THE UNIVERSITY OF SYDNEY

THE UNIVERSITY OF SYDNEY BUSINESS SCHOOL

DISCIPLINE OF BUSINESS ANALYTICS

(Honours Thesis)

A Magnetic Framelet-Based Convolutional Neural Network for Directed Graphs

---

Author

Lequan Lin

SID

[REDACTION]

Supervisor

Professor Junbin Gao

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

Bachelor of Advanced Studies (Honours)

November 2022

STATEMENT OF ORIGINALITY

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at the University of Sydney or at any other educational institution, except where due acknowledgement is made in the thesis.

Any contribution made to the research by others, with whom I have worked at the University of Sydney or elsewhere, is explicitly acknowledged in this thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the projects design and conception or in style, presentation and linguistic expression is acknowledged.

Signature of Author

Lequan Lena LIN

November 2022

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the following people who supported me during my Honours year.

I would like to thank my supervisor, Professor Junbin Gao, for his dedicated support, relentless help, and invaluable suggestions during the running of my Honours project. I would also like to extend my deepest appreciation to the Honours program team at the University of Sydney Business School. Especially, I would like to acknowledge the assistance of the Honours Program Director Associate Professor Boris Choy, the Business Analytics Honours Coordinator Professor Richard Gerlach, lecturers of my Honours courses Dr Andreea Constantin, Dr Nam Ho-Nguyen, Professor Artem Prokhorov, and Associate Professor Jie Yin. Thank you for all the unwavering guidance, constructive advice, and precious knowledge, which makes this year so enjoyable and memorable.

Thanks also to my family and friends: my parents, Associate Professor Di Lin and Professor Hua Han, my boyfriend, Yibai Li, and my friends, Patrick Conroy, Huidong Liang, Jerry Jiali Weng, Ruohan Xia, and Renjie Yu. Thank you for your unconditional company and unrelenting support during my Honours year. This work cannot be finished without you.

Lastly, I would like to acknowledge the valuable suggestion of Xitong Zhang, the author of "MagNet: A Neural Network for Directed Graphs", on how to reproduce the MagNet experiment results.

CONTRIBUTION

Paper List

The following paper is based on this thesis:

- **Lequan Lin** and Junbin Gao, A Magnetic Framelet-Based Convolutional Neural Network for Directed Graphs, submitted to *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, arXiv:2210.10993, 2023.

The author also contributes to the following paper during the Honours year:

- Chunya Zou, Andi Han, **Lequan Lin**, Ming Li, and Junbin Gao, A Simple Yet Effective SVD-GCN for Directed Graphs, submitted to *IEEE Transactions on Artificial Intelligence*, arXiv:2205.09335, 2022.

Contents

List of Tables

List of Figures

List of Notations

The following list summarizes the most important notations that will be used in the thesis. They are only a fraction of all the notations we will use. For other less important notations, we refer to their definitions in the main contents of the thesis.

$\mathcal{G}\{\mathcal{V}, \mathcal{E}\}$	An undirected graph
$\mathcal{G}_d\{\mathcal{V}, \mathcal{E}\}$	A directed graph
$\mathcal{V}$	A set of graph vertices
v_i	A vertex with index number i
$\mathcal{E}$	A set of graph edges
(v_i, v_j)	An edge between vertex v_i and vertex v_j
$\mathbf{A}$	The adjacency matrix
$\mathbf{D}$	The degree matrix
$\mathcal{L}$	The graph Laplacian
$\mathbf{A}_{sym}$	The symmetric adjacency matrix
$\mathbf{A}_{skew}$	The skew-symmetric adjacency matrix
$\mathcal{L}^{(q)}$	The magnetic Laplacian
q	The electric charge parameter
$\mathbf{U}$	The matrix of (magnetic) Laplacian eigenvectors
u_k	The k^{th} eigenvector of (magnetic) Laplacian
$\mathbf{\Lambda}$	The matrix of (magnetic) Laplacian eigenvalues
λ_k	The k^{th} eigenvalue of (magnetic) Laplacian

g_ω	The learnable filter in graph convolution
$\{g_\omega * x\}$	Graph convolution
$Z = \{\zeta_0, ..., \zeta_R\}$	A set of scaling functions
$z = \{z_0, ..., z_R\}$	A filter bank
n	Framelet position
s	Framelet dilation level
$\rho_{n,s}^{(q)}$	The low-pass magnetic graph framelet at dilation level s and position v_n
$\varrho_{n,s,r}^{(q)}$	The r^{th} high-pass magnetic graph framelet at dilation level s and position v_n
$MGFS(Z, z; \mathcal{G}_d)$	The magnetic graph framelet system for a directed graph $\mathcal{G}_d$
$F_x^{(q)}$	The magnetic framelet coefficients of a single graph signal x
$\mathcal{F}_{r,s}^{(q)}$	The magnetic framelet transform operator
$\mathcal{F}^{(q)}$	The magnetic framelet transform matrix
$\mathcal{F}^{(q)^*}$	The magnetic framelet inversion matrix
$\tilde{\mathcal{F}}_{r,s}^{(q)}$	The fast magnetic framelet transform operator
$\tilde{\mathcal{F}}^{(q)}$	The fast magnetic framelet transform matrix
$\tilde{\mathcal{F}}^{(q)^*}$	The fast magnetic framelet inversion matrix

Abstract

Recent years have witnessed the surging popularity among studies on directed graphs (digraphs) and digraph neural networks. With the unique capability of encoding directional relationships, digraphs have shown their superiority in modelling many real-life applications, such as citation analysis and website hyperlinks. Spectral Graph Convolutional Neural Networks (spectral GCNNs), a powerful tool for processing and analyzing undirected graph data, have been recently introduced to digraphs. Although spectral GCNNs typically apply frequency filtering via Fourier transform to obtain representations with selective information, research shows that model performance can be enhanced by framelet transform-based filtering. However, the massive majority of such research only considers spectral GCNNs for undirected graphs. In this thesis, we introduce Framelet-MagNet, a magnetic framelet-based spectral GCNN for digraphs. The model adopts magnetic framelet transform which decomposes the input digraph data to low-pass and high-pass frequency components in the spectral domain, forming a more sophisticated digraph representation for filtering. Digraph framelets are constructed with the complex-valued magnetic Laplacian, simultaneously leading to signal processing in both real and complex domains. To our best knowledge, this approach is the first attempt to conduct framelet-based convolution on digraph data in both real and complex domains. We empirically validate the predictive power of Framelet-MagNet via various tasks, including node classification, link prediction, and denoising. Besides, we show through experiment results that Framelet-MagNet can outperform the state-of-the-art approaches across several benchmark datasets.

Word count including abstract, chapters, in-text citations, tables, figures, formulas, and references based on Overleaf: 20,749

Chapter 1

Introduction

Our research will focus on the extension of framelets to directed graph neural networks. More specifically, we will design a magnetic framelet-based Graph Convolutional Neural Network for directed graphs. In this Chapter, we will establish a background for our research while identifying the research gap that motivates our study. In Section 1.1, we will have an introduction to Graph Convolutional Neural Networks. Then, in Section 1.2, we will introduce directed graphs, their applications, and relevant challenges. To highlight our motivation, we will explain why we pay special attention to graph framelets in Section 1.3. Eventually, in Section 1.4, we will present our research questions and have an overview of the thesis structure.

1.1 Graph Convolutional Neural Network

With the exclusive ability to capture relational information in real-world data, graphs become one of the most popular research topics in the machine-learning community. In the early study of graph algorithms, researchers faced two major challenges. Firstly, graphs do not have consistent structures. Many variates can lead to inconsistency among graph instances, such as the number of nodes and the type of edges. Secondly, unlike images whose pixels have a definitive position in the grid, graph nodes do not have a specific order for analysis and processing. So, graph algorithms should be node-order equivalent, namely, not depending on the order of nodes in processing and analysis. Inspired by the fact that deep learning has illustrated its power in

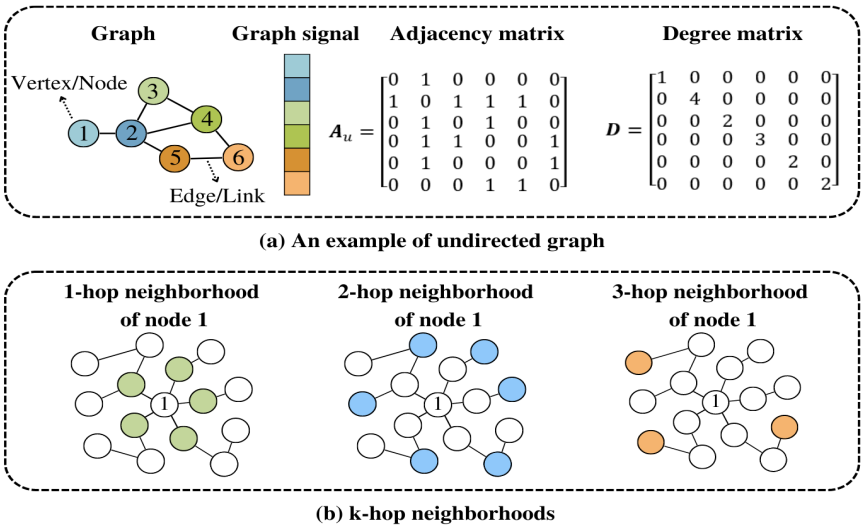

Figure 1.1: Illustration of some fundamental graph concepts. (a) An undirected graph with 6 nodes and corresponding graph signal, adjacency matrix, and degree matrix. (b) k-hop neighborhoods of node 1.

numerous applications, researchers proposed Graph Neural Networks (GNNs) [2, 3]. It has been validated that GNNs can process complicated graph topology with no dependence on the node order [4]. The most important branch of GNNs is the Graph Convolutional Neural Networks (GCNNs).

