperduti & Starita, 1997). The experimental results also prove that the new RNN architecture could better handle directed acyclic graphs (DAGs). Recently, motivated by the successful application of convolutional neural networks (CNNs) in computer

vision, researchers developed new generalizations of operations and re-formulate the concepts of graph convolutions to better handle the complex graph data, and these new methods are generally called Convolutional Graph Neural Network (ConvGNN) (Wu et al., 2020). Quite different from the RecGNNs' iterating node states with contractive constraints, ConvGNNs mainly stacks multiple graph convolutional layers with variable weights of each layer and develop the corresponding node representations or edge representation based on the different learning tasks (Wu et al., 2020).

There are two major classes of convolutional graph neural networks (ConvGNN), spectral-based and spatial-based approaches. For spatial-based approaches, they define the graph convolutions by message passing or feature aggregation from each node's neighbourhood nodes; while for spectral-based approaches, they apply convolution in the graph Fourier domain by eigen-decomposition (EVD) from the graph Laplacian (Zou et al., 2022), which are motivated by the concept of "filter" from the graph signal processing perspective. Both approaches could be applied to the directed graph data (Digraph), but compared to the spectral-based approaches, spatial-based methods are less favoured since they are not able to extract information at different frequencies. As explained in (Balcilar et al., 2021; Nt & Maehara, 2019), many spatial-based models have low-pass filters, which will fail to capture the high-frequency information while the information could be very useful in the learning tasks.

Furthermore, it is not easy to generalize the spectral methods to digraph data because the asymmetric adjacency and Laplacian matrices cannot generate the orthonormal systems for the signal decomposition (Tong et al. 2020). If we ignore the direction information in the directed graph data and by only performing convolutions on the symmetric Laplacian or the Adjacency matrix, it will cause a very severe loss of information and the models' performance in the learning tasks will be even worse. In

Ma et al. (2019) work, the convolution on the directed graph is generated on a symmetric Laplacian that is developed through the transition probability matrix. Tong et al. (2020) research work extends this idea in Ma et al. (2019) that only works for graphs which are strongly connected to the general graphs by adding a small teleport probability in the transition matrix. Meanwhile, this work also improved the network's overall performance via generating the scalable receptive fields based on the idea of inception module in (Szegedy et al., 2016). And the corresponding experimental results also proves that this specialized model's efficacy is better many other spatial-based approaches. In recent year, spatial-based ConvGNNs have developed very fast because of their high efficiency and flexibility and most of the existing ConvGNNs architectures are spatial-based.

2.2 Spectral-based ConvGNNs

Spectral-based methods depend on the spectral graph theory. In this network, the graph signals are filtered via eigen-decomposition of graph Laplacian (Balcilar et al., 2020), which is determined by $L = D - A$ while the normalized graph Laplacian is determined by $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, in which A is adjacency matrix, I is the identity matrix, $D \in \mathbb{R}^{n_k \times n_k}$ represents the diagonal degree matrix with $D_{i,j} = \sum_j A_{j,i}$. It could be decomposed into $L = U \Lambda U^T$ where U is the eigenvectors matrix in the order based on the eigenvalues as this Laplacian is positive semidefinite, while Λ denotes diagonal matrix of eigenvalues, while $\Lambda_{ii} = \lambda_i$ where λ denotes positive eigenvalues' vectors.

The eigenvectors from the normalized Laplacian matrix construct an orthonormal space and it can be denoted mathematically as $U^T U = I$. While the graph Fourier convert to a signal x from graph data can be denoted as $F(x) = U^T$ while the inverse version of

this graph Fourier transform could be written as $F^{-1}(\hat{x}) = U^T\hat{x}$, in which $\hat{x}$ represents the final result from the signal through the transform of the graph Fourier. The overall procedure is that the graph Fourier transform first coverts the input signal from the graph into the orthonormal space in which foundation is built utilizing the normalized graph Laplacian's eigenvectors. Meanwhile the instances in this transformed signal $\hat{x}$ represents the graph signal's new coordinates in a new space, while the input signal is denoted as shown here: $x = \sum_i \hat{x}_i u_i$, that is also representation of the inverse version of graph Fourier transform. Then input signal x graph convolution can be written and expressed as follow,

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

in which $\odot$ represents the element-wise product, and filter could be denoted as $g_\theta = \mathrm{diag}(\mathrm{U}^T g)$, so the spectral graph convolution could be written as follows,

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

The equation above is the main principle of all the spectral-based ConvGNNs. While the difference among all different spectral ConvGNNs architectures is the different choice of the filter g_θ.

Spectral Convolutional Neural Network holds the assumption that the filter $g_\theta = \Theta_{i.j}^{(l)}$ is a large group of learnable parameters and the filters are able to recognize the graph signals via the multiple channels (Wu et al., 2019). Meanwhile in the Spectral ConvGNN, a graph convolution layer could be a concatenation of all filtered signals with the activation function σ , for example, ReLU (Rectified Linear Unit),

$$H_i^{(l+1)} = \sigma\left(\sum_{i=1}^{f_l} U\, \Theta_{i.j}^{(l)}\, U^T H_i^{(l)}\right)$$

where for all l is the layer index, f_l denote input channels' number while f_{l+1} represents output channels' number, and $j \in \{1, \dots, f_{l+1}\}$, $H^{(l)} \in \mathbb{R}^{n \times f_l}$ represents the input graph signal, while $\Theta_{i.j}^{(k)}$ represents a diagonal matrix containing learnable parameters. However, since spectral ConvGNNs utilizes eigen-decomposition of Laplacian matrix, spectral ConvGNNs have three main disadvantages. Firstly, any perturbation on a graph will lead to a change in the eigen-basis. Second of all, the learned filters are mainly based on the domain, and this means that these learned filters are specifically unique to each graph and cannot be generalized and implemented to a different-structured graph. Thirdly, eigen-decomposition needs computational complexity of $O(n^3)$ and this workload is very heavy (Wu et al., 2019). There are other research works that have proposed new structures such as GCN and ChebNet (Kipf & Welling, 2016) which decrease the computational complexity to $O(m)$ via applying some approximations and simplification techniques.

Chebyshev spectral ConvGNN (ChebNet) approximate graph filter g_θ via the Chebyshev polynomials of the eigenvalues' diagonal matrix. The ChebNet filters are localized in the local space and this is quite different from the normal spectral ConvGNNs because it proves that filters in this work could extract important local features in the graph size independently (Wu et al., 2019).

Graph Convolutional Network (GCN) proposes the first-order approximation of ChebNet while GCN as a kind of spectral-based method, can be regarded as a spatial-based approach as well. Because from the perspective of a spatial-based approach, GCN could also be regarded as putting all the feature information all together according to the target node's neighbourhood nodes. While several papers have explored the alternative symmetric matrices to further improve this deep learning architecture.

Adaptive Graph Convolutional Network (AGCN) (Li et al., 2018) learns the hidden structural relations in the graph data via the graph adjacency matrix while it also generates an adjacency matrix for the graph residual utilizing a learnable distance function, that is able to regard two target nodes' features as the inputs.

Dual Graph Convolutional Network (DGCN) (Zhuang & Ma, 2018) introduces and proposes a dual graph convolutional structure, containing two layers of graph convolution in parallel, at the same time these two layers share the same parameters. While the two graph convolutional layers utilize the normalized adjacency matrix as well as the positive pointwise mutual information (PPMI) matrix, that can help capture the latent information about nodes' co-occurrence via the random walks sampled from the original graph data. By ensemble outputs from the dual graph convolutional layers, Dual Graph Convolutional Network could encode both local and global structural information instead of stacking multiple graph convolutional layers (Wu et al., 2019).

2.3 Spatial-based ConvGNNs

Spatial-based ConvGNNs develop graph convolutions according to the target node's spatial relations, and this is very similar with the convolutional operation from the normal CNN structure on the classification task for image analysis. Because images could be regarded as graph data as well where each pixel could be represented as a node and each pixel is linked to its neighbourhood nodes. A filter could be applied (add picture) to a 3*3 patch via taking the pixel value's weighted average of the target node and its neighbours over all other channels (Wu et al., 2019). Likewise, the spatial-based graph convolutions exploit that target nodes' representation and its neighbour nodes' representations to generate target node's new representation. The spatial graph

convolutional operation passes the nodes' message via the links between the nodes (Wu et al., 2019) and spatial ConvGNNs share the similar information propagation concept in Recurrent GNNs.

Diffusion Convolutional Neural Network (DCNN) (Atwood & Towsley, 2016) considers convolution operation on the graph as a process of diffusion, in which it assumes that information is propagated from one node to one of its neighbour nodes with a transition probability and then the information distribution will be able to reach equilibrium after several rounds of information propagation. DCNN's diffusion graph convolution is defined as follows,

$$H^{(k)} = f\left(W^{(k)} \odot P^k X\right),$$

In which $f(\cdot)$ denotes an activation function while $P \in \mathbb{R}^{n \times n}$ is probability transition matrix and it is calculated via the formula $P = D^{-1}A$. In Diffusion Convolutional Neural Network model, the hidden representation matrix $H^{(k)}$ has the same dimension as the input feature matrix X. Then DCNN concatenates all the $H^{(1)}, H^{(2)}, H^{(3)}, \dots, H^{(k)}$ as the final output. Because a diffusion process's stationary distribution is adding all the probability transition matrices' power series, Diffusion Graph Convolution (DGC) (Li et al., 2017) gathers all the output results during every round of diffusion process to replace the concatenation operation. The diffusion graph convolution is written as follow,

$$H = \sum_{k=0}^{k} f\left(W^{(k)} P^k X\right)$$

in which $W^{(k)} \in \mathbb{R}^{D \times F}$ and $f(\cdot)$ represents the activation function. While the power of a transition probability matrix could help the very distant neighbour node still pass information onto the central node. PGC-DGCNN (Tran et al., 2018) improve the

contributions of distant neighbour nodes according to the shortest path which is defined by the shortest path adjacency matrix $S^{(j)}$. Consider that the shortest path from a node v to another node u is of length j, then $S_{v,u}^{(j)} = 1$ or it will be 0. There is a hyperparameter denoted as r to help regulate size of that receptive field. While PGC-DGCNN defines the corresponding convolutional operation as follow,

$$H^{(k)} = \|_{j=0}^{r} f\left((\widetilde{D}^{(j)})^{-1} S^{(j)} H^{(k-1)} W^{(j,k)}\right),$$

in which $\|$ represents the concatenation of the vectors, and $\widetilde{D}_{ii}^{(j)} = \sum_l S_{i,l}^{(j)}$, $H^{(0)} = X$. However, the computation process of this shortest path adjacency matrix is very expensive and the computational complexity of $O(n^3)$ at maximum. While Partition Graph Convolution (PGC) (Yan et al., 2018) partition all the neighbourhood nodes of the target node into P groups. Then PGC generates P adjacency matrices on the defined neighbourhood group. While PGC utilizes GCN (Kipf & Welling, 2016) and applies a different matrix on each neighbourhood group and gather all the results,

$$H^{(k)} = \sum_{j=1}^{P} \bar{A}^{(j)} H^{(k-1)} W^{(j,k)}$$

where $\bar{A}^{(j)} = (\widetilde{D}^{(j)})^{-\frac{1}{2}} \tilde{A}^{(j)} (\widetilde{D}^{(j)})^{-\frac{1}{2}}$, $\tilde{A}^{(j)} = A^{(j)} + I$, $H^{(0)} = X$.