Before we introduce the basic concepts of GCNNs, we will define some graph notations (visualized illustrations are shown in **Figure 1.1**). An undirected graph can be denoted as $\mathcal{G}\{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is a set of N vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. Conventionally, vertices and edges are also known as nodes and links. For two nodes $v_i, v_j \in \mathcal{V}$, we denote (v_i, v_j) as an edge from v_i to v_j, and (v_j, v_i) as an edge of the other direction. If both $(v_i, v_j) \in \mathcal{E}$ and $(v_j, v_i) \in \mathcal{E}$, we say there is an undirected edge between v_i and v_j. An undirected graph is defined as a graph whose edges are all undirected. The graph topological structure is stored in the adjacency matrix $\mathbf{A}$. It is an $N \times N$ square matrix, in which $\mathbf{A}(i, j) = 1$ indicates the existence of an edge from vertex v_i to vertex v_j, and $\mathbf{A}(i, j) = 0$ otherwise. For the sake of simplify, in this thesis, we will not consider weighted graphs nor graphs with self-loop edges. Nevertheless, unless otherwise stated, computation in the remainder of this work can be smoothly extended to graphs with self-loops or non-negative weights.

A graph normally have a node attribute matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$ whose columns $x_1, ..., x_d$ are d graph signals. From a mathematical perspective, a graph signal can be regarded as a mapping from vertices $\mathcal{V}$ to a set of numbers as $x : \mathcal{V} \to \mathbb{R}$.

The k-hop neighborhood of a node is composed of nodes that are connected to this node up to the k-hop distance. A node v_j is the k-hop neighbor of node v_i if v_i can reach v_j through k edges. We denote the k-hop neighborhood of v_i as $\mathcal{N}(v_i, k)$. The degree-matrix $\mathbf{D}$ is a diagonal matrix whose diagonal entries record the number of 1-hop neighbors of each node. Mathematically, we can derive $\mathbf{D}$ from the adjacency matrix $\mathbf{A}$ by $\mathbf{D}(i,i) = \sum_{v_j \in \mathcal{V}} \mathbf{A}(i,j)$.

Generally, machine learning models for graphs are designed by carefully taking account of the graph structure and the irregularity of graph data. To integrate graph structure and graph signals, GCNNs conduct the aggregation of neighboring node information in a manner analogous to traditional convolutional networks for images [2, 3, 5]. This process is accomplished by the graph convolution in each convolutional layer. In addition, graph convolution also tolerates signal filtering via applying learnable and/or pre-designed filters. Hence, we can understand graph convolution as a convolution between the input signals and a filter. Depending on how graph convolution is defined, GCNNs can be categorized into two classes, spatial-based and spectral-based. The major difference between spatial and spectral convolutions exists in the design of their filtering system. For simplicity, the following discussions will be based on a single graph signal x, which is an N-dimensional vector. We will let $x(i)$ denote the signal value at node v_i.

Based on [2], spatial GCNNs use vertex filtering while spectral GCNNs use frequency filtering. With vertex filtering, the graph signals are processed in the graph domain, where the output is a linear combination of the signal values at each node and its associated neighborhood. The most basic mathematical expression of spatial graph convolution with vertex filtering is denoted as

$$\omega * x(i) = \omega_{i,i} x(i) + \sum_{v_j \in \mathcal{N}(v_i,k)} \omega_{i,j} x(j), \tag{1.1}$$

where ω is a learnable filter that contains all the weights $\{\omega_{i,j}\}$, and $*$ is the convolution symbol.

Frequency filtering, on the other hand, processes graph signals in the spectral domain. Based on the graph spectral theory, graph signals can be converted to their representations in the spectral domain via the graph Fourier transform. This process is assisted by the eigendecomposition of graph Laplacian. Since relevant literature usually adopts the normalized graph Laplacian, we will follow this convention and define the graph Laplacian as

$$\mathcal{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{1.2}$$

It is well-known that the Laplacian of an undirected graph is symmetric and positive semi-definite. So, applying eigendecomposition, we have $\mathcal{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$, where $\mathbf{U}$ is an $N \times N$ matrix whose columns are eigenvectors $u_k, k = 0, ..., N-1$, and $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal entries are eigenvalues $\lambda_k, k = 0, ..., N-1$. Now, we can write the Fourier transform function as $\hat{x} = \mathbf{U}^\intercal x$, where $\hat{x}$ is the representation of x in the spectral domain, known as the Fourier coefficients of x. We can reconstruct x from $\hat{x}$ with the Fourier inversion function, given by $x = \mathbf{U}\hat{x}$. From the computation, we can see that the Fourier transform is intrinsically a projection of the graph signal to the basis formed by orthonormal eigenvectors of the graph Laplacian. With the Fourier transform, we can realize the frequency filtering of spectral coefficients in the spectral domain.

The first notable spectral graph convolution with frequency filtering was proposed by Bruna et al. [6], denoted as

$$g_\omega * x = \mathbf{U} g_\omega \mathbf{U}^\intercal x, \tag{1.3}$$

where g_ω is an $N \times N$ diagonal matrix, $\text{diag}(\omega)$. $\omega \in \mathbb{R}^N$ is considered as a set of learnable weights. However, the computational cost of this convolution is very expensive. This is mainly due to the multiplication with $\mathbf{U}$ and the eigendecomposition of $\mathcal{L}$. For fast computation, Kipf and Welling [7] proposed ChebNet, which adopts Chebyshev polynomial-based spectral convolution. We firstly assume that g_ω is a function of the eigenvalues of $\mathcal{L}$ (i.e., $g_\omega(\Lambda)$), which means the diagonal entries of the filter matrix now becomes $g_\omega(\lambda_k)$, for $k = 0, ..., N-1$. Then, we can approximate $g_\omega(\Lambda)$ with Chebyshev polynomials and define the convolution as

$$g_\omega * x = \sum_{\ell=0}^{L} \omega_\ell \mathcal{T}_\ell(\tilde{\mathcal{L}})x, \tag{1.4}$$

where $\tilde{\mathcal{L}} = \frac{2}{\lambda_{max}}\mathcal{L} - \mathbf{I}$ with λ_{max} being the largest eigenvalue of $\mathcal{L}$. The Chebyshev polynomials are defined as $\mathcal{T}_\ell(\tilde{\mathcal{L}}) = 2\tilde{\mathcal{L}}\mathcal{T}_{\ell-1}(\tilde{\mathcal{L}}) + \mathcal{T}_{\ell-2}(\tilde{\mathcal{L}})$ with $\mathcal{T}_0(\tilde{\mathcal{L}}) = \mathbf{I}$ and $\mathcal{T}_1(\tilde{\mathcal{L}}) = \tilde{\mathcal{L}}$ for $\ell \geq 2$. To further avoid overfitting, we can set $\ell = 1$, $\lambda_{max} = 2$, and $\omega = \omega_0 = -\omega_1$, such that

$$g_\omega * x = \omega\left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)x. \tag{1.5}$$

Furthermore, to avoid the exploding/vanishing gradient problem, spectral GCNNs often adopts $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and its corresponding degree matrix $\tilde{\mathbf{D}}$. This produces the simplified convolution used in GCN [7]:

$$g_\omega * x = \omega\left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\right)x. \tag{1.6}$$

Now we can define the convolutional layers of spectral GCNNs and extend to multiple graph signals stored in $\mathbf{X} \in \mathbb{R}^{N\times d}$. Generally, the convolutional layers have two components: graph convolution and non-linear activation function. For the basic graph convolution defined in equation (1.3), we write the i^{th} layer of the network as

$$\mathbf{X}_i = \sigma(g_{\omega_i} * \mathbf{X}_{i-1}) = \sigma(\mathbf{U}\mathrm{diag}(\omega_i)\mathbf{U}^\intercal(\mathbf{X}_{i-1}\mathbf{W}_i)), \tag{1.7}$$

where σ is a non-linear activation function, $g_{\omega_i} = \mathrm{diag}(\omega_i)$ is a learnable filter, $\mathbf{X}_{i-1}$ is an $N \times d_{i-1}$ feature matrix with $\mathbf{X}_0$ being the original graph feature matrix $\mathbf{X}$, and $\mathbf{W}_i$ is a $d_{i-1} \times d_i$ learnable matrix, where d_{i-1} and d_i are dimensions of the input and output channels.

With the simplified convolution in equation (1.6), the convolutional layers of GCN let the learnable matrix $\mathbf{W}$ absorb the weight ω, generating a simpler mathematical expression as

$$\mathbf{X}_i = \sigma(g_{\omega_i} * \mathbf{X}_{i-1}) = \sigma\left(\left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\right)\mathbf{X}_{i-1}\mathbf{W}_i\right). \tag{1.8}$$

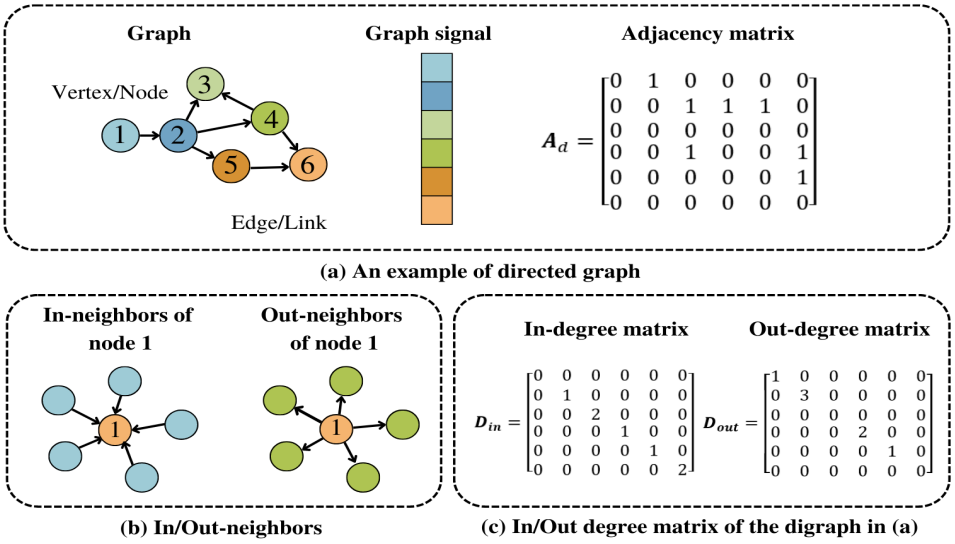

Figure 1.2: Illustration of digraphs. (a) A digraph with 6 nodes and its corresponding graph signal and adjacency matrix. (b) In-neighbors and out-neighbors of node 1. (c) In-degree and out-degree matrices of the digraph in (a).

1.2 Directed Graphs

Directed graphs (digraphs) are network-like graphs with directed edges (see **Figure 1.2 (a)** as an example). We denote a digraph as $\mathcal{G}_d(\mathcal{V}, \mathcal{E})$. For two nodes $v_i, v_j \in \mathcal{V}$, if $(v_i, v_j) \in \mathcal{E}$, then there is a directed edge from v_i to v_j. Recall that in undirected graphs, if $(v_i, v_j) \in \mathcal{E}$, then $(v_j, v_i) \in \mathcal{E}$ as well. In a strict digraph (i.e., oriented graph), however, if $(v_i, v_j) \in \mathcal{E}$, then $(v_j, v_i) \notin \mathcal{E}$. Since oriented graphs are not very common in real-life problems, when we say digraphs, we usually refer to mixed graphs, in which both directed and undirected edges exist.

In some real-world problems, the relationship between two objects is directional, thus the data are naturally represented as digraphs. Incorporating directional information in graph edges backbones the exploration of more insightful aspects of the underlying data, thus potentially providing more effective solutions. For example, in website hyperlink analysis [8, 9], if a hyperlink in website A leads to website B, it can be naturally modelled as a directed edge from node A to node B. With undirected graphs, however, we can only identify the existence of a hyperlink, which is far less informative. As another example, digraphs are also very useful in business decision-making, such as portfolio management [10]. To construct a good investment

portfolio, financial analysts often conduct diversification by including securities from different sectors. When a sector suffers from a severe downtrend, securities from other sectors will not be negatively impacted, which mitigates the overall risk of the portfolio. So, accurate sectorization is of significance. Usually, this is achieved by calculating correlations between assets, which only indicates the existence of relevance. To produce a more comprehensive picture of the security market, Abrams et al. [10] suggest that we can model this problem as a digraph with nodes being securities. A directed edge exists from security A to security B if price changes of A will lead to price variations of B. Now, we may investigate the direction of impact for better sectorization. Other applications of digraphs include but not limited to citation analysis [11], financial crisis detection [12], traffic condition prediction [13, 14], social network analysis [15], etc.

What can we do after modelling these problems as digraphs? Typically, we can conduct node-level tasks, link-level tasks, and graph-level tasks [3]. Here we concentrate on node-level and link-level tasks. The most common node-level task is node classification, in which we try to categorize nodes into several prescribed classes. In the aforementioned portfolio example, sectorization can be considered as a node-classification task. Link-level tasks usually involve link existence prediction and link direction prediction. Link existence prediction aims to detect whether an edge exists between two nodes, while link direction prediction targets at finding the potential directional relationship between two nodes. For instance, in hyperlink analysis, we may predict whether a directed edge exists between two websites. Suppose that two websites are predicted to be connected. If there is actually no hyperlink between them, we may consider adding one to improve the hyperlink structure. Such improvement may enhance the convenience of websites and upgrade visitor experience. GCNNs have been proven to be a powerful tool for conducting these tasks for undirected graphs[2, 3, 5]. Driven by this fact, much recent research engages with introducing GCNNs to digraph applications.

By construction, spatial GCNNs such as GraphSAGE [16], GAT [17], APPNP [18], and GIN [19] typically have natural applications on digraphs, because the input can be incorporated in the spatial convolution very easily [5]. Recall that spatial-based networks directly perform convolution in the graph domain by aggregating information from the k-hop neighborhoods. To encode edge directions, we can categorize the neighboring nodes as “in-neighbors” and

"out-neighbors" (see **Figure 1.2 (b)** for illustration). For two nodes v_i, v_j linked by a directed edge, if the edge direction is from v_i to v_j, then v_j is the out-neighbor of v_i. Otherwise, v_j is the in-neighbor of v_i. Hence, the model can capture information from two directions simultaneously by aggregating in-neighbors and out-neighbors separately in the convolutional layer. Mathematically, we can extend the graph convolution in equation (1.1) to digraphs as

$$\omega * x(i) = \omega_{i,i}x(i) + \sum_{v_j \in \mathcal{N}_{out}(v_i,k)} \omega_{i,j}x(j) + \sum_{v_j \in \mathcal{N}_{in}(v_i,k)} \omega_{j,i}x(j),$$

where $\mathcal{N}_{in}(v_i, k)$ ($\mathcal{N}_{out}(v_i, k)$) is the k-hop in(out)-neighborhood of v_i.

Despite the straightforward extension of spatial methods to digraph problems, criticisms have been drawn in three aspects. Firstly, it is pointed out in [20] that convolution with two neighborhoods usually adds significant computational costs compared with the original model. Likewise, Li et al. [14] suggest that their spatial-based model works better with sparse graphs due to computational consideration. In addition, Zhang et al. [21] suggest that spatial methods generally perform better with symmetrized adjacency matrices. Such matrices are constructed by replacing all the directed edges with undirected ones, leading to ignorance of directional information. Furthermore, Ma et al. [22] argue that most spatial approaches suffer from a lack of signal processing mechanisms, so their ability to extract useful information from graph data is very limited compared with spectral methods. Due to the aforementioned limitations, how to extend spectral GCNNs to digraph data becomes a popular research topic.

However, it is not easy to apply spectral GCNNs on digraphs. Recall that spectral convolution is supported by the eigendecomposition of graph Laplacian, which requires the Laplacian to be symmetric and positive semi-definite. But the digraph adjacency matrix is not symmetric. Accordingly, the digraph Laplacian is not symmetric either. A naive solution to this problem is to symmetrize the Laplacian by replacing directed edges with undirected ones, which effectively creates an undirected graph. As we have discussed, this approach is not a good option, because it discards all the directional information. Hence, researchers focus on designing symmetric alternatives that can preserve directional information [1, 20, 21, 22]. Magnetic Laplacian [21, 23] is one of the most successful instances. It is a complex-valued Hermitian matrix, whose real part

shows edge existence, and imaginary part indicates edge directions. Magnetic Laplacian-based digraph networks exploit magnetic Laplacian in classic spectral GCNN architectures and have demonstrated their power in various graph tasks [21].

1.3 Why Use Framelets?

The rationale behind magnetic Laplacian-based GCNNs is to incorporate magnetic Laplacian in traditional spectral GCNN architectures, such as ChebNet [24] and GCN [7]. All these architectures adopt frequency filtering, thus intrinsically assuming the role of graph Fourier transform.

Basically, a transform involves two domains, the original domain (e.g., the graph domain) and the new domain (e.g. the spectral domain)[1]. The purpose of transform is to develop transform coefficients, a representation of the input data in the new domain, for analysis and processing. Transform coefficients are often obtained by projecting signals to a spectral basis (e.g., the orthonormal eigenvector system). Normally, the processed coefficients will be converted back to the original domain by an inversion function. When processing topologically complicated data such as graphs, transform is of considerable significance, because many techniques are more expressive or more powerful in other domains [25].

Recall that the graph Fourier transform is realized by taking the inner product between Laplacian eigenvectors and graph signals. In the graph spectral theory, eigenvectors corresponding to larger eigenvalues can measure higher signal frequency [4]. This means that the projection of graph signals to these eigenvectors allows us to harvest high-frequency signal components. On the contrary, eigenvectors associated with small eigenvalues help us to explore low-frequency signal components. But, the major drawback of Fourier transform is that it only adopts a global basis for transform, which means although we can detect signal frequency, we cannot identify the position where it occurs.

To solve this problem, we need a more localized transform technique, that is, graph framelet

[1] Some literature may assign different names to the new domains after different transforms (e.g., Fourier domain after Fourier transform), but, for the sake of convenience, we refer them collectively as the spectral domain.

transform. Framelet transform is developed from wavelet transform. Wavelet transform uses graph wavelets as the basis. Generally speaking, graph wavelets are constructed through dilation and translation of a "mother wavelet", which is a single function centered at a certain node. Translation moves this function to other nodes in the graph, while dilation changes the scale of this function such that it can reflect different frequencies. Framelet transform upgrades this method by implementing multiple mother wavelets (also known as scaling functions in the framelet terminology), which enriches the diversity of transform bases.