Message Passing Neural Network (MPNN) (Gilmer et al., 2017) proposed a spatial-based ConvGNN framework, which considers the graph convolution as a message passing step where all the information could be propagated from one node to another through that edge between them directly. While MPNN runs the k-step iterations of information propagation and this message passing function (spatial graph convolution) could be written as follows,

$$h_v^{(k)} = U_k(h_v^{(k-1)}, \sum_{u \in N(v)} M_k(h_v^{(k-1)}, h_u^{(k-1)}, x_{vu}^e))$$

In which $U_k(\cdot)$ and $M_k(\cdot)$ represents the functions with learnable parameters and $h_v^{(0)} = x_v$. After generating the hidden representation for each node, $h_v^{(k)}$ could be passed to the output layer and perform a readout function to do a node-level or graph-level learning task. While the readout function could construct a representation of the entire graph data according to the nodes' hidden representations and this could be written and expressed as: $h_G = R(h_v^{(K)} | v \in G$, in which $R(\cdot)$ denotes the readout function with parameters. And MPNN actually covers many existing GNNs, but the difference is that its forms of $U_k(\cdot), M_k(\cdot)$ and $R(\cdot)$ is different.

However, Graph Isomorphism Network (GIN) (Xu et al., 2018) realized that the MPNN-based architectures are not able to distinguish different graph structures based on the graph embeddings previously generated. To solve the problem, GIN could adjust the weight of the target node (central node) via a learnable parameter $\epsilon^{(k)}$. The graph convolution is GIN could be written as follow,

$$h_v^{(k)} = MLP((1 + \epsilon^{(k)}) h_v^{(k-1)} + \sum_{u \in N(v)} h_u^{(k-1)})$$

Where $MLP(\cdot)$ denotes the multi-layer perceptron. In the real-world cases, the number of a node's neighbours are usually different which could be a hundred or a million (Xu et al., 2018). So it might not be efficient to take the full size of the neighbourhood node when calculating and constructing the graph convolution. GraphSage utilizes random sampling to retain a fixed number of neighbour nodes for each target node and the graph convolution could be written as follow,

$$h_v^{(k)} = \sigma(W^{(k)} \cdot f_k(h_v^{(k-1)}, \{h_u^{(k-1)}, \forall u \in S_{N(v)}\}))$$

Where $h_v^{(0)} = x_v$, $S_{N(v)}$ represents the random sample from the target node v's neighbours, $f_k(\cdot)$ is the aggregation function which is usually invariant to the permutations from the node orderings, such as a mean function (Xu et al., 2018).

Graph Attention Network (GAT) (Velickovic et al., 2017) is also a very popular graph-based deep learning structure recently. It holds the assumption that the contributions from the neighbour nodes to the target node are neither pre-determined like GCN (Kipf & Welling, 2016), nor identical like in GraphSage (Hamilton et al., 2017). GAT applied the attention mechanism to learn the relative weights between two connected nodes in the graph. The graph convolution layer in GAT is denoted as follows,

$$h_v^{(k)} = \sigma\Big(\sum_{u \in N(v) \cup v} \alpha_{vu}^{(k)} W^{(k)} h_u^{(k-1)}\Big)$$

Where $h_v^{(0)} = x_v$, and the attention weight $\alpha_{vu}^{(k)}$ calculate the strength of the connectivity between the node v and its neighbour u and the corresponding formula is shown below,

$$\alpha_{vu}^{(k)} = softmax\Big(g\Big(\mathbb{a}^T\Big[W^{(k)} h_v^{(k-1)} \parallel W^{(k)} h_u^{(k-1)}\Big]\Big)\Big)$$

Where $g(\cdot)$ represents an activation function which might be ReLU function and $\mathbb{a}^T$ is the vector of a learnable parameter and the softmax function is applied to make sure that the attention weights over the neighbour nodes of that central node could be summed up to 1 at the end. GAT then applies multi-head attention mechanism to improve the expressive ability of this architecture. The experimental results prove that GAT does achieves remarkable improvement over GraphSage (Hamilton et al., 2017) on the node classification task.

Mixture Model Network (MoNet) (Monti et al., 2017) generates a different method to calculate the different weights on a central node's neighbouring nodes. It proposed the

pseudo-coordinates of the target node to further calculate the relative position information in between this node and its neighbouring nodes. After all, a weight function will be applied to map out their relative position according to the calculated relative weights on each node. In this way, the parameters of a graph filter can be shared over different locations in the same graph data.

Because many ConvGNNs requires considerable but unnecessary computational complexity when running, this research work (Wu et al., 2019) simplifies the complex and tedious structures of digraphs and reduces the complexity via repeatedly removing the nonlinearities between consecutive GCN layers and integrating the result function into a single linear transformation step. The researchers call this simplified architecture - Simple Graph Convolution (SGC) (Wu et al., 2019). They also find out that the resulting linear architecture could correspond to a fixed low-pass filter and a linear classifier, which then be added into the SGC model. Thus, SGC could smooth the features locally in the graph data at the same time maintain even improve the accuracy on the learning task such as node classification task.

In graph neural network, the high computational complexity is always an issue and it usually could be alleviated through utilizing a sampling strategy, which retains a subset of nodes or graphs at training time step. In the Rossi et al. (2020) research work, researchers proposed a scalable and efficient graph deep learning structures, that can perform group sampling by utilizing different sizes graph convolutional filters which could allow fast inference and training. This novel model is called Simple Scalable Graph Neural Network (SIGN), motivated by the inception module (Szegedy et al., 2016). And this architecture is suitable to do learning tasks on the large-scale graph. From the experimental result, SIGN with only one graph convolutional layer could still

obtain outstanding performances as the state-of-the-art architectures on several large-scale graph datasets.

2.4 Directed Graph

Nowadays, the graph data extracted from the real world could contain more information in the graph such as direction, thus this direction information is also very important to be taken into consideration when it comes to learning tasks. Thus, it is a significant field to have more research so that people could better and more deeply understand and interpret directed graph information.

Recently, more attention has been drawn to the learning from directed graph (digraph) data. Directed Graph Convolutional Network (DGCN) (Tong et al., 2020) was proposed to adapt to the digraph data. The main concept is that they re-define a symmetric normalized Laplacian Matrix for the digraph data by normalizing and symmetrizing the transition probability matrix. Compared to those state-of-the-art ConvGNNs, DGCN performs better on the digraph datasets in the node classification task. However, DGCN also has some disadvantages and limitations: 1). It requires large memory space and high-level computational capacity; 2). It holds the assumption that the digraph data should be strongly connected (Tong et al., 2020).

In Monti et al. (2017), researchers introduced and proposed an architecture called MotifNet, which could handle the digraph data by applying the local graph motifs. The main idea is that the model utilizes the motif-induced adjacencies and at the same time generates graph filters just like convolution. Through the results using the real digraph datasets, it is proved that MotifNet has remarkable performance in directed graph data processing task without requiring higher computational cost.

While in Tong et al. (2020), researcher was inspired by the Inception Network module (Szegedy et al., 2016) and presented the Digraph Inception Convolutional Networks (DiGCN). The researchers construct several scalable receptive fields and at the same time reduce those receptive fields that are unbalanced and generated by the non-symmetric digraph data. The experimental results on various benchmarks demonstrate that DiGCN could learn digraph representation very effectively at the same time it outperforms many state-of-the-art methods.

One of the common research ideas utilized in many architectures is to apply heuristics to generate and adjust the Laplacian matrix to further improve the model's performance in learning tasks. In the very recent work, Zhang et al. (2021) presents a method called MagNet and the main idea is to define a Magnetic Laplacian as a complex Hermitian matrix which could then encode important direction information via these complex numbers. Meanwhile MagNet is also a very flexible approach that could be adapted to many spectral-based ConvGNN models. The experiments manifest that MagNet does outperform all other mainstream graph deep learning structures on the directed graph data's learning tasks including node classification and link prediction (Zhang et al., 2021).

There has been increasing attention in digraph data, however, the digraph focused research is still limited compared to the amount of research work on the normal graph data. Thus, there is still potential in digraph data learning and improve the architectures' performance accuracy by applying novel techniques on the existing models or proposing new structures based on new concepts or ideas from other fields.

2.5 Framelet-based Approach

Framelet method will be applied in the proposed structure because framelet-based method is proved to have advantages of decomposing and reconstructing the signals from the data (Zheng et al). The paper Zheng et al. (2021) explores the graph framelet implementation and application, in which the main idea is that framelet decomposition will induce a graph pooling method by aggregating the graph features which comprised of feature values as well as geometric information of the graph data into high-pass and low pass spectra. While researchers also present shrinkage function as a novel activation function for the framelet convolution and this shrinkage function could assist threshold high-frequency information on different scale levels. The graph neural network with this proposed framelet method and pooling strategy has better performance on the node classification and graph prediction tasks compared to the existing ConvGNNs. Moreover, this framelet-based convolution method enjoys the benefit of fast algorithm during the process of decomposition and reconstruction on signals (Yang et al., 2022).

Manifold defined Framelet-based convolution for signal processing (Dong, 2017) is also implemented and explored for graph signal in paper (Zheng et al., 2021). In Dong (2017) paper, the linear framelet function utilizes a single modulation function $g(\cdot)$, and one set of modulation functions such as the scaling functions in the term of Framelet. Specifically, g_K is used to control the lower frequency at the same time g_0 is used to control the high frequency, while the others will be utilized to regulate all the frequencies left that are not included in g_K and g_0. The normal instances include the linear and quadratic framelet functions (Dong, 2017) and sigmoid and entropy quasi-framelet functions (Yang et al., 2022). The representative linear framelet functions' formula and entropy framelet functions' formula are shown as follows,

Linear Framelet Functions (Dong, 2017):

$$g_0(\xi) = cos^2\left(\frac{\xi}{2}\right);\ \ g_1(\xi) = \frac{1}{\sqrt{2}} sin(\xi);\ \ g_2(\xi) = sin^2\left(\frac{\xi}{2}\right).$$

Entropy Framelet Functions (Yang et al., 2022):

$$g_0(\xi) = \begin{cases} \sqrt{1 - g_1^2(\xi)} & \xi \le \pi/2 \\ 0\,, & otherwise \end{cases};\quad g_1(\xi) = \sqrt{4\alpha\frac{\xi}{\pi} - 4\alpha\left(\frac{\xi}{\pi}\right)^2};$$

$$g_2(\xi) = \begin{cases} \sqrt{1 - g_1^2(\xi)} & \xi > \pi/2 \\ 0\,, & otherwise \end{cases}$$

In which $0 < \alpha \le 1$ denotes a hyper-parameter and note that when $\alpha = 1$, $g_1^2(\pi\xi)$ is a binary entropy function.