Applying framelet transform instead of Fourier transform has the following advantages. Firstly, as we have mentioned, framelet transform not only enables the measurement of frequencies but also supports the identification of positions, namely, the node where a frequency component occurs. So, framelet coefficients are more sophisticated than Fourier coefficients, providing more informative graph representations for prediction. Secondly, graph framelets are sparse compared with Laplacian eigenvectors, which leads to a more efficient transform [26]. Lastly, quasi-framelet transform [27] tolerates "double regulation" on graph signals, which supports frequency filtering with both pre-designed and learnable filters.

There are several attempts to assemble GCNNs with the wavelet or framelet transform, however, the majority of works are only applicable to undirected graphs [26, 27, 28]. Although SVD-GCN [29] realizes digraph framelet transform via singular value decomposition (SVD) of the asymmetric digraph Laplacian, its theoretical rationale is very vague, for example, how the Laplacian singular values can be linked to the signal frequency in the SVD domain. In this thesis, we aim to design a framelet-based spectral GCNN for digraphs without discarding the role of Laplacian eigendecomposition. The framelet transform for digraphs will be accomplished with the support of the magnetic Laplacian. So, we name our model as **Framelet-MagNet**.

1.4 Thesis Organization and Research Questions

The structure of this thesis is as follows. In Chapter 2, we will have a review of two topics, spectral-based digraph GCNN and graph framelets. The review of graph framelets will include

the development of both wavelet and framelet theories and framelet-based GCNNs. Then, in Chapter 3, we will introduce our model, Framelet-MagNet, which is a magnetic framelet-based GCNN for digraphs. In addition, we will elaborate on how to conduct node classification and link prediction tasks with Framelet-MagNet. In Chapter 4, we will present the implementation details of our experiments and use the experiment results to validate the power of our model over a range of state-of-the-art models. Eventually, in Chapter 5, we will conclude the thesis, list current limitations and future works, and discuss relevant ethical considerations.

Here we further clarify how our research questions guide our research works.

Research Question 1. *How to construct a magnetic framelet-based GCNN for digraphs?*

To answer this question, we will introduce Framelet-MagNet in Chapter 3. The most important component in framelet-based spectral GCNNs is the framelet convolution, in which we firstly apply framelet transform to convert graph signals to the spectral domain for filtering, then reconstruct the processed data back to the graph domain with the framelet inversion function. Intuitively, we desire no information loss during the whole process, which means we expect the framelet transform to be "tight". Accordingly, we will design the Magnetic Graph Framelet Transform (MGFT), which is a tight framelet transform defined on digraphs. The fundamental principle of MGFT is to incorporate magnetic Laplacian in the traditional undecimated tight framelet transform on undirected graphs. For fast computation, we will propose Fast Magnetic Framelet Transform (FMFT) assisted by Chebyshev polynomial approximation. Eventually, we will assemble Framelet-MagNet with the magnetic framelet convolution supported by FMFT.

Research Question 2. *Can Framelet-MagNet outperform the state-of-the-art approaches?*

We will conduct experiments and present the results in Chapter 4. Our experiments consist of three parts, node classification, link prediction, and denoising. In each part, we will compare the performance of Framelet-MagNet with a range of state-of-the-art models across several benchmark datasets. In addition, we will pay special attention to the comparison between Framelet-MagNet and MagNet [21], which is a magnetic Laplacian-based GCNN with the Fourier transform. This will directly show the superiority of the framelet transform over the

Fourier transform in improving GCNN predictive performance. The metrics of our interest are average prediction accuracy and accuracy standard deviation across ten subsets randomly generated from each dataset, which are typical choices in GNN experiments.

Chapter 2

Literature Review

Our literature review will revolve around two topics, spectral GCNNs and the graph framelet theory.

Theoretically, if we symmetrize the digraph adjacency matrix by substituting directed edges with undirected edges, then most spectral GCNNs can be applied to digraph problems. However, as we have discussed before, this will lead to a leakage of important directional information, which may further lead to a deterioration in prediction accuracy. Therefore, our review in Section 2.1 will focus on spectral GCNNs that are specifically designed for digraphs. To conduct an informative review of digraph GCNNs, we will have a method-based discussion on the state-of-the-art approaches proposed in recent years. This will contribute to our understanding of how digraph spectral GCNNs work and demonstrate the difference between our method and the existing techniques. Besides, it will also build a theoretical foundation for our experiments since some of the networks mentioned will be adopted as baseline models.

Our review on the framelet transform in Section 2.2 will be around recent research on the wavelet/framelet theory in graph machine learning. Particularly, we will focus on framelet-based spectral GCNNs. Our review will indicate the power of framelet transform in the refinement of spectral GCNNs. As little endeavors have been taken to investigate the effectiveness of framelet transform in digraph neural networks, this review will highlight the importance of our research question.

2.1 Spectral-Based Digraph Convolutional Networks

The major challenge in designing spectral-based digraph GCNNs is how to conduct eigendecomposition with digraph Laplacian. The digraph Laplacian is asymmetric, so, it cannot provide real-valued eigenvalues and orthonormal eigenvectors required by the graph Fourier transform [5, 20]. Consequently, the extension of spectral methods to digraphs needs us to design a symmetric digraph adjacency/Laplacian. As we know, symmetrizing the digraph adjacency matrix by replacing directed edges with undirected edges is not a good option, because all directional information is discarded in this process. Therefore, a suitable alternative for digraph adjacency/Laplacian is expected to be symmetric while simultaneously preserving the directional information. In the rest of this subsection, we will elaborate on some popular solutions. We will adopt the GCN architecture for all models because it is the most popular choice in related literature.

Inspired by [30], MotifNet [1] constructs a symmetric motif adjacency matrix using digraph motifs that represents meaningful graph connectivity patterns. This approach starts with defining a collection of "graph motifs", $M_1, ..., M_T$, denoting connected digraph patterns. We provide an example of graph motifs in **Figure 2.1**.

For an edge $(v_i, v_j) \in \mathcal{E}$, MotifNet counts the number of times that node v_i and v_j presents in motif M_t. This count is denoted $c_{t,ij}$. Note that v_i and v_j can participate in more than one graph motif. So, for each graph motif M_t, MotifNet will count for all the node pairs. Then, for $t = 1, ..., T$, the motif adjacency matrix is defined as

$$\tilde{\mathbf{A}}_t(i, j) = c_{t,ij}(\mathbf{A}(i, j) + \mathbf{A}(j, i)).$$

Accordingly, the motif Laplacian is computed as

$$\tilde{\mathcal{L}}_t = \mathbf{I} - \tilde{\mathbf{D}}_t^{-1/2}\tilde{\mathbf{A}}_t\tilde{\mathbf{D}}_t^{-1/2}.$$

where the motif degree matrix $\tilde{\mathbf{D}}_t$ is generated from the new motif adjacency matrix. Since

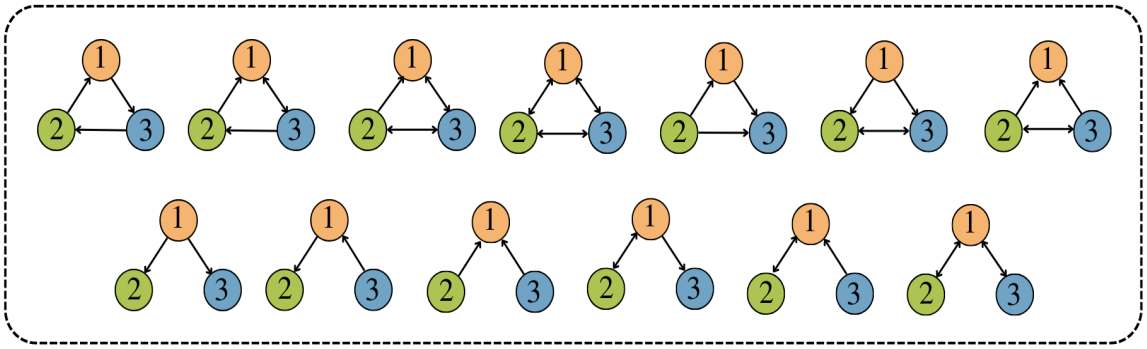

Figure 2.1: Graph motifs used in the experiment of MotifNet[1]

the motif adjacency matrix is symmetric, and the motif degree matrix is diagonal, the motif Laplacian is symmetric as well. With the motif Laplacian $\tilde{\mathcal{L}}_1, ..., \tilde{\mathcal{L}}_T$ and a polynomial degree P, the motif graph convolution is defined as

$$g_\omega * x = \mathcal{P}_\omega(\tilde{\mathcal{L}}_1, ..., \tilde{\mathcal{L}}_T)x,$$

where

$$\begin{aligned}
\mathcal{P}_\omega(\tilde{\mathcal{L}}_1, ..., \tilde{\mathcal{L}}_T) &= \sum_{p=0}^{P} \omega_p \mathcal{P}_p, \\
\mathcal{P}_p(\tilde{\mathcal{L}}_1, ..., \tilde{\mathcal{L}}_T) &= \sum_{t=1}^{T} w_{t,p} \tilde{\mathcal{L}}_t \mathcal{P}_{t-1}, \quad p = 1, ..., P, \\
\mathcal{P}_0 &= \mathbf{I},
\end{aligned}$$

and $\omega = (\omega_0, ..., \omega_P, w_1, 1, ..., w_{T,P})$ with $0 \leq w_{t,p} \leq 1$. Although MotifNet provides a good example of incorporating directional information in a symmetric graph Laplacian, it fails to account for the global topological graph structure [31]. For example, MotifNet's experiment is based on thirteen 3-vertex graph motifs, so, the motif Laplacian only encodes directional information in very small directed subgraphs. If we try to investigate more complex patterns with motifs containing more vertices, then possible motif patterns will increase dramatically, resulting in computational inefficiency.