Graph framelets (Dong, 2017) which is similar with the traditional wavelet method, provide multiresolution analysis for the graph signals (Zheng et al., 2021). While the fully tensorized framelet transformation makes sure that the efficient graph convolution combines both high-pass and low-pass information, and the transform process only requires graph Laplacian, Chebyshev Polynomial approximation and filter bank $\eta = \{a; b^{(1)}, b^{(2)}, \dots, b^{(K)}\} \subset l_0(Z)$. Therefore, the group of modulation functions is designed according to the multiresolution analysis (MRA) using the filter bank and it could regulate the spectral frequency. However, Yang et al. (2022) further explored and found that the MRA is not necessary, and this paper proposed a group modulation function for quasi-framelets.

Definition 2.1 Modulation functions for Quasi-Framelets

Consider a group of modulation functions K + 1 which are positive and were defined on $[0, \pi]$ and $\mathcal{F} = \{ g_0(\xi), g_1(\xi), \dots, g_K(\xi)\}$; it is a quasi-framelet when it meets the identity condition requirements as follows,

$$g_0(\xi)^2 + g_1(\xi)^2 + \cdots + g_K(\xi)^2 \equiv 1, \qquad \forall \xi \in [0, \pi]$$

In which g_k could increase from 0 to 1, and g_0 could decrease from 1 to 0 over the spectral domain of $[0, \pi]$.

The paper Yang et al. (2022) developed filtering functions in the spectral domain from the perspective of spectral ConvGNNs. Researchers proposed and presented the undecimated quasi-framelet graph (QUFG) convolution for graph neural network via introducing two groups of novel modulation functions. The experimental results further proved QUFG's outstanding denoising ability and flexibility in the node classification task and demonstrated its remarkable performance as the state-of-the-art benchmark architectures'.

2.6 SVD Application in Adversarial Machine Learning and Recommendation System

There has been more and more research studies about the effects of adversarial attacks on graph data and the robustness of the architecture's defences as adversarial attack study is a very important field that could represent many complex problems among AI and machine learning domain and assist effectively to increase the architecture's stability. In paper Entezari et al. (2020), because of the vulnerability of the node classification methods towards the adversarial attacks, thus it's necessary to retain a very robust node classification approach. Then researchers applied the truncated SVD decomposition to compute and derive the low-rank approximation of the feature matrices and adjacency matrices, and further retrain the GCN model with these matrices to boost the performance of GCN when encountering an attacked graph data and compare with the performance of the GCN on a clean graph. And the experiment

results on real-world datasets further prove that using the low-rank (rank-10) SVD approximation of the feature matrices and adjacency matrices is robust enough to vaccinate the GCN model against the attacks in the graph data.

The paper Mujkanovic et al. (2022) is mainly about the adversarial defence robustness, researchers point out the importance of utilizing custom adaptive attacks instead of the non-adaptive attacks which were normally used in the previous works leading to very optimistic robustness estimates. From the experiment result, it is very surprising that none of the assessed Graph Neural Network defense architectures are robust under adaptive attacks. One of the assessed defense model is the SVD-GCN which is introduced in the paper Entezari et al. (2020). And the preliminary experimental result shows that this SVD-GCN cannot achieve considerable robustness gains, compared to an undefended GCN model, not only under adaptive attacks but also non-adaptive attacks.

The third paper I would like to compare here is the paper Peng et al. (2022), the researchers proposed a simplified GCN architecture and replaced the neighbourhood aggregation process with a truncated SVD which exploits the K-largest singular vectors and values, and it is basically designed for the recommendation system. The experimental results further prove that this SVD-GCN does show a positive effect in learning the user-user and item-item relations in the recommendation system while the renormalization trick they proposed to adjust the singular value gap could also significantly alleviate the over-smoothing issue caused by stacking many graph convolution layers in the original GCN architecture.

While in this thesis, we research about the SVD implementation in the GCN architecture using the framelet approach (SVD-framelet) to perform the node

classification and link prediction problems. According to the experiments on several real-world directed graph datasets, it is proved that SVD could improve the original GCN's performance on both tasks because it benefits from the advantages of the SVD-framelets in filtering and transforming the signals from directed graph.

Chapter 3

Deep Learning Structure for Directed Graph Data - SVD-GCN

3.1 Motivation of using SVD and Background

It has been proved by much research works that spectral-based Graph Neural Networks is very useful and powerful regarding node-classification task. This is constructed according to Laplacian on node or the 0^{th} order Hodge Laplacian (Lim, 2020). Consider that X is the graph data's signals, so in spectral-based GNN the basic operation is that $Y = L\,X$, where L is the Laplacian matrix to process the graph signals and this graph Laplacian symmetric and positive semi-definite. For undirected graph, operating singular value decomposition (SVD) of a symmetric matrix is the same as the eigenvalue decomposition (EVD), but there is still a bit different as the only difference is the sign. Suppose that eigenvalue decomposition of normalized Laplacian matrix $\hat{L} = I - \hat{A}$ in which $\hat{A} = (D + I)^{-\frac{1}{2}}(A + I)(D + I)^{-\frac{1}{2}}$, because $\hat{L} = U\Lambda U^T = U(1 - \Sigma)U^T$, where $\hat{A} = U\Sigma U^T$, the eigenvalues of $\hat{A}$ falls in [-1,1] and the eigenvalues of $\hat{L}$ fall in [0,2]. The Laplacian's eigenvalues could be regarded as the frequencies of the signals from graph nodes.

However, when it comes to the case of directed graph, it would be a completely different story since we cannot have the advantage of Laplacian's symmetric property anymore. Thus, a possible method based on the singular value decomposition (SVD) of the adjacency matrix which could be considered to address the issue. This alternative approach depends on the basic operation: $Y = A\,X$, in which A represents the adjacency matrix and this adjacency matrix is asymmetric for directed graph data. While this adjacency matrix is considered as the shift operator for the graph and this shift operator can replace the graph signal on the target node with the linear combination result of the neighbourhood nodes' representations (Gavili & Zhang, 2017).

If the adjacency matrix is diagonalizable, then it doesn't matter whether the graph is undirected or directed, we could always have the equation as $A = V\Lambda V^{-1}$ and the Fourier transform can be generalized as $\hat{x} = V^{-1}x$. Noted that the adjacency matrix is diagonalizable for those graphs that are strongly connected and directed (van Dam & Omidi, 2018), but if the diagonalizable condition is violated then Jordan decomposition could be taken into consideration (Sandryhaila & Moura, 2013). Meanwhile it needs to be noted that generally for directed graph data, $V^{-1} \neq V^{T}$, and S and V are both normally complex-valued, that hinders the extension to use the classic framelet theories on directed graph data.

To solve this problem, singular value decomposition (SVD) is considered as the graph shift operator, $A = U\Lambda V^{T}$, in which U, V represent two groups of orthonormal bases which contains positive and real singular values Λ. Because SVD is a suitable approach to decompose and reconstruct the signal matrix utilizing orthogonal system and this is also the reason why singular value decomposition (SVD) is applied instead of other methods here. Utilizing this graph shift operator on the graph signal X could be

considered as decomposing the signals via the two groups of bases which is based on the V's columns then the decomposition result will be put into a scaling operation determined by Λ, then the scaled signal result will be transformed and reconstructed by another group of orthonormal bases determined by the U's columns. Because the magnitude of Λ represents "frequency", then Λ could be regulated by a modulation function g and the filter could be defined as follows,

$$Y = \sigma\left(\left(V_g(\Lambda)U^T\right) \cdot g_\theta \circ \left(U_g(\Lambda)V^T X\right)\right) \quad (1)$$

In short, this first motivation is the successful improvement via utilizing the multiple frequency separation for the graph signals through framelet decomposition while Framelet (Zheng et al. 2021) relies heavily on the Fourier decomposition of the graph signals and because it needs asymmetric adjacency matrix which is quite difficult for the directed graph to generate. However, SVD could provide a direct method to the graph signal decomposition, and it assists us to exploit the benefits of the frequency separation.

The second motivation is that we would like to retain the superiority of the specialized spectral GNNs for the directed graph data at the same time avoid the demand of a carefully designed Laplacian matrix or adjacency matrix. We basically perform convolution over the spectral domain which is provided by the singular value decomposition of the directed adjacency matrix and this approach could be applied to any structured matrix of a directed graph data while it is not limited to the graph adjacency either.

3.2 Methodology

3.2.1 SVD-Framelets

A recent research paper regarding undecimated framelets-enhanced graph neural network architecture which is also called UFG achieved outstanding results in different learning tasks on graph data (Zheng et al. 2021). Because UFG is developed based on the framework of spectral graph signal analysis by utilizing multiresolution analysis developed by applying classic framelet theory (Dong, 2017), but it is quite limited to apply this classic framelet on the directed graph. While for directed graph data signals, it is important to figure out the method to explore the multiresolution lens. And this motivated us to look back to find out if there is any signal analysis technique that could be applied into the graph neural network on directed graph data.

Consider that a directed graph (homogeneous) $G = (V, E)$ with graph signal X and n nodes in the graph; Suppose the $A \in \mathbb{R}^{n*n}$ represents its asymmetric adjacency matrix while D_1 is the in-degree diagonal matrix and D_2 is the out-degree diagonal matrix. If we consider it as self-looped normalized adjacency matrix first, then the matrix could be written as $\hat{A} = (D_1 + I)^{-\frac{1}{2}}(A + I)(D_2 + I)^{-\frac{1}{2}}$. In the spatial-based GNN architectures, $\hat{A}$ is usually utilized to determine the convolutional layer as follows,

$$X' = \hat{A}\, X\, W \qquad (2)$$

And now we set the SVD for the normalized adjacency matrix as follows,

$$\hat{A} = U\, \Lambda\, \mathrm{V}^T, \qquad (3)$$

In which V holds the right singular vectors while U holds the left vectors and $\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_N)$ represents all the singular values' diagonal matrix and the singular values are listed in a decreasing order. Now integrating equation (3) into equation (2),

this step means the projection of the node signals X on the graph onto the orthogonal system that was generated based on V's columns, and then signals are reconstructed based on the dual orthogonal system which was constructed using U's columns, at the same time this process is assisted by certain scaling function via singular values Λ. The procedure is about how to filter the graph signals from the directed graph data via the dual orthogonal systems.