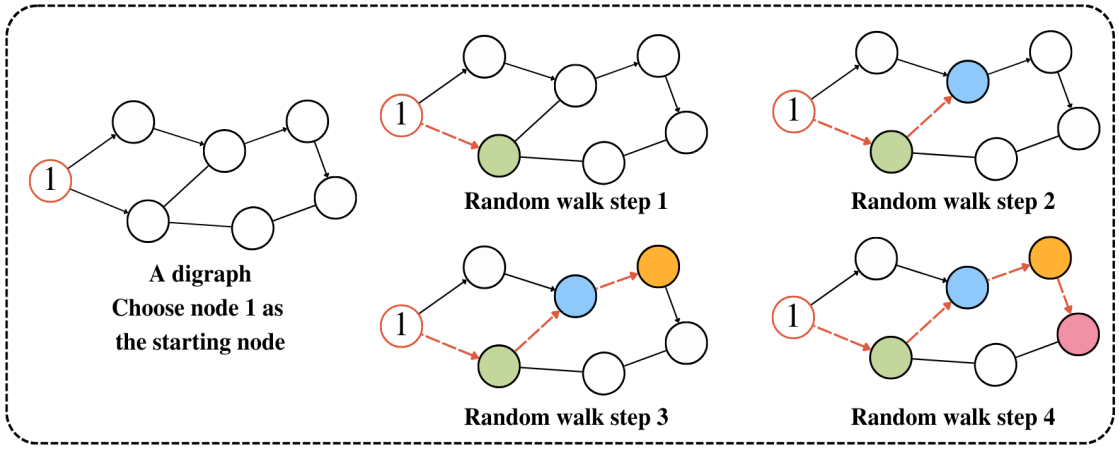

Figure 2.2: Illustration of a 4-step random walk on a graph. This process starts from node 1 and walks to one of the 1-hop out-neighbors of the current node at each step.

Then, inspired by [32], DGCN-Ma [22] uses the transition probability matrix and its Perron vector to construct symmetric Laplacian (since we will introduce two different models that were originally named as DGCN, we distinguish them by the name of the first authors). In graphs, a random walk is a random process to define a path that can describe the transition among nodes (see **Figure 2.2** for visualized illustration). If we begin with the node v_i, then at the next random walk step, we randomly move to one of the closest out-neighbors of v_i, such as v_j (assume that $(v_i, v_j) \in \mathcal{E}$). Next, to lengthen the path, we take another random walk step to move from v_j to its random out-neighbor v_k (assume that $(v_j, v_k) \in \mathcal{E}$). We repeat this process until the path is sufficiently long. Mathematically, the random walk can be realized by a Markov process with a transition probability matrix $\mathbf{P}$:

$$\mathbf{P}(i,j) = \begin{cases} \frac{1}{D_i^{out}}, & \text{if}(v_i, v_j) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases},$$

where D_i^{out} is the out-degree of node v_i. $\mathbf{P}(i,j)$ is the probability of traveling from node v_i to node v_j. With the out-degree matrix and the adjacency matrix, $\mathbf{P}$ can be obtained by

$$\mathbf{P} = \mathbf{D}_{out}^{-1}\mathbf{A},$$

where $\mathbf{D}_{out}$ is a diagonal matrix given by

$$\mathbf{D}_{out}(i,i) = \sum_{v_j \in \mathcal{V}} \mathbf{A}(i,j).$$

According to the Perron-Frobenius Theorem [33], an irreducible non-negative matrix has a unique positive eigenvector, whose associated eigenvalue has the largest absolute value over all other eigenvalues. Based on this theorem, Ma et al. [22] proves that $\mathbf{P}$ of strongly connected digraphs has a unique left eigenvector (a row vector), μ, whose entries are all positive. With this eigenvector, the following equation holds:

$$\mu\mathbf{P} = 1\mu.$$

Then, the normalized vector μ_{norm} is called the Perron vector of $\mathbf{P}$. Let $\mathcal{M} = \text{diag}(\mu_{norm})$, where the diagonal entries are the elements in μ_{norm}. With matrix $\mathcal{M}$, the Perron Laplacian is defined as

$$\tilde{\mathcal{L}}_P = \mathbf{I} - \frac{1}{2}\left(\mathcal{M}^{1/2}\mathbf{P}\mathcal{M}^{-1/2} + \mathcal{M}^{-1/2}\mathbf{P}^\top\mathcal{M}^{1/2}\right).$$

Eventually, DGCN-Ma uses the Perron digraph Laplacian for digraph convolution:

$$g_\omega * x = \omega\left(\mathbf{I} + \frac{1}{2}\left(\mathcal{M}^{1/2}\mathbf{P}\mathcal{M}^{-1/2} + \mathcal{M}^{-1/2}\mathbf{P}^\top\mathcal{M}^{1/2}\right)\right)x.$$

To avoid the exploding or vanishing gradient problem, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is used to calculate $\tilde{\mathbf{P}}$ and $\tilde{\mathcal{M}}$ [7]. Consequently, the Perron-based graph convolution is denoted as

$$g_\omega * x = \frac{\omega}{2}\left(\tilde{\mathcal{M}}^{1/2}\tilde{\mathbf{P}}\tilde{\mathcal{M}}^{-1/2} + \tilde{\mathcal{M}}^{-1/2}\tilde{\mathbf{P}}^\top\tilde{\mathcal{M}}^{1/2}\right)x.$$

As the initial attempt of Perron vector-based methods, DGCN-Ma has opened up a new perspective for researchers to extend spectral GCNNs to digraphs. Nonetheless, the adaptability of this

method in large-size graphs is very weak due to the time-consuming matrix decomposition in graph convolution [34]. In addition, DGCN-Ma is criticized for being only suitable for strongly connected graphs [34]. Perron-Frobenius Theorem holds only for irreducible non-negative matrices, which means the directed graph is required to be strongly connected. Otherwise, the transition probability matrix $\mathbf{P}$ is simply non-negative, leading to weaker properties. According to [35], a non-negative matrix has a non-negative eigenvector, and the corresponding eigenvalue is only no less than the absolute value of other eigenvalues. So, the generalization capability of DGCN-Ma in real-life applications is very limited.

As a refinement of DGCN-Ma, FDGCN [34] is a scalable spectral GCNN with fast localized frequency filters. Since obtaining $\tilde{\mathcal{M}}$ is time-consuming, FDGCN implements a special case of the Perron vector μ_{norm}:

$$\tilde{\mu} = \left(\frac{1}{N}, ..., \frac{1}{N}\right) \in \mathbb{R}^N,$$

where N is the number of nodes. It is proved in [34] that this special case is equal to the Perron vector of strongly connected digraphs, and is also applicable to general digraphs. With this newly defined Perron vector, the Perron-based graph convolution is rewritten as:

$$\begin{aligned} g_\omega * x &= \frac{\omega}{2}\left(\left(\frac{1}{\sqrt{N}}\mathbf{I}\right)\tilde{\mathbf{P}}\left(\sqrt{N}\mathbf{I}\right) + \left(\sqrt{N}\mathbf{I}\right)\tilde{\mathbf{P}}^\intercal\left(\frac{1}{\sqrt{N}}\mathbf{I}\right)\right)x \\ &= \frac{\omega}{2}(\tilde{\mathbf{P}} + \tilde{\mathbf{P}}^\intercal)x \\ &= \frac{\omega}{2}\left(\tilde{\mathbf{D}}_{out}^{-1}(\mathbf{A}+\mathbf{I}) + (\mathbf{A}+\mathbf{I})^\intercal\tilde{\mathbf{D}}_{out}^{-1}\right)x \\ &= \frac{\omega}{2}\left(\tilde{\mathbf{D}}_{out}^{-1/2}(\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^\intercal)\tilde{\mathbf{D}}_{out}^{-1/2}\right)x. \end{aligned}$$

Now, the computationally expensive component $\tilde{\mathcal{M}}$ is removed from the convolution. Thus, FDGCN is much faster than DGCN, thus allowing the processing and analysis of large graphs. Moreover, as $\tilde{\mu}$ is applicable to general digraphs, FDGCN successfully overcomes the limitation in digraph connection and broadens the range of applications of Perron-based methods. However, although FDGCN has demonstrated its power over many state-of-the-art models, the existing experiment of FDGCN only involves the node classification task. So, the performance of FDGCN in other tasks such as link prediction remains unknown.