While motivated by concept about utilizing the undecimated framelets in the proposed orthogonal systems of Laplacian, we implemented framelets onto the orthogonal systems that are determined by SVD. Consider one group of framelet functions $\mathcal{F} = \{g_0(\xi), g_1(\xi), \dots, g_K(\xi)\}$ which falls on the range $[0,\pi]$ (Yang et al., 2022; Zheng et al., 2021), and given a multiresolution level L, then the framelet signal decomposition and reconstruction operators could be defined as follow,

$$\mathcal{W}_{0,L} = V g_0\left(\frac{\Lambda}{2^{m+L}}\right)\cdots g_0\left(\frac{\Lambda}{2^m}\right)\Lambda^{\frac{1}{2}}V^T, \mathcal{W}_{k,0} = V g_k\left(\frac{\Lambda}{2^m}\right)\Lambda^{\frac{1}{2}}V^T, \text{for } k = 1,2,\dots,K,$$

$$\mathcal{W}_{k,l} = V g_k\left(\frac{\Lambda}{2^{m+l}}\right) g_0\left(\frac{\Lambda}{2^{m+l-1}}\right)\cdots\ g_0\left(\frac{\Lambda}{2^m}\right)\Lambda^{\frac{1}{2}}V^T \text{ for } k = 1,2,\dots,K, l = 1,2,\dots,L. \quad (4)$$

and

$$\mathcal{V}_{0,L} = U\Lambda^{\frac{1}{2}} g_0\left(\frac{\Lambda}{2^m}\right)\cdots g_0\left(\frac{\Lambda}{2^{m+L}}\right)V^T, \mathcal{V}_{k,0} = U\Lambda^{\frac{1}{2}} g_k\left(\frac{\Lambda}{2^m}\right)V^T, \text{for } k = 1,2,\dots,K,$$

$$\mathcal{V}_{k,l} = U\Lambda^{\frac{1}{2}} g_0\left(\frac{\Lambda}{2^m}\right)\cdots g_0\left(\frac{\Lambda}{2^{m+L-1}}\right) g_k\left(\frac{\Lambda}{2^{m+L}}\right)\ V^T \text{ for } k = 1,2,\dots,K, l = 1,2,\dots,L. \quad (5)$$

We then stack the corresponding equations together in the column direction as $\mathcal{W} = [\mathcal{W}_{0,L}; \mathcal{W}_{1,0}; \dots; \mathcal{W}_{K,0}; \mathcal{W}_{1,1}; \dots; \mathcal{W}_{K,L}]$ while stack them in the row direction as $\mathcal{V} = [\mathcal{V}_{0,L}; \mathcal{V}_{1,0}; \dots; \mathcal{V}_{K,0}; \mathcal{V}_{1,1}; \dots; \mathcal{V}_{K,L}]$.

Theorem 3.1 *The SVD-GCN layer could be applied via a step comprised of decomposition and reconstruction which are determined by operator $\mathcal{W}$ and $\mathcal{V}$, then taking these two operators into the SVD equation $\hat{A} = U\,\Lambda\,\mathrm{V}^T$, the result will be*

$$X' = \hat{A}\,X\,W = \mathcal{V}\,(\mathcal{W}XW).$$

Proof. We will prove that $\hat{A} = \mathcal{W}\mathcal{V}$ from the equation shown above. While the framelet functions' identity property will be utilized $\sum_{k=0}^{K} g_k^2(\xi) \equiv 1$, in accordance with the previous equations of matrices $\mathcal{V}$ & $\mathcal{W}$, the proof process is shown as follow,

$$\mathcal{W}\mathcal{V} = \mathcal{W}_{0,L}\mathcal{V}_{0,L} + \sum_{l=0}^{L}\sum_{k=1}^{K}\mathcal{W}_{k,l}\mathcal{V}_{k,l} = \mathcal{W}_{0,L}\mathcal{V}_{0,L} + \sum_{k=1}^{K}\mathcal{W}_{k,L}\mathcal{V}_{k,L} + \sum_{l=0}^{L-1}\sum_{k=1}^{K}\mathcal{W}_{k,l}\mathcal{V}_{k,l}$$

$$= V g_0\left(\tfrac{\Lambda}{2^{m+L}}\right)\cdots g_0\left(\tfrac{\Lambda}{2^{m}}\right)\Lambda^{\frac{1}{2}}V^T \cdot U\Lambda^{\frac{1}{2}} g_0\left(\tfrac{\Lambda}{2^{m}}\right)\cdots g_0\left(\tfrac{\Lambda}{2^{m+L}}\right)V^T$$

$$+ \sum_{k=1}^{K} V\, g_0\left(\frac{\Lambda}{2^{m+l-1}}\right)\cdots\, g_0\left(\frac{\Lambda}{2^{m}}\right)\Lambda^{\frac{1}{2}}V^T \cdot U\Lambda^{\frac{1}{2}} g_0\left(\frac{\Lambda}{2^{m}}\right)\cdot\cdot$$

$$\cdot\, g_0\left(\frac{\Lambda}{2^{m+L-1}}\right) g_k\left(\frac{\Lambda}{2^{m+L}}\right) V^T$$

$$+ \sum_{l=0}^{L-1}\sum_{k=1}^{K}\mathcal{W}_{k,l}\mathcal{V}_{k,l}$$

$$= U\Lambda^{\frac{1}{2}} g_0\left(\frac{\Lambda}{2^{m}}\right)\cdots g_0\left(\frac{\Lambda}{2^{m+L-1}}\right)\left(\sum_{k=0}^{K} g_k^2\left(\frac{\Lambda}{2^{m+L}}\right)\right)\cdot g_0\left(\frac{\Lambda}{2^{m+L-1}}\right)\cdots g_0\left(\frac{\Lambda}{2^{m}}\right)\Lambda^{\frac{1}{2}}V^T$$

$$+ \sum_{l=0}^{L-1}\sum_{k=1}^{K}\mathcal{W}_{k,l}\mathcal{V}_{k,l} = \mathcal{W}_{0,L-1}\mathcal{V}_{0,L-1} + \sum_{l=0}^{L-1}\sum_{k=1}^{K}\mathcal{W}_{k,l}\mathcal{V}_{k,l} = \cdots$$

$$= epeated\ this\ argument\ L\ -\ 1\ more\ times$$

$$= \mathcal{W}_{0,0}\mathcal{V}_{0,0} + \sum_{k=1}^{K}\mathcal{W}_{k,0}\mathcal{V}_{k,0}$$

$$= U\Lambda^{\frac{1}{2}}\left(\sum\nolimits_{k=0}^{K} g_k^2\left(\frac{\Lambda}{2^{m}}\right)\right)\Lambda^{\frac{1}{2}}V^T = U\,\Lambda\,V^T = \hat{A}$$

The calculation process above completest the proof that $\hat{A} = \mathcal{W}\mathcal{V}$.

3.2.2 Decomposition and Reconstruction of SVD-Framelet Signal

In this step, the graph SVD framelet needs to be defined to further construct the SVD-GCN layer.

Suppose that there are all singular vector triples and singular values for normalized adjacency $\hat{A}$ in the set of $\{(\lambda_i, u_i, v_i)\}_{i=1}^{N}$ for graph G, in which there are N nodes, and u_i represents the columns of U and v_i represents the columns of V, and $\{\lambda_i\}$ are in a decreasing order accordingly. While note that $\beta_0^k(\xi) = g_k\left(\frac{\xi}{2^m}\right)$ as well as

$$\beta_l^k(\xi) = g_k\left(\frac{\xi}{2^m}\right) g_0\left(\frac{\xi}{2^{m-1}}\right) \cdots g_0\left(\frac{\xi}{2^{m-l}}\right) \text{ for } l = 1,2,\dots,L, k = 1,2,\dots,K.$$

Thus, consider a group of modulation functions $\mathcal{F} = \{g_0(\xi), g_1(\xi), \dots, g_K(\xi)\}$, then the forward SVD framelet for a graph G at scale level l could be written as follows,

$$\phi_{o,p}(q) = \sum_{i=1}^{N} \sqrt{\lambda_i}\, \beta_L^0\left(\frac{\lambda_i}{2^L}\right) v_i(p) v_i(q), \qquad \psi_{o,p}^k(q) = \sum_{i=1}^{N} \sqrt{\lambda_i}\, \beta_0^k\left(\frac{\lambda_i}{2^0}\right) v_i(p) v_i(q)$$

$$\psi_{l,p}^k(q) = \sum_{i=1}^{N} \sqrt{\lambda_i}\, \beta_l^k\left(\frac{\lambda_i}{2^l}\right) v_i(p) v_i(q), \; l = 1,2,\dots L; k = 1,2,\dots,K. \tag{6}$$

While the backward SVD framelet could be written as shown below,

$$\bar{\phi}_{o,p}(q) = \sum_{i=1}^{N} \sqrt{\lambda_i}\, \beta_L^0\left(\frac{\lambda_i}{2^L}\right) u_i(p) v_i(q), \; \bar{\psi}_{o,p}^k(q) = \sum_{i=1}^{N} \sqrt{\lambda_i}\, \beta_0^k\left(\frac{\lambda_i}{2^0}\right) u_i(p) v_i(q)$$

$$\bar{\psi}_{l,p}^k(q) = \sum_{i=1}^{N} \sqrt{\lambda_i}\, \beta_l^k\left(\frac{\lambda_i}{2^l}\right) u_i(p) v_i(q), \; l = 1,2,\dots L; k = 1,2,\dots,K. \tag{7}$$

Where for all nodes q, $\phi_{l,p}(q)$ & $\bar{\phi}_{l,p}(q)$ denote that at node p, the low-pass SVD framelet translated meanwhile $\psi_{l,p}^k(q)$ & $\bar{\psi}_{l,p}^k(q)$ denote the high-pass SVD framelet translated.

If the concept of undecimated framelet system (Dong, 2017) is applied as well, then the two SVD- Framelet operator could be defined as follows,

$$\text{SVD-UFS-}F_L(\mathcal{F},G) \coloneqq \{\phi_{o,p} : p \in \mathcal{V}\} \cup \{\psi_{l,p}^k : p \in \mathcal{V},\ l = 0,1,\dots,L\}_{k=1}^{K}, \quad (8)$$

$$\text{SVD-UFS-}B_L(\mathcal{F},G) \coloneqq \{\bar{\phi}_{o,p} : p \in \mathcal{V}\} \cup \{\bar{\psi}_{l,p}^k : p \in \mathcal{V},\ l = 0,1,\dots,L\}_{k=1}^{K}, \quad (9)$$

While the signal transform $x' = \hat{A}\,x$ can also be applied in the SVD framelet transform operator and the theorem is shown below.

Theorem 3.2 Transform of SVD-Framelet

Consider the forward and backward SVD framelet systems' definition, the signal transform could be represented as follows,

$$x' = \sum_{p\in\mathcal{V}}\langle \phi_{L,p}, x\rangle \bar{\phi}_{L,p} + \sum_{k=1}^{K}\sum_{l=0}^{L}\sum_{p\in\mathcal{V}}\langle \psi_{l,p}^k, x\rangle \bar{\psi}_{l,p}^k \qquad (10)$$

The decomposition process here is to re-write $x' = \hat{A}\,x$, from the equation shown above, the transformed graph signal x' is indicated as a linear equation expression of integrating backward SVD framelet operator with the partial signal from the system of forward SVD framelet. Therefore, the filtering process of signals is mormally developed by filtrating forward SVD framelet coefficient $\langle \psi_{l,p}^k, x\rangle$.