Another very popular approach is based on proximity matrices. This idea was first proposed by DGCN-Tong [36]. In general, this method generates three symmetric Laplacian matrices including one first-order proximity matrix and two second-order proximity matrices. The first-order proximity indicates the local pairwise proximity between the nodes in a graph. For two nodes v_i, v_j, and the edge (v_i, v_j), the first proximity adjacency matrix is defined as

$$\mathbf{A}_1(i,j) = \mathbf{A}_s(i,j),$$

where $\mathbf{A}_s$ is the symmetrized digraph adjacency matrix. If there is no linkage between v_i and v_j, then $\mathbf{A}_1(i,j) = \mathbf{A}_1(j,i) = 0$. It is not hard to see that $\mathbf{A}_1$ is symmetric. However, one may notice that the first-order proximity matrix contains no directional information. To address this problem, DGCN-Tong uses second-order proximity matrices to retain the missing information. The design of second-order proximity matrices is under the assumption that two nodes with similar neighborhoods share similar characteristics. Hence, second-order proximity matrices aim to connect similar nodes. For a node pair v_i, v_j, the second-order in-degree and out-degree proximity adjacency matrices are given by

$$\mathbf{A}_{2_{in}}(i,j) = \sum_{v_k \in \mathcal{V}} \frac{\mathbf{A}(k,i)\mathbf{A}(k,j)}{\sum_{v_p \in \mathcal{V}} \mathbf{A}(k,p)},$$

$$\mathbf{A}_{2_{out}}(i,j) = \sum_{v_k \in \mathcal{V}} \frac{\mathbf{A}(i,k)\mathbf{A}(j,k)}{\sum_{v_p \in \mathcal{V}} \mathbf{A}(p,k)}.$$

In the in-degree adjacency matrix $\mathbf{A}_{2_{in}}$, a larger value of $\mathbf{A}_{2_{in}}(i,j)$ alludes the higher similarity between v_i and v_j based on the second-order in-degree. Likewise, in the out-degree adjacency matrix $\mathbf{A}_{2_{out}}$, each entry measures the similarity between two nodes based on the second-order out-degree, where larger values also indicate higher similarity. By construction, both matrices are symmetric. The first-order proximity graph convolution is defined as

$$\{g_\omega * x\}_1 = \omega \left(\tilde{\mathbf{D}}_1^{-1/2} \tilde{\mathbf{A}}_1 \tilde{\mathbf{D}}_1^{-1/2} \right) x,$$

where $\tilde{\mathbf{A}}_1 = \mathbf{A}_1 + \mathbf{I}$, and $\tilde{\mathbf{D}}_1$ is the corresponding degree matrix. In addition, the second-order proximity graph convolutions are given by

$$\{g_\omega * x\}_{2_{in}} = \omega\left(\tilde{\mathbf{D}}_{2_{in}}^{-1/2}\tilde{\mathbf{A}}_{2_{in}}\tilde{\mathbf{D}}_{2_{in}}^{-1/2}\right)x,$$

$$\{g_\omega * x\}_{2_{out}} = \omega\left(\tilde{\mathbf{D}}_{2_{out}}^{-1/2}\tilde{\mathbf{A}}_{2_{out}}\tilde{\mathbf{D}}_{2_{out}}^{-1/2}\right)x,$$

where $\tilde{\mathbf{A}}_{2_{in}} = \mathbf{A}_{2_{in}} + \mathbf{I}$ and $\tilde{\mathbf{A}}_{2_{out}} = \mathbf{A}_{2_{out}} + \mathbf{I}$ with $\tilde{\mathbf{D}}_{2_{in}}$ and $\tilde{\mathbf{D}}_{2_{out}}$ being the corresponding degree matrices. Lastly, DGCN-Tong adopts a fusion operation to combine three separate convolutions. It is suggested in [36] that the most effective way is applying the concatenation fusion operator "Concat$(\cdot)$" to stack the convolutions, that is,

$$g_\omega * x = \text{Concat}(\{\omega * x\}_1, \alpha\{\omega * x\}_{2_{in}}, \beta\{\omega * x\}_{2_{out}}),$$

where α and β are weights that control the contribution of each proximity convolution. However, in the experiment of DGCN-Tong, α and β are set manually. So, the importance of different proximity information is not determined by graph data. This raises our concern that selecting the appropriate weights for different graphs may be impractical in real-life applications. This process can be rather costly, and it cannot guarantee finding the optimal weights. Besides, as stated by Tong et al. [20], the second-order proximity measures the similarity between nodes only based on the 1-hop neighborhood, so it fails to capture the global graph structure. As we can see in the definition of $\mathbf{A}_{2_{in}}$ and $\mathbf{A}_{2_{out}}$, the similarity score (i.e., the value of each entry) is measured based on all the shared 1-hop neighbors $v_k \in \mathcal{N}(v_i, 1) \cap \mathcal{N}(v_j, 1)$. Therefore, they fail to account for more distant neighbors that may also be compelling indicators for similarity, especially in dense graphs. Another very popular approach is based on proximity matrices. This idea was first proposed by DGCN-Tong [36]. In general, this method generates three symmetric Laplacian matrices including one first-order proximity matrix and two second-order proximity matrices. The first-order proximity indicates the local pairwise proximity between the nodes in a graph. For two nodes v_i, v_j, and the edge (v_i, v_j), the first proximity adjacency matrix is defined as

$$\mathbf{A}_1(i,j) = \mathbf{A}_s(i,j),$$

where $\mathbf{A}_s$ is the symmetrized digraph adjacency matrix. If there is no linkage between v_i and v_j, then $\mathbf{A}_1(i,j) = \mathbf{A}_1(j,i) = 0$. It is not hard to see that $\mathbf{A}_1$ is symmetric. However, one may notice that the first-order proximity matrix contains no directional information. To address this problem, DGCN-Tong uses second-order proximity matrices to retain the missing information. The design of second-order proximity matrices is under the assumption that two nodes with similar neighborhoods share similar characteristics. Hence, second-order proximity matrices aim to connect similar nodes. For a node pair v_i, v_j, the second-order in-degree and out-degree proximity adjacency matrices are given by

$$\mathbf{A}_{2_{in}}(i,j) = \sum_{v_k \in \mathcal{V}} \frac{\mathbf{A}(k,i)\mathbf{A}(k,j)}{\sum_{v_p \in \mathcal{V}} \mathbf{A}(k,p)},$$

$$\mathbf{A}_{2_{out}}(i,j) = \sum_{v_k \in \mathcal{V}} \frac{\mathbf{A}(i,k)\mathbf{A}(j,k)}{\sum_{v_p \in \mathcal{V}} \mathbf{A}(p,k)}.$$

In the in-degree adjacency matrix $\mathbf{A}_{2_{in}}$, a larger value of $\mathbf{A}_{2_{in}}(i,j)$ alludes the higher similarity between v_i and v_j based on the second-order in-degree. Likewise, in the out-degree adjacency matrix $\mathbf{A}_{2_{out}}$, each entry measures the similarity between two nodes based on the second-order out-degree, where larger values also indicate higher similarity. By construction, both matrices are symmetric. The first-order proximity graph convolution is defined as

$$\{g_\omega * x\}_1 = \omega \left(\tilde{\mathbf{D}}_1^{-1/2} \tilde{\mathbf{A}}_1 \tilde{\mathbf{D}}_1^{-1/2} \right) x,$$

where $\tilde{\mathbf{A}}_1 = \mathbf{A}_1 + \mathbf{I}$, and $\tilde{\mathbf{D}}_1$ is the corresponding degree matrix. In addition, the second-order proximity graph convolutions are given by

$$\{g_\omega * x\}_{2_{in}} = \omega \left(\tilde{\mathbf{D}}_{2_{in}}^{-1/2} \tilde{\mathbf{A}}_{2_{in}} \tilde{\mathbf{D}}_{2_{in}}^{-1/2} \right) x,$$

$$\{g_\omega * x\}_{2_{out}} = \omega \left(\tilde{\mathbf{D}}_{2_{out}}^{-1/2} \tilde{\mathbf{A}}_{2_{out}} \tilde{\mathbf{D}}_{2_{out}}^{-1/2} \right) x,$$

where $\tilde{\mathbf{A}}_{2_{in}} = \mathbf{A}_{2_{in}} + \mathbf{I}$ and $\tilde{\mathbf{A}}_{2_{out}} = \mathbf{A}_{2_{out}} + \mathbf{I}$ with $\tilde{\mathbf{D}}_{2_{in}}$ and $\tilde{\mathbf{D}}_{2_{out}}$ being the corresponding degree matrices. Lastly, DGCN-Tong adopts a fusion operation to combine three separate convolutions. It is suggested in [36] that the most effective way is applying the concatenation fusion operator "Concat(·)" to stack the convolutions, that is,

$$g_\omega * x = \text{Concat}(\{\omega * x\}_1, \alpha\{\omega * x\}_{2_{in}}, \beta\{\omega * x\}_{2_{out}}),$$

where α and β are weights that control the contribution of each proximity convolution. However, in the experiment of DGCN-Tong, α and β are set manually. So, the importance of different proximity information is not determined by graph data. This raises our concern that selecting the appropriate weights for different graphs may be impractical in real-life applications. This process can be rather costly, and it cannot guarantee finding the optimal weights. Besides, as stated by Tong et al. [20], the second-order proximity measures the similarity between nodes only based on the 1-hop neighborhood, so it fails to capture the global graph structure. As we can see in the definition of $\mathbf{A}_{2_{in}}$ and $\mathbf{A}_{2_{out}}$, the similarity score (i.e., the value of each entry) is measured based on all the shared 1-hop neighbors $v_k \in \mathcal{N}(v_i, 1) \cap \mathcal{N}(v_j, 1)$. Therefore, they fail to account for more distant neighbors that may also be compelling indicators for similarity, especially in dense graphs.

[20] is a successful integration of DGCN-Ma [22] and DGCN-Tong [36]. This work begins with defining a digraph Laplacian similar to DGCN-Ma, but with a PaperRank matrix instead of a random walk matrix. This PaperRank matrix is proven to be non-negative and irreducible even if a digraph is not strongly connected. So, the new digraph Laplacian is universally applicable to any digraph. Then, a new model called Digraph is constructed using this Laplacian. However, this method only allows us to extract information from the 1-hop neighborhood. To encode more information, k^{th}-order proximity, which is developed from the proximity approach in DGCN-Tong, is implemented to construct a more flexible network structure. With the k^{th}-order proximity, the model can explore the information contained in the $(k-1)$-hop neighborhood. The model built upon the k^{th}-order proximity architecture is known as DiGCN. Now we will have a brief review of the mathematical principles underlying Digraph and DiGCN.

Since the probability transition probability matrix $\mathbf{P}$ in DGCN-Ma is not irreducible when a digraph is not strongly connected, Digraph introduces a PaperRank matrix $\mathbf{R} = (1-\eta)\mathbf{P} + \frac{\eta}{N}\mathbf{I}$, where $\eta \in (0,1)$ is the teleport probability [37], and N is the number of nodes. $\mathbf{R}$ is an irreducible non-negative matrix, so the Perron-Frobenius Theorem holds. Namely, following [22], the Perron vector μ_R (normalized) can be found and used to construct the PaperRank-based Perron Laplacian as

$$\mathcal{L}_R = \mathbf{I} - \frac{1}{2}\left(\mathcal{M}_R^{1/2}\mathbf{R}\mathcal{M}_R^{-1/2} + \mathcal{M}_R^{-1/2}\mathbf{R}^\intercal\mathcal{M}_R^{1/2}\right),$$

where $\mathcal{M}_R = \mathrm{diag}(\mu_R)$. However, the new Perron Laplacian is very dense due to the incorporation of a teleport back to every node in the graph, which significantly increases the computational overhead. To address this issue, the authors introduce an auxiliary node as the personalized PaperRank teleport, and refine the PaperRank matrix as

$$\mathbf{R}_p = \begin{bmatrix} (1-\eta)\tilde{\mathbf{P}} & \eta \\ \frac{\eta}{N} & 0 \end{bmatrix} \in \mathbb{R}^{(N+1)\times(N+1)},$$

where $\tilde{\mathbf{P}}$ is derived from $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. $\mathbf{R}_p$ is also an irreducible non-negative matrix, and the Perron vector of $\mathbf{R}_p$ is used to approximate the Perron vector of $\tilde{\mathbf{P}}$. Nevertheless, it is worth noting that the Perron vector μ_{R_p} cannot be straightforwardly applied to build the Perron Laplacian, because it has one element corresponding to the auxiliary node. Thus, the Perron vector is split to two parts $\mu_{R_p} = (\mu_{R_p,N}, \mu_{R_p,A})$, then $\mu_{R_p,A} \in \mathbb{R}$ is removed, and $\mu_{R_p,N} \in \mathbb{R}^N$ is used to construct the refined Perron Laplacian:

$$\mathcal{L}_{R_p} = \mathbf{I} - \frac{1}{2}\left(\mathcal{M}_{R_p}^{1/2}\tilde{\mathbf{P}}\mathcal{M}_{R_p}^{-1/2} + \mathcal{M}_{R_p}^{-1/2}\tilde{\mathbf{P}}^\intercal\mathcal{M}_{R_p}^{1/2}\right),$$

where $\mathcal{M}_{R_p} = \mathrm{diag}(\mu_{R_p,N})$. With $\mathcal{L}_{R_p}$, Digraph defines the graph convolution as

$$g_\omega * x = \frac{\omega}{2}\left(\tilde{\mathcal{M}}_{R_p}^{1/2}\tilde{\mathbf{P}}\tilde{\mathcal{M}}_{R_p}^{-1/2} + \tilde{\mathcal{M}}_{R_p}^{-1/2}\tilde{\mathbf{P}}^\intercal\tilde{\mathcal{M}}_{R_p}^{1/2}\right)x.$$

Next, authors of [20] considered how to incorporate proximity in the Perron-based approaches,

and designed DiGCN based on this idea. The aim of applying proximity is to obtain a scalable receptive field such that the model can extract information from neighborhoods of different scales. More specifically, defining the digraph Laplacian with k^{th}-order proximity enables information extraction in the $(k-1)$-hop neighborhood, which means the model can explore node neighbors of different distances via setting different k values.

The k^{th}-order proximity matrix is defined as

$$\mathbf{P}_k = \begin{cases} \mathbf{I}, & k = 1 \\ \tilde{\mathbf{P}}, & k = 2 \\ \text{Intersect}\left(\mathbf{P}_2^{k-2}\left(\mathbf{P}_2^{\intercal}\right)^{k-2}, \left(\mathbf{P}_2^{\intercal}\right)^{k-2}\mathbf{P}_2^{k-2}\right)/2, & k > 2 \end{cases}.$$

Here, Intersect$(\cdot)$ is the element-wise intersection of matrices. If both v_i and v_j can reach node v_p in k steps, a k-step meeting path exists between v_i and v_j. Alternatively, if v_p can reach both v_i and v_j in k steps, then a k-step diffusion path exists between v_i and v_j. In $\mathbf{P}_2^{k-2}\left(\mathbf{P}_2^{\intercal}\right)^{k-2}$ and $\left(\mathbf{P}_2^{\intercal}\right)^{k-2}\mathbf{P}_2^{k-2}$, an element (i, j) has two corresponding nodes v_i and v_j. For $k > 2$, if both k-step meeting and diffusion path exist between v_i and v_j, the intersection operator performs summation of $\mathbf{P}_2^{k-2}\left(\mathbf{P}_2^{\intercal}\right)^{k-2}(i, j)$ and $\left(\mathbf{P}_2^{\intercal}\right)^{k-2}\mathbf{P}_2^{k-2}(i, j)$, otherwise, it is assigned to 0. Through the rule of the intersection operator, it is not hard to see that $\mathbf{P}_k$ for $k > 2$ is symmetric.

Eventually, the graph convolution of DiGCN is given by

$$\{g_\omega * x\}_k = \begin{cases} \omega_1 x, & k = 1 \\ \frac{1}{2}\omega_2\left(\tilde{\mathcal{M}}_2^{1/2}\mathbf{P}_2\tilde{\mathcal{M}}_2^{-1/2} + \tilde{\mathcal{M}}_2^{-1/2}\mathbf{P}_2^{\intercal}\tilde{\mathcal{M}}_2^{1/2}\right)x, & k = 2 \\ \omega_k\left(\mathbf{W}_k^{-1/2}\mathbf{P}_k\mathbf{W}_k^{-1/2}\right)x, & k > 2 \end{cases},$$

where $\mathbf{W}_k$ is a diagonal weight matrix of $\mathbf{P}_k$. The connection between DiGCN and Digraph is that when $k = 2$, $\{g_\omega * x\}_2$ is identical with the graph convolution of Digraph. To combine all the convolution outputs, DiGCN adopts the same way as DCGN-Tong in its experiment, that is, the concatenation operator, Concat$(\cdot)$. The major drawback of Digraph and DiGCN is their inability to be implemented for large digraphs because these two models require full batch training to express their power [20].

Later, a magnetic Laplacian-based neural network, MagNet [21], demonstrates its power in node classification and link prediction tasks across various datasets. Unprecedentedly, MagNet adopts a complex digraph Laplacian and allows graph signal processing in the complex domain. The underlying rationale of MagNet is to incorporate the magnetic Laplacian, a classic concept that has been studied at least from 1993 [38], in traditional spectral GCNN architectures. Detailed mathematical illustrations of the magnetic Laplacian will be presented in Section 3.1 of Chapter 3.

To summarize, we have reviewed popular spectral digraph GCNs, including MotifNet, DGCN-Ma, FDGCN, DGCN-Tong, Digraph, DiGCN, and MagNet. These spectral networks are different from our model, Framelet-MagNet, in the following aspects. Firstly, models except for MagNet use real-valued digraph Laplacian, while our model adopts the complex magnetic Laplacian to explore a new perspective of digraph information. Next, experiments conducted for most models only involve the node classification task, which means the predictive power of these methods in other tasks is not validated. However, Framelet-MagNet shows its superior competence in not only node classification, but also link prediction and denoising. Compared with DGCN-Ma, our model does not require the input digraph to be strongly connected. Distinct from DGCN-Tong and DiGCN, our method does not include a complicated construction of multiple digraph Laplacian. Moreover, all the aforementioned models apply graph Fourier transform in signal frequency filtering, failing to account for more sophisticated frequency decomposition. By contrast, our model implements graph Framelet transform, allowing more meticulous extraction of different frequency components.

2.2 Graph Wavelets, Graph Framelets, and Relevant Graph Convolutional Networks

The word “framelet” is an abbreviation of “wavelet frame”. As a matter of fact, wavelets are special cases of framelets. In the $\mathbb{R}^d$ domain, wavelets and framelets have been studied for a long time and have a collection of well-defined systems [39, 40, 41]. Then, how to build

wavelet/framelet systems on manifolds gradually gained attention from the research community. Various solutions have been proposed. For example, Coifman and Maggioni [42] constructed an orthogonal diffusion wavelet system using diffusion operators for smooth manifolds. As another example, Wang and Zhuang [43] designed a tight framelet system for compact and smooth manifolds. For a more comprehensive review of wavelet/framelet systems on $\mathbb{R}^d$ and manifolds, we refer to [44]. Given the fact that graphs can be considered as discrete samples of manifolds, in 2003, Crovella and Kolaczyk [45] first introduced wavelet theory to graphs. Henceforth, a variety of studies on graph wavelet/framelet theory have taken place [25, 46, 47, 48, 49]. In [47], the wavelet scattering system is designed as a GCNN-like machine learning algorithm for graph analysis. However, this approach substitutes the whole GCNN architecture with the wavelet scattering system. Our method in this paper, on the other hand, aims to conserve the original GCNN architecture. The most related works of our method are UFG [28], QUFG [27], and SVD-GCN [29]. All these works are built upon the "undecimated graph framelet system" [44, 48], which is developed from the spectral graph wavelet system in [25]. In the rest of this section, we will review these works and their underlying principles.