3.2.3 Model Architecture & Simplified SVD-Framelet Filtering

According to the Theorem 3.1 & Theorem 3.2 shown above, the simplified SVD framelet filtering step could be presented as follows,

$$Y = \sigma\left(\sum_{k=0}^{K}\left(U\Lambda^{\frac{1}{2}}\, g_k(\Lambda)V^T\right)\cdot g_\theta^k \circ \left(V\, g_k(\Lambda)\Lambda^{\frac{1}{2}}\, V^T XW\right)\right) \qquad (11)$$

Where W is the transformation weight of learnable features, g_θ^k represents the filters based on each modulation function g_k and σ is the activation function. Because it

would be not always mandatory to further process the graph signal to that system of backward SVD-framelet. Thus, SVD framelet filtering process could be simplified via considering the transformation with only the forward SVD framelet operator as follows,

$$Y = \sigma(\sum_{k=0}^{K}(V\Lambda^{\frac{1}{2}}\, g_k(\Lambda)V^T) \cdot g_\theta^k \circ (V\, g_k(\Lambda)\Lambda^{\frac{1}{2}}\, V^T XW)) \qquad (12)$$

Where the difference between (11) and (12) is that U is replaced by V.

In the Figure 7 below, it demonstrates how the SVD framelet layer works. Firstly, the normalized adjacency matrix $\hat{A}$ is generated from the graph to get the framelet matrices $\mathcal{W}$ and $\mathcal{V}$ at certain scale level L; secondly, the framelet matrices $\mathcal{W}$ which is the primary matrices, is adapted into matrix X of input node signal. Then the result will be passed into learnable filters g_θ on each node. While the dual framelet matrices $\mathcal{V}$ will be utilized to filter the signal and pass back to the transformed signal domain, which is represented as X' and this will be forwarded to the next layer for further processing. It is important to note that when several layers of SVD are implemented in the model, both framelet matrices $\mathcal{W}$ and $\mathcal{V}$ are shared throughout all those SVD layers.

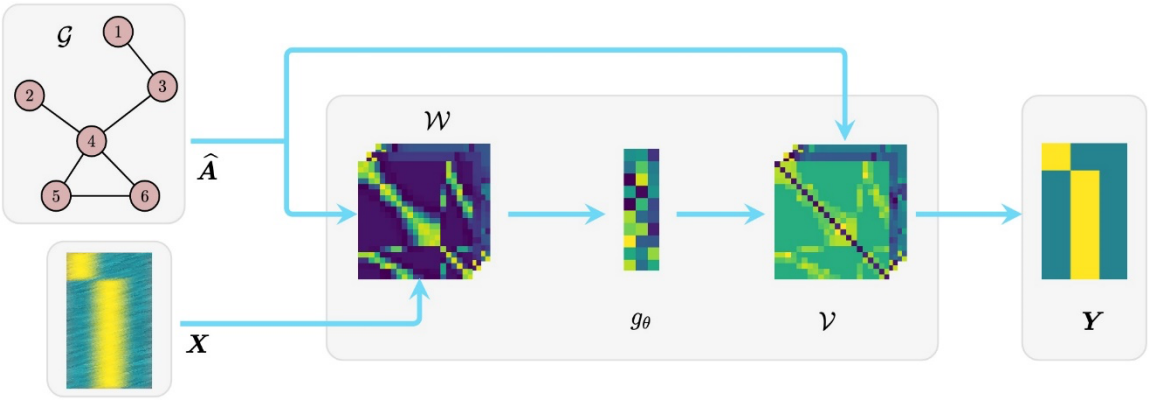


Figure 7 SVD-framelet System: SVD framelet layer transforms the feature X of the input node utilizing SVD framelet matrices W and V and applying learnable filters g_ϑ on those new features Y; this process is illustrated in the simplified framelet parts (11) and (12).

3.2.4 Faster Filtering for Large Graphs

In real world, there are many datasets that are in large scale, thus if we would like to utilize the proposed model to solve the real problems in the future, it is very significant to make sure that this architecture is applicable on the large-scale datasets because usually large-scale datasets have the issues of extremely high-level computational complexity. Implementing SVD technique on adjacency matrix of large graph data must be very costly. Thus, an approximated filter developed on Chebyshev polynomials is considered, we adapt the idea proposed in (Onuki et al., 2017) and construct the fast filtering. For the graph normalized adjacency matrix $\hat{A} = U\,\Lambda\,V^T = \hat{A}\,V\,V^T$ and $\hat{A}^T\hat{A} = V\Lambda^2\,V^T$, which means that V's columns provide the eigenvector systems for $\hat{A}^T\hat{A}$, and then framelet analysis can be finished for the Laplacian matrix.

Given a group of framelet functions or quasi-framelet functions $F = \{g_0(\xi), g_1(\xi), \dots, g_K(\xi)\}$ defined on $[0,\pi]$, the framelet signal decomposition and reconstruction operator are written as follows,

$$\mathcal{W}_{0,L} = V g_0\left(\frac{\Lambda^2}{2^{m+L}}\right)\cdots g_0\left(\frac{\Lambda^2}{2^m}\right)V^T, \mathcal{W}_{k,0} = V g_k\left(\frac{\Lambda^2}{2^m}\right)V^T, \text{for } k = 1,2,\dots,K,$$

$$\mathcal{W}_{k,l} = V g_k\left(\frac{\Lambda^2}{2^{m+l}}\right) g_0\left(\frac{\Lambda^2}{2^{m+l-1}}\right)\cdots\ g_0\left(\frac{\Lambda^2}{2^m}\right)V^T \text{ for } k = 1,2,\dots,K, l = 1,2,\dots,L. \quad (13)$$

These equations shown above are quite similar with the equation (4), while the difference is that the original Λ is replaced by Λ^2 in the $g(\cdot)$, and no extra term of $\Lambda^{\frac{1}{2}}$.

Meanwhile, the polynomial approximation method could also be considered to be applied to each modulation function $g_j(\xi)$ where $j = 0,1,2,\dots,K$, to avoid the explicit SVD decomposition of V. Thus, $g_j(\xi)$ could be approximated by Chebyshev polynomials $\mathcal{T}_j^n(\xi)$, in which n is a fixed integer and it needs to be selected so that

Chebyshev polynomial approximation is of high precision. To simplify and get a less complicated expression of the equation, $\mathcal{T}_j(\xi)$ will be utilized in the equation below rather than using $\mathcal{T}_j^n(\xi)$. Afterwards, the transformation equations of matrices' SVD-framelet can be written and determined as follow,

$$\mathcal{W}_{0,L} \approx \mathcal{T}_0\left(\frac{1}{2^{m+L}}\hat{A}^T\hat{A}\right)\cdots\mathcal{T}_0\left(\frac{1}{2^m}\hat{A}^T\hat{A}\right), \qquad \mathcal{W}_{k,0} \approx \mathcal{T}_k\left(\frac{1}{2^m}\hat{A}^T\hat{A}\right),$$

$$\mathcal{W}_{k,l} \approx \mathcal{T}_k\left(\frac{1}{2^{m+l}}\hat{A}^T\hat{A}\right)\mathcal{T}_0\left(\frac{1}{2^{m+l-1}}\hat{A}^T\hat{A}\right)\cdots\mathcal{T}_0\left(\frac{1}{2^m}\hat{A}^T\hat{A}\right), \tag{14}$$

where $k = 1, 2, \ldots, K, l = 1, 2, \ldots, L$.

When it is not necessary to always utilize the adjacency SVD to generate the framelet matrices. The large scale simplified SVD-Framelet-III when L = 0 could be denoted as follows and it will be further exploited in the experiments later,

$$Y = \sigma(\hat{A}\sum_{k=0}^{K}\mathcal{W}_{k,0}^T \cdot g_\theta^k \circ (\mathcal{W}_{k,0}\, X\, W)) \tag{15}$$

3.3 Node Classification Experiment

In this project, the directed graph data utilized can be found in: https://pytorch-geometric.readthedocs.io/, including dataset Citeseer, Citeseer_full and Cora_ml, all of them are citation networks, meanwhile the Amazon Photo (Amazon_photo) and Amazon Computers (Amazon_cs) which are both co-purchase network. The code for this experiment is available at https://github.com/ThisIsForReview/SVD-GCN. Brief descriptions of all the datasets will be provided below and Table 1 will summarize the datasets' basic statistics.

3.3.1 Experimental Protocol

3.3.1.1 Datasets

Cora_ml & **Cora_full** (Bojchevski & Günnemann, 2017): Cora_ml dataset is a relatively small dataset that is taken from the original Cora dataset and they are classic citation network datasets as well as directed network datasets, in which nodes represent published papers and edges represent their citation relationship.

Citeseer (Yang et al., 2016) & **Citeseer_full** (Chen et al., 2018): Citeseer dataset is also a popular citation network dataset, in which all the nodes means papers while edges denote the citation relationship between papers. The main difference between Citeseer and Citeseer_full dataset is that in Citeseer_full the data split type is full.

Amazon_cs & **Amazon_photo** (Shchur et al., 2018): The dataset Amazon computer and dataset Amazon photo both are separately taken from the original Amazon co-purchase network. While in both datasets, nodes represent products and goods such as computers or photos, while edges represent the relationship that these two products are bought together by clients frequently, meanwhile the products' review are the features of the nodes.

Table 1 Datasets Statistics

Dataset Name	# of Node	# of Edges	# of Classes	# of Features
Cora_ml	2,995	8,416	7	2,879
Citeseer	3,312	4,715	6	3,703
Citeseer_full	3,327	3,703	6	602
Amazon_photo	7,650	143,663	8	745
Amazon_cs	13,752	287,209	10	767
Cora_full	19,793	65,311	70	8,710

3.3.1.2 Baseline Architectures

In this node classification task, the proposed model SVD-GCN will be compared with fourteen existing state-of-the-art architectures, including spatial-based GNNs such as GAT (Velickovic et al., 2018) and GraphSage (Hamilton et al., 2017); spectral-based GNNs such as GCN (Kipf & Welling, 2016), ChebNet (Defferrard et al., 2016), APPNP (Klicpera et al., 2019), SGC(Wu et al., 2019), and InfoMax (Velickovic et al., 2019); Digraph GNNs including DGCN (Tong et al., 2020); Graph Inception including SIGN (Rossi et al., 2020); Digraph Inception including DiGCN-PR (Tong et al., 2020), DiGCN-APPR-IB (Tong et al., 2020), DiGCN-APPR (Tong et al., 2020). Meanwhile UFG/QUFG generated based on Linear framelet functions (Zheng et al., 2021) and Entropy framelet functions (Yang et al., 2022) will also be utilized in the experiments. However, because both UFG and QUFG are designed for undirected graph data, before applying these two models on the directed graph, there will be a simple step to simply convert a directed graph to an undirected one by adding reversed edges.

3.3.2 Training Setup

In this experiment, the hyperparameter are selected as follows, 20 nodes are chosen for model training for one class, while 500 nodes are grouped as a validation set at the same time the rest of nodes are included in the testing set; the basic epoch is 200 and the two-level framelets (L=1) are utilized in the experiment, while the dilation scale in the framelets is also tested for values of 1.1, 1.5 and 2.0; the number of hidden layers is tested for values of 16, 32 and 64, while the dropout ratio is tested for values of 0.1, 0.3 and 0.6 and the value with the best accuracy rate will be retained; the framelet modulation function is neither Entropy nor Linear and the hyperparameter α will be tried at 0.1, 0.3, 0.5, 0.7 and 0.9, and it showed that both framelet modulation functions do not have large different effects on the final results, so all the reported results'

experiments utilize the linear framelet modulation functions. Overall, this SVD-GCN architecture comprised of one SVD framelet layer, then followed by a fully connected linear layer then a softmax output layer to generate final outputs.

3.3.3 Result Analysis

The experimental results are listed in Table 2. The proposed SVD-GCN obtains outstanding performances in most cases on all five digraph datasets compared to all the state-of-the-art baseline architectures, if not it still achieves comparable performances. For each dataset, the highest accuracy rate is bolded for highlight in the table, and it is clear that most of the bolded results are obtained by SVD-GCN. For Citeseer, Citeseer_full and Cora_ml datasets, more than 1% increase in the accuracy rate has been achieved by the proposed method; while for the two Amazon datasets, the SVD-GCN structure's results are similar with the results of DiGCN-APPR model which obtains the highest accuracy rate. It is noticeable that in the experiment on dataset Citeseer_full, SVD-GCN achieves larger than 6% increase in accuracy rate compared to the highest rate obtained by model DiGCN-PR and DiGCN-APPR (Tong et al.) among the state-of-the-art models, and this could be called a remarkable improvement achieved by this proposed architecture.

Tong et al, paper further presented and proposed the Digraph Inception Convolutional Networks (DiGCN-APPR-IB) where the directed graph's convolution and k^{th}-order proximity are utilized to construct larger receptive fields as well as to learn the multi-scale features in directed graph data. This strategy is also applied and integrated with this SVD-GCN architecture, resulting model is called SVD-GCN-IB. The results are shown in the bottom two tows of Table 2. It clearly demonstrates that larger receptive fields could improve the performances of SVD-GCN.

While for each dataset, we did quick experiments using UFG and QUFG and the reported results were compared against the results' of SVD-GCN. From the accuracy rates in the table, it is obvious that UFG/QUFG performs generally well and sometimes even better than the performances of some GNNs which are specifically designed for digraph data. Thus, it further proves that utilizing multiple-scale decomposition in Graph Neural Network is beneficial.

Table 2 Results for Node Classification Accuracy (%); Note: OOM means "out of memory"

Models	Cora_ml	Citeseer	Citeseer_full	Amazon_photo	Amazon_cs
ChebNet	66.11±1.5	58.57±1.0	62.29±0.3	80.91±1.0	73.25±0.8
GCN	71.69±0.6	61.83±0.3	64.71±0.5	53.20±0.4	60.50±1.6
SGC	71.75±0.5	62.40±0.4	56.56±0.4	71.25±1.3	76.17±0.1
APPNP	70.07±1.1	65.39±0.9	67.53±0.4	79.37±0.9	63.16±1.4
InfoMax	58.00±2.4	60.51±1.7	72.93±1.1	74.40±1.2	47.32±0.7
GraphSAGE	72.06±0.9	63.19±0.7	65.18±0.8	87.57±0.9	79.29±1.3
GAT	71.91±0.9	63.03±0.6	66.67±0.4	89.10±0.7	79.45±1.5
UFG/QUFG	76.98±5.3	64.70±1.7	80.44±1.3	80.98±1.5	77.32±4.3
DGCN	75.02±0.5	66.00±0.4	78.35±0.3	83.66±0.8	OOM
SIGN	64.47±0.9	60.69±0.4	77.44±0.1	74.13±1.0	69.40±4.8
DiGCN-PR	77.11±0.5	64.77±0.6	74.18±0.7	OOM	OOM
DiGCN-APPR	77.01±0.4	64.92±0.3	74.52±0.4	88.72±0.3	85.55±0.4
SVD-GCN	**78.84±0.29**	**66.15±0.39**	**80.95±0.36**	**88.76±0.21**	**85.55±0.31**
DiGCN-APPR-IB	80.25±0.5	66.11±0.7	80.10±0.3	**90.02±0.5**	**85.94±0.5**
SVD-GCN-IB	**81.11±0.24**	64.26±0.77	**83.12±0.68**	89.38±0.48	85.03±0.37

3.3.4 Fast Algorithm Experiment

3.3.4.1 Experimental Protocol

This experiment will further explore the fast SVD-Framelet-III introduced in the the previous chapter[1]. The dataset Cora_full is utilized and this dataset is also used in the Bojchevski & Günnemann, (2017). Cora_full is full extension of the other Cora dataset Cora_ml. From the dataset's statistics table, it is clear that Cora_full is a quite large datasets compared to other datasets utilized in the node classification learning task, with 19,793 nodes ad 65,311 edges and the number of node classes is 70 while there are 8,710 feature dimensions. Usually, regarding the graph data with more than 15k nodes, we need to convert to use the CPU for slow training because the large dataset cannot fit on the GPU memory.

The purpose here is to test the reliability of the proposed simplified version of SVD-Framelet for fast algorithm called SVD-Framelet-III determined by the Chebyshev Polynomial approximation. While corresponding experiments will compare this SVD-Framelet-III's results with the very basic state-of-the-art model GCN. We also originally would like to compare with DiGCN-PR and DiGCN-APPR architecture, however we cannot even run these experiments on the CPU. Thus, the results cannot be provided.

The related results are provided in Table 3. The coding parameters' setup in python for this experiment is similar with the setting for the previous node classification learning task. 20 nodes per class is chosen for training, 500 random nodes are assigned to validation set and the rest of the nodes are assigned to the test set. The framelet scale

[1] We originally would like to conduct this experiment on the Large-scale dataset to test the proposed architecture's reliability and efficiency, However, we didn't find suitable large-scale directed graph dataset as some large-scale directed graph datasets do not have node labels or features, otherwise we would encounter nnz (number of non-zeros) overflow issue when conducting the experiment.

is set to 1.1, which means that linear framelet modulation functions are selected in this experiment and at the same time the dropout rate is set to 0.1, and ReLU is chosen as the activation function. The network model comprises of one layer of SVD-framelet, followed by one layer of fully linear and then output will be feed into the output softmax layer. Each experiment will be conducted on the replicate of 10 each time while it runs 200 epochs with a fix 0.005 learning rate in each replicate.

3.3.4.2 Result Analysis

The corresponding average accuracy rate with its standard deviation is reported in the table below. The results demonstrate that the most ideal hidden unit size is 128 for this large dataset, while SVD-GCN achieved 1-3% increase in the accuracy rate in all cases compared to the GCN architecture. This further proves that the simplified fast SVD-Framelet-III does help improve the architecture's performances in node classification tasks when the dataset is relatively large.

Table 3 Results between GCN and SVD-Framelet-III over Cora_Full

# Hidden Units	GCN	**SVD-GCN**
64	52.54±0.41	53.52±0.33
128	53.75±0.17	56.17±0.16
256	54.33±0.33	57.30±0.21

3.3.5 Denoising Capability and Robustness

3.3.5.1 Dataset and Baseline

Experiments are set to be conducted to further evaluate the robustness of the SVD-GCN and the results will be compared with the DiGCN-APPR model's. The process to test the robustness is that we manually adjust the σ which is the standard deviation of the noises to add different levels of noise to the dataset, and the dataset utilized in this

experiment is Cora_ml. While to evaluate the denoising capability of the proposed model SVD-GCN, experiments are set up which is to randomly inject the "noise" of 0 mean with 0.01-5 standard deviation into the dataset and then compare the results with the results from DiGCN-APPR architecture.

3.3.5.2 Result Analysis

In the Table 4 below, the results with all noise levels from 0.01 to 5.0 are reported. However, the results for DiGCN-APPR for noise level of 1.0 and 5.0 are not reported because the result figures are too poor and when the accuracy rate is less than 40%, it is usually considered as poor figures and not comparable. From the reported experimental results in Table 4, it is clear that SVD-GCN has much better ability of denoising because its accuracy rates keep at a high level when the injected noise level α is relatively large. When noise level α becomes large than 0.01, the accuracy rate of DiGCN-APPR has dropped dramatically from 53.39% to 35.72%, while the proposed SVD-GCN architecture still has a comparable rate of accuracy.

Table 4 Results between SVD-GCN and DiGCN-APPR on Cora_ml on Different noise levels

Noise Level	DiGCN-APPR (Tong et al., 2020)	**SVD-GCN**
$\alpha = 0.0$	77.01 ± 0.40	78.84 ± 0.29
$\alpha = 0.01$	53.39 ± 0.61	76.04 ± 0.51
$\alpha = 0.05$	35.72 ± 0.80	71.04 ± 0.50
$\alpha = 0.1$	34.08 ± 1.34	68.25 ± 0.57
$\alpha = 0.5$	30.40 ± 1.92	67.37 ± 0.47
$\alpha = 1.0$	--	66.38 ± 0.82
$\alpha = 5.0$	--	59.63 ± 0.74

3.3.5.3 Sensitivity Analysis

From the Table 4 above, it is quite clear that DiGCN-APPR fails in the denoising testing experiments However, SVD-GCN is more robust and has much better denoising

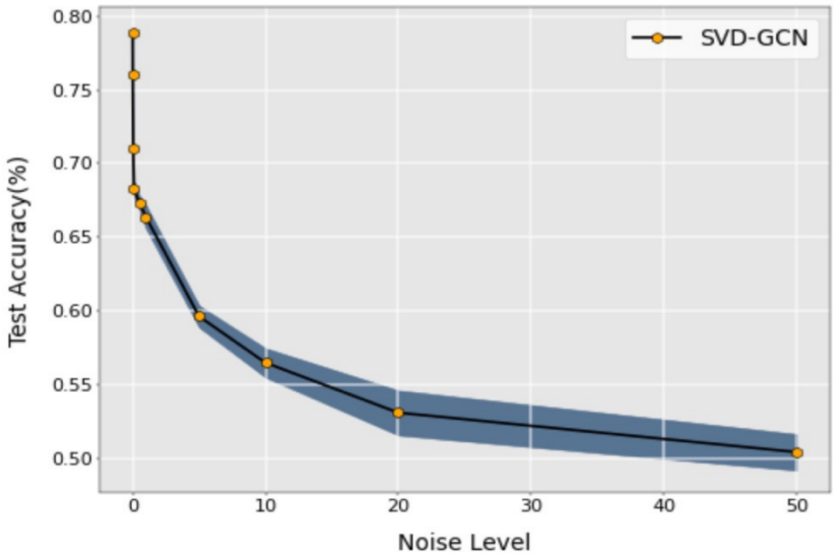


Figure 8 Analysis of Node Attribute Perturbation on the Cora_ml dataset

capability compared to DiGCN-APPR because it has the advantage of framelet decomposition on the domain of the SVD "frequency" and the filtering step into learning process. More experiments are conducted using SVD-GCN on even larger noise level of 10.0, 20.0 and etc. and the results are utilized to draw a line graph to have a better overview of the test accuracy, shown in Figure 8. The yellow dot represents the accuracy rate on each noise level while the blue shade area represents the standard deviation of the accuracy rates. From the Figure 8 below, it is quite evident and apparent that when the noise level is at 50.0, the result is around 50% and the corresponding standard deviation is still acceptable. The results further prove and demonstrate that SVD-GCN is quite robust and consistent in its performance in node classification task when it is even facing much bigger attack from noise.

3.3.6 Contribution and Discussion

The application of framelets on the dual orthogonal system is further explored while this dual orthogonal system is constructed by singular vectors from the singular value decomposition (SVD) on the graph data. In this project, SVD-GCN is proposed for directed graph data. The SVD-GCN is proven to improve the original GCN's performance on node classification task because it benefits from the advantages of the SVD-framelets in filtering and transforming the signals from directed graph. The results prove that SVD-GCN performs better than state-of-the-art architectures tested in this experiment on those five benchmark digraph datasets, which demonstrates that this proposed SVD-GCN has remarkable performance in processing digraph data. While the sensitivity analysis and the robustness experiments further manifest that SVD-GCN is robust and reliable to handle high-level noise attack.

For the fast algorithm experiment, originally we would like to conduct this experiment on the Large-scale dataset to test the proposed SVD-GCN architecture's reliability and efficiency. However, Cora_full is actually not large enough to be considered as a large-scale dataset thus the related experiment might not be material to prove SVD-GCN's reliability and improvement in processing large-scale dataset on node classification tasks. Meanwhile, we also consider using the OGBN-Arxiv which is much larger dataset with 169,343 nodes and 1,166,243 edges and it is more suitable to be utilized in the large-scale dataset experiment. However, it failed to run the coding on OGBN-Arxiv dataset because of a nnz overflow issue from the sparse matrix. And some other large-scale directed graph dataset that is suitable in this experiment do not have node labels or features, which means that we still cannot run the experiments because of too few information.

3.4 Link Prediction Experiment

This is an extension of the SVD-GCN application. In the previous sections, the proposed SVD-GCN is applied to do the node classification tasks while in the following section, it will be applied to do the link prediction task which is also a common edge-level learning task.

3.4.1 Background and Motivation

There are many applications of graph link prediction in the real world, for example, product recommendation in the online shopping website, friend recommendation in the social network and knowledge graph completion (Cai et al., 2021). While there have been some research works in link prediction tasks such as the link existence prediction on the graph data (Liben-Nowell et al., 2007; Schlichtkrull et al., 2018; Al Hasan et al., 2006; Lü & Zhou, 2011). Many heuristic methods have been presented and proposed to measure the similarity between the two connected nodes and then perform the link existence prediction tasks. However, most of the heuristic architectures are usually designed for certain network dataset specifically, and almost all of the heuristic methods hold strong assumption on when the two nodes have high possibility to have edge in between (Cai et al., 2021). Therefore, the heuristic approaches are quite limited to be utilized on different graph datasets with different conditions.

To address the issues mentioned above, the “SEAL” architecture was proposed and presented to automatically learn the heuristic functions from the target nodes’ h-hop neighbourhood in the graph data and this model is able to extract the local enclosing subgraphs which are centred on the two target nodes and then learn and output a function mapping the subgraph patterns and information, then perform the link

existence prediction task based on the topology of the local enclosing subgraphs (Zhang & Chen, 2018). In this research work, researchers transform and convert the link prediction task into a graph classification task and present the "SEAL" structure which is a graph neural network structure for link prediction, and this novel model has been proved to outperform all heuristic methods, latent feature approaches as well as the state-of-the-art Weisfeiler-Lehman Neural Machine (WLNM) (Zhang & Chen, 2018).

Compared to the research works on the normal graph data's link prediction, the literatures on directed graph data's link prediction are very limited. For a directed graph data, there are many sub-tasks under the category of edge-level learning task, not only that the existence of the link could be predicted, but also the direction of the link could be predicted which is a very important and informative learning task because the direction information contains significant relationship information between the two connected nodes (He et al., 2022). Therefore, for directed graph data, two more edge-level experiments will be conducted, one is the experiment to predict the direction of the edge of the vertices pair u, v, for which for which either $(u, v) \in \mathcal{E}$ or $(v, u) \in \mathcal{E}$; while the other experiment is to perform a three-class classification, which is to classify and predict if an edge between two target node is $(u, v) \in \mathcal{E}$, $(v, u) \in \mathcal{E}$ or $(v, u), (u, v) \notin \mathcal{E}$ (He et al., 2022).

3.4.2 Experimental Protocol

3.4.2.1 Datasets and Baselines

Because there are not many existing research works and deep learning architectures specifically designed for directed graph data, while researchers in the paper He et al. (2022) conducted link prediction experiments utilizing DiGCN (Tong et al., 2020),

DGCN (Tong et al., 2020), MagNet (Zhang et al., 2021) and DiGCN-IB (Tong et al., 2020). These four architectures are the benchmark structures specifically for directed graph data. Then SVD-GCN method will be applied on the same digraph datasets and results will be compared with the four models listed above. For convenience, the results for these four models are copied from the paper He et al. (2022). There are five digraph datasets in this experiment, among which the Cora_ml and Citeseer datasets are the same as the ones used in the previous node classification task and are both available from the link: https://github.com/flyingtango/DiGCN, while the other three WebKB datasets are available on the open-source website as well.

In the following part, some brief description regarding the datasets will be provided and the basic statistics will be shown in the Table 5:

Cora_ml (Bojchevski & Günnemann, 2017): This is a subset of the dataset that is extracted from the original classic citation network dataset Cora and Cora_ml is a directed graph dataset.

Citeseer (Yang et al., 2016): It is also a popular citation network and its structure is similar with Cora_ml's, whose nodes represent the publications and papers and edges represent the citation relationship. In Citeseet, the nodes are classified into six classes.

WebKB-Cornell & **WebKB-Texas** & **WebKB-Wisconsin**[2]: These three datasets are all from the online Alchemy WebKB dataset. And this WebKB comprises of seven classes of hyperlinks and web pages from the computer science departments of four universities which are The University of Washington, The University of Texas, Cornell

Table 5 Statistics of Datasets

[2] https://lig-membres.imag.fr/grimal/data.html

University and The University of Wisconsin. In this experiment, only three of them will be utilized, which are Texas, Cornell and Wisconsin.

Datasets	# Node	# Edges	# Classes	# Features
Cora_ml	2,995	8,416	7	2,879
Citeseer	3,312	4,715	6	3,703
WebKB-Cornell	183	295	5	1703
WebKB-Texas	183	309	5	1703
WebKB-Wisconsin	251	499	5	1703

3.4.2.2 Experimental Setup

In the graph link prediction task, the hyperparameters are set at follows: the basic epoch is 100, since we have tried several epoch values and found that the best result appears within 100 epochs in each rep, so it is not necessary to use 200 epochs in each rep. The two-level framelets (L=1) is also utilized and the dilation scale in the framelet is set to be 1.1 as usual. The number of hidden features is tested for 16, 32, 64 or 128 and is tested and adjusted during the experiment to make sure that the numbers are set to give the best output for each dataset. The dropout ratio is set to be 0.3 by default; while the framelet modulation function could be set to Entropy, Linear or Sigmoid. Usually if the framelet is Entropy, α is usually set to be 0.5 but if the framelet is Sigmoid, α will be set to be 20; if the framelet type is Linear, then any value of α listed above can be utilized.

As not only link existence prediction task will be performed, but also the link direction prediction will be done to test the SVD-GCN architecture's performance in edge-level tasks, thus there are two new parameters in the parameter setting part, one is called "task" and the other one is called "num_class_link". So when the experiment is about link existence prediction, then "task" should be set to 1 while "num_class_link" is not necessary to be set up because this is a parameter related to the link direction prediction.

When the experiment is about the two-class link direction prediction, the “task” should be set to 2 and “num_class_link” should be 2; but when the experiment is to predict the three-class link prediction task, then “task” should be 2 and the “num_class_link” should be set to 3.

3.4.3 Result Analysis

All the results for three prediction experiments are shown below in the three tables and for each dataset, the highest accuracy rate among all five architectures is bolded and highlighted in the tables. It is very clear that the proposed SVD-GCN architecture achieves remarkable performances in all three experiments. While in the link existence prediction task, the accuracy of SVD-GCN is the highest among all 5 datasets. In the link direction prediction experiment, SVD-GCN still achieves the highest accuracy in dataset WebKB-Cornell, WebKB-Texas and Citeseer while its performance in WebKB-Wisconsin and Cora_ml though is not the best, still ranked the second among all the structures. In the three-class link prediction, SVD-GCN still achieves the best performance in all the datasets except Cora_ml. Overall, the proposed SVD-GCN architecture outperforms the state-of-the-art benchmark models in the link prediction tasks for directed graph data.

Table 6 Direction Prediction (%)

Models	Cora_ml	Citeseer	WebKB-Cornell	WebKB-Texas	WebKB-Wisconsin
DGCN	83.2±0.1	81.4±0.3	74.9±0.5	82.3±0.7	82.3±0.7
DiGCN	85.5±0.1	85.5±0.1	82.6±0.4	89.9±0.6	88.0±0.4
DiGCN- IB	86.5±0.1	87.0±0.1	82.9±0.4	88.8±0.6	88.4±0.4
Magnet	**88.4±0.1**	88.8±0.2	83.1±0.3	87.8±0.7	**88.7±0.5**
SVD-GCN	87.46±0.5	**90.21±0.1**	**90.6±0.4**	**91.2±0.3**	88.1±0.4

Table 7 Existence Link Prediction (%)

Models	Cora_ml	Citeseer	WebKB-Cornell	WebKB-Texas	WebKB-Wisconsin
DGCN	74.8±0.2	69.5±0.2	62.5±0.11	67.5±0.6	71.9±0.7
DiGCN	75.7±0.1	72.8±0.1	74.2±0.8	72.7±0.6	73.7±0.5
DiGCN- IB	73.8±0.1	73.4±0.1	75.0±0.4	76.5±0.8	75.7±0.5
Magnet	76.1±0.1	72.4±0.2	73.2±0.6	70.7±0.8	75.9±0.3
SVD-GCN	**77.16±0.1**	**77.29±0.1**	**76.0±0.7**	**80.1±0.5**	**79.8±0.6**

Table 8 Three Classes Link Prediction (%)

Models	Cora_ml	Citeseer	WebKB-Cornell	WebKB-Texas	WebKB-Wisconsin
DGCN	66.2±0.1	63.5±0.1	60.7±0.4	71.3±0.4	69.3±0.3
DiGCN	65.4±0.1	59.0±0.1	62.6±0.7	71.3±0.4	72.9±0.1
DiGCN- IB	63.1±0.1	58.6±0.3	55.8±0.4	70.4±0.2	69.8±0.4
Magnet	**66.7±0.0**	62.9±0.1	64.1±0.3	72.0±0.3	70.4±0.1
SVD-GCN	64.6±0.6	**68.55±0.1**	**69.1±0.3**	**75.4±0.3**	**72.8±0.2**

3.4.4 Conclusion and Discussion

In this project, we extend to apply the proposed SVD-GCN on the link prediction tasks and based on the results of the three experiments, it has been proved that SVD-GCN could also achieve remarkable performances in edge-level learning tasks for digraph data. It further demonstrates that this proposed novel SVD-GCN structure is

convincingly useful and appropriate when it comes to handle the directed graph data and could be applied to address the real-world issues.

Chapter 4

Conclusion

This dissertation proposed a novel architecture, simple yet effective SVD-GCN and applied it on the learning tasks of node classification and link prediction. In this chapter, I will briefly summarize the contributions again, followed by a discussion about potential future research direction.

4.1 Main Contribution

The main contributions of this research thesis are included in Chapter 3 and can be five-fold:

1) It should be the first attempt to utilize the adjacency SVD for the graph convolution neural networks. Quasi-framelet decomposition is applied to better filter the graph signals on spectral domain and improve performance of the proposed SVD-GCN as well as its robustness when encountering high level of noise attack.
2) It is theoretically proved that the dual orthogonal systems provided by the SVD ensure the successful implementation of the graph signal decomposition and reconstruction which is in accordance with the spectral theory.

3) We investigate the method to scale up the proposed SVD-GCN for large graph datasets according to the Chebyshev Polynomial approximation via attaining the fast filtering for singular values while not running SVD.
4) Experimental results from the node classification task prove that the node representation learning method by the framelet SVD-GCN is effective and the proposed SVD-GCN achieves better performance compared to the state-of-the-art architectures for digraph data.
5) SVD-GCN is also applied to perform on the link prediction tasks, and the results further prove its effectiveness and remarkable performances on link prediction on the directed graph data against the state-of-the-art structures.

4.2 Future Research Directions

In this thesis, the node-level task (node classification) and edge-level task (link prediction) have been explored, there is another possible extension of applying the SVD-GCN which is to perform on the graph-level tasks. Multi-Relational MR-GCN was proposed in Huang et al. (2020), the researchers utilized the generalized tensor product computation into the normal graph convolution theory and define the multi-relational graph convolution operator (MR-GCO). While if the transformed tensor SVD (Song et al., 2019) is applied in the SVD-GCN, then it is possible to make the SVD-GCN better process the multi-relational graph data and the results could be outstanding as well because it has been proved that framelet SVD-GCN is really good and robust at the handling graph signals and make predictions for the graph classification task.

Reference

Al Hasan, M., Chaoji, V., Salem, S., & Zaki, M. (2006, April). Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security* (Vol. 30, pp. 798-805).

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164.*

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data, 8, 1-74.*

Asif, N. A., Sarker, Y., Chakrabortty, R. K., Ryan, M. J., Ahamed, M. H., Saha, D. K., ... & Tasneem, Z. (2021). Graph neural network: A comprehensive review on non-euclidean space. *IEEE Access, 9, 60588-60606.*

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. *Advances in neural information processing systems, 29.*

Balcilar, M., Guillaume, R., Héroux, P., Gaüzère, B., Adam, S., & Honeine, P. (2021, May). Analyzing the expressive power of graph neural networks in a spectral perspective. In *Proceedings of the International Conference on Learning Representations (ICLR).*

Balcilar, M., Renton, G., Héroux, P., Gauzere, B., Adam, S., & Honeine, P. (2020). Bridging the gap between spectral and spatial domains in graph neural networks. *arXiv preprint arXiv:2003.11702.*

Bojchevski, A., & Günnemann, S. (2017). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815.*

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine, 34(4), 18-42.*

Cai, L., Li, J., Wang, J., & Ji, S. (2021). Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Charikar, M., Chatziafratis, V., Niazadeh, R., & Yaroslavtsev, G. (2019, April). Hierarchical clustering for euclidean data. In *The 22nd International Conference on Artificial Intelligence and Statistics* (pp. 2721-2730). PMLR.

Chen, J., Ma, T., & Xiao, C. (2018). Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247.*

Chiang, W. L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C. J. (2019, July). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 257-266).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems, 29.*

Dong, B. (2017). Sparse representation on graphs by tight wavelet frames and applications. *Applied and Computational Harmonic Analysis, 42(3), 452-479.*

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., & Papalexakis, E. E. (2020, January). All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 169-177).*

Gao, H., Wang, Z., & Ji, S. (2018, July). Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1416-1424).

Gavili, A., & Zhang, X. P. (2017). On the shift operator, graph frequency, and optimal filtering in graph signal processing. *IEEE Transactions on Signal Processing, 65(23), 6303-6318.*

Gers, F. A., & Schmidhuber, J. (2000, July). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* (Vol. 3, pp. 189-194). IEEE.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, July). Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263-1272). PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM, 63(11), 139-144.*

Gori, M., Monfardini, G., & Scarselli, F. (2005, July). A new model for learning in graph domains. In Proceedings. *2005 IEEE international joint conference on neural networks* (Vol. 2, No. 2005, pp. 729-734).

Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864).

Into the Wild: Machine Learning In Non-Euclidean Spaces · Stanford DAWN. (2019, October 10). Retrieved February 15, 2023, from https://dawn.cs.stanford.edu/2019/10/10/noneuclidean/

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems, 30.*

He, Y., Zhang, X., Huang, J., Cucuringu, M., & Reinert, G. (2022). PyTorch Geometric Signed Directed: A Survey and Software on Graph Neural Networks for Signed and Directed Graphs. *arXiv preprint arXiv:2202.10793.*

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9(8), 1735-1780.*

Hu, B., Zhang, Z., Shi, C., Zhou, J., Li, X., & Qi, Y. (2019, July). Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 946-953).

Huang, Z., Li, X., Ye, Y., & Ng, M. K. (2020). MR-GCN: Multi-Relational Graph Convolutional Networks based on Generalized Tensor Product. In *IJCAI* (pp. 1258-1264).

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*

Klicpera, J., Bojchevski, A., & Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997.*

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation, 1(4), 541-551.*

Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing, 67(1), 97-109.*

Li, C., Qin, X., Xu, X., Yang, D., & Wei, G. (2020). Scalable graph convolutional networks with fast localized spectral filter for directed graphs. *IEEE Access, 8, 105634-105644.*

Li, R., Wang, S., Zhu, F., & Huang, J. (2018, April). Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1).

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926.*

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems.*

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology, 58(7), 1019-1031.*

Lim, L. H. (2020). Hodge Laplacians on graphs. *Siam Review, 62(3), 685-715.*

Lü, L., & Zhou, T. (2011). Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications, 390(6), 1150-1170.*

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025.*

Ma, Y., Hao, J., Yang, Y., Li, H., Jin, J., & Chen, G. (2019). Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990.*

Madhu, P., Kosti, R., Mührenberg, L., Bell, P., Maier, A., & Christlein, V. (2019, October). Recognizing characters in art history using deep learning. In *Proceedings of the 1st Workshop on Structuring and Understanding of Multimedia heritage Contents* (pp. 15-22).

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5115-5124).

Mujkanovic, F., Geisler, S., Günnemann, S., & Bojchevski, A. (2022). Are Defenses for Graph Neural Networks Robust?. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022).*

Nt, H., & Maehara, T. (2019). Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550.*

Onuki, M., Ono, S., Shirai, K., & Tanaka, Y. (2017). Fast singular value shrinkage with Chebyshev polynomial approximation based on signal sparsity. *IEEE Transactions on Signal Processing, 65(22), 6083-6096.f*

Peng, S., Sugiyama, K., & Mine, T. (2022, October). SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (pp. 1625-1634).*

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701-710).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems, 28.*

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637.*

Rossi, E., Frasca, F., Chamberlain, B., Eynard, D., Bronstein, M., & Monti, F. (2020). Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198, 7, 15.*

Sandryhaila, A., & Moura, J. M. (2013). Discrete signal processing on graphs. *IEEE transactions on signal processing, 61(7), 1644-1656.*

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks, 20(1), 61-80.*

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. V. D., Titov, I., & Welling, M. (2018, June). Modeling relational data with graph convolutional networks. In *European semantic web conference* (pp. 593-607). Springer, Cham.

Shchur, O., Mumme, M., Bojchevski, A., & Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868.*

Song, G., Ng, M. K., & Zhang, X. (2019). Robust tensor completion using transformed tensor svd. *arXiv preprint arXiv:1907.01113.*

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks, 8(3), 714-735.*

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015, May). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067-1077).

Tong, Z., Liang, Y., Sun, C., Li, X., Rosenblum, D., & Lim, A. (2020). Digraph inception convolutional networks. *Advances in neural information processing systems, 33, 17907-17918.*

Tran, D. V., Navarin, N., & Sperduti, A. (2018, November). On filter size in graph convolutional networks. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1534-1541). IEEE.

van Dam, E. R., & Omidi, G. R. (2018). Directed strongly walk-regular graphs. *Journal of Algebraic Combinatorics, 47(4), 623-639.*

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903.*

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2019). Deep Graph Infomax. *ICLR (Poster), 2(3), 4.*

Wang, H., & Raj, B. (2017). On the origin of deep learning. *arXiv preprint arXiv:1702.07800.*

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019, May). Heterogeneous graph attention network. In *The world wide web conference* (pp. 2022-2032).

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019, May). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861-6871). PMLR.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144.*

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems, 32(1), 4-24.*

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. *arXiv preprint arXiv:1810.00826.*

Yan, S., Xiong, Y., & Lin, D. (2018, April). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence.*

Yang, M., Zheng, X., Yin, J., & Gao, J. (2022). Quasi-Framelets: Another Improvement to GraphNeural Networks. *arXiv preprint arXiv:2201.04728.*

Yang, Z., Cohen, W., & Salakhudinov, R. (2016, June). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning* (pp. 40-48). PMLR.

Yun, S., Jeong, M., Kim, R., Kang, J., & Kim, H. J. (2019). Graph transformer networks. *Advances in neural information processing systems, 32.*

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in neural information processing systems, 31.*

Zhang, X., He, Y., Brugnone, N., Perlmutter, M., & Hirn, M. (2021). Magnet: A neural network for directed graphs. *Advances in Neural Information Processing Systems, 34, 27003-27015.*

Zheng, X., Zhou, B., Gao, J., Wang, Y. G., Lió, P., Li, M., & Montúfar, G. (2021). How framelets enhance graph neural networks. *arXiv preprint arXiv:2102.06986.*

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open, 1, 57-81.*

Zhuang, C., & Ma, Q. (2018, April). Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference* (pp. 499-508).

Zou, C., Han, A., Lin, L., & Gao, J. (2022). A Simple Yet Effective SVD-GCN for Directed Graphs. *arXiv preprint arXiv:2205.09335.*