Wavelet transform employs a set of self-similar wavelets, developed from a "mother wavelet" through translation and dilation, then obtains transform coefficients by taking the inner product between signals and wavelets. Let $\eta(t)$ be a mother wavelet. With a translation parameter τ and a dilation level $s > 0$, wavelets are calculated as

$$\eta_{\tau,s}(t) = \frac{1}{s}\eta\left(\frac{t-\tau}{s}\right).$$

Then, wavelet coefficients for a given signal f from a continuous domain $\mathbb{R}$ are given by

$$W_f(\tau, s) = \int_{-\infty}^{\infty} \frac{1}{s}\eta^*\left(\frac{t-\tau}{s}\right) f(t)dt,$$

where η^* is the conjugate transpose of η.

However, defining translation and dilation on graph data in this way is difficult. For example, while a normal signal f is defined on the time domain, graph signals are defined on the graph

domain which is usually a discrete set $n = 1, ..., N$ (i.e. the node number). Dividing t by dilation level s can be regarded as expanding or shrinking the time span. Yet, the meaning of dividing the node number i by s is ambiguous. Therefore, Hammond et al. [25] proposed a brilliant solution by defining scaling in the Fourier domain then expressing translation through a delta impulse. This approach starts with defining a graph wavelet generating kernel ξ that satisfies $\xi(0) = 0$. Let the wavelet operator be a function of the graph Laplacian $\Omega_\xi = \xi(\mathcal{L})$ such that $\Omega_\xi x$ generates the wavelet coefficients for graph signal x at the unit scaling level ($s = 1$). The scale level can be altered by applying a multiplier to $\mathcal{L}$. More specifically, the wavelet operator at a level $s > 0$ is $\Omega_\xi^s = \xi(s\mathcal{L})$. Furthermore, to realize transition, this operator is localized to an impulse. This process is fulfilled by a Dirac delta function, which is then replaced by a position indicator, represented by the eigenvectors of graph Laplacian. Eventually, a graph wavelet centered at node position n (i.e., the node v_n) with dilation level s is computed as

$$\eta_{n,s}(m) = \sum_{k=0}^{N-1} \xi(s\lambda_k) u_k^*(n) u_k(m), \tag{2.1}$$

where $m = 1, ..., N$, and $\{u_k, \lambda_k\}_{k=0}^{N-1}$ are eigenvector and eigenvalue pairs of the graph Laplacian $\mathcal{L}$. The spectral graph wavelet transform (SGWT) of a single graph signal x is given by

$$W_x(n, s) = \langle \eta_{n,s}, x \rangle = \sum_{k=0}^{N-1} \xi(s\lambda_k) u_k(n) \hat{x}(k). \tag{2.2}$$

Developed from SGWT, wavelet frame transform for graphs (WFTG) was proposed by Dong [48]. This idea was also introduced as undecimated framelet transform (UFT) by Zheng et al. [44] in Section 4.1. The undecimated framelet system is constructed through a filter bank. This method is known as Multiresolution Analysis in relevant literature [50, 51, 52]. Rather than using a single mother wavelet, the framelet system adopts a set of finite functions as mother wavelets (also known as scaling functions in the framelet terminology), enriching the basis of transformation.

The construction of graph framelets requires a set of scaling functions $A = \{\alpha_0, \alpha_1, ..., \alpha_R\}$, and a filter bank $a = \{a_0, a_1, ..., a_R\}$. Normally, a_0 is considered as a low-pass filter, and a_r for

$r = 1, ..., R$ are considered as high-pass filters. With $\alpha_0, \alpha_1, ..., \alpha_R$, graph framelets with dilation level $s = 1, ..., S$ centered at node position n are defined as

$$\begin{aligned} \rho_{n,s}(m) &= \sum_{k=0}^{N-1} u_k(m)\hat{\alpha}_0\left(2^{-s}\lambda_k\right) u_k^*(n), \\ \varrho_{n,s,r}(m) &= \sum_{k=0}^{N-1} u_k(m)\hat{\alpha}_r\left(2^{-s}\lambda_k\right) u_k^*(n), \quad r \geq 1. \end{aligned} \tag{2.3}$$

By definition, $\rho_{n,s}$ and $\varrho_{n,s,r}$ are used to investigate low-frequency and high-frequency components of graph signals, respectively [28, 48]. 2^{-s} in both framlet formulas are adaptable to any other similar expressions δ^{-s} where $\delta > 1$. Then, similar to the aforementioned operations, framelet coefficients are generated by inner products $\langle \rho_{n,s}, x \rangle$ and $\langle \varrho_{n,s,r}, x \rangle$.

For each pair of scaling functions and filters, the following equation should hold for any $\delta \in \mathbb{R}$:

$$\hat{\alpha}_r(2\delta) = \hat{\alpha}_0(\delta)\hat{a}_r(\delta). \tag{2.4}$$

So, graph framelets can be simply defined with an appropriate filter bank, which means finding the exact scaling functions is not necessary. Intuitively, it is expected that no loss will occur during the transform and inversion process between the spectral and graph domain. The framelet system that satisfies this expectation is called a "tight" frame. To meet this objective, the filter bank should satisfy:

$$\sum_{r=0}^{R} \hat{a}_r^2(\delta) \equiv 1, \tag{2.5}$$

for any $\delta \in [0, \pi]$.

Recall that the eigendecomposition of a graph Laplacian is $\mathcal{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$. According to [48], the matrix-vector form of framelet transform can be expressed with a set of framelet operators, $\mathcal{F}_{r,s}$, as following:

$$\mathcal{F}_{r,s}x = \begin{cases} \mathbf{U}\hat{a}_r(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^\intercal x, & s = 1 \\ \mathbf{U}\hat{a}_r(2^{-\gamma-s+1}\mathbf{\Lambda})\hat{a}_0(2^{-\gamma-s+2}\mathbf{\Lambda})\cdots\hat{a}_0(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^\intercal x, & 2 \le s \le S, \end{cases} \tag{2.6}$$

where $(r, s) \in \{(1,1), ..., (1,S), ..., (r,1), ..., (r,S), ..., (R,1), ..., (R,S)\} \cup \{(0,S)\}$. γ is the smallest real number such that $\lambda_{\max} \le 2^\gamma \pi$. $\hat{a}_r(\cdot)$ denotes a diagonal matrix whose diagonal elements are corresponding to each eigenvalue. For example,

$$\hat{a}_r(2^{-\gamma}\mathbf{\Lambda}) = \mathrm{diag}(\hat{a}_r(2^{-\gamma}\lambda_{\mathbf{0}}), ..., \hat{a}_r(2^{-\gamma}\lambda_{\mathbf{N-1}})).$$

Equation (2.6) shows that framelet transform can be achieved by a recursive process with the Fourier transformed filters $\hat{a}_0$ and $\hat{a}_r$. Finally, low-pass and high-pass graph framelet coefficients are denoted as

$$\begin{aligned} F_x(0,S) &= \langle \rho_{n,S}, x\rangle = \mathcal{F}_{0,S}x, \\ F_x(r,s) &= \langle \varrho_{n,s,r}, x\rangle = \mathcal{F}_{r,s}x, \quad 1 \le r \le R, 1 \le s \le S. \end{aligned} \tag{2.7}$$

Accordingly, the reconstruction of these coefficients back to the graph domain is computed as

$$x = \mathcal{F}_{0,S}^\intercal F_x(0,S) + \sum_{s=1}^{S}\sum_{r=1}^{R}\mathcal{F}_{r,s}^\intercal F_x(r,s). \tag{2.8}$$

This framelet system is directly extended to spectral GCNs by [28]. Let x be a single graph signal. With the framelet transform matrix, $\mathcal{F} = \left[\mathcal{F}_{0,S}; \mathcal{F}_{1,1}; ...; \mathcal{F}_{1,S}; ...; \mathcal{F}_{R,S}\right]$, and a learnable filter $g_\omega = \mathrm{diag}(\omega)$, the UFG graph convolution is defined as

$$g_\omega * x = \mathcal{F}^\intercal\left(g_\omega(\mathcal{F}x)\right). \tag{2.9}$$

Although UFG encodes both low and high frequency components of graph signals via framelet transform, we are uncertain about whether to retain or remove a component of a certain frequency [27]. As a result, we leave this task completely to the learnable filter g_ω. Hence, if one wishes to specifically cut off certain spectral information, UFG will not achieve this goal. The solution

proposed by Yang et al. [27] constructs a quasi-filter bank $b = \{b_0, b_1, ..., b_R\}$ straightforwardly in the spectral domain, which satisfies the following condition for any $\delta \in [0, \pi]$:

$$\sum_{r=0}^{R} b_r(\delta)^2 \equiv 1, \tag{2.10}$$

such that the value of b_0 decreases from 1 to 0, and the value of b_R gradually increases from 0 to 1 across the spectral domain. In graph spectral theory, larger Laplacian eigenvalues imply higher frequency. So, b_0 and b_R will attenuate high and low frequency components, respectively. In addition, other filters $b_1, ..., b_{R-1}$ will regulate the frequency components in between. Graph framelets based on $b = \{b_0, b_1, ..., b_R\}$ are know as quasi-framelets. Quasi-framelet transform can be expressed with the quasi-framelet operator, $\dot{\mathcal{F}}_{r,s}$, as

$$\dot{\mathcal{F}}_{r,s}x = \begin{cases} \mathbf{U}b_r(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^\intercal x, & s = 1 \\ \mathbf{U}b_r(2^{-\gamma-s+1}\mathbf{\Lambda})b_0(2^{-\gamma-s+2}\mathbf{\Lambda})\cdots b_0(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^\intercal x, & 2 \le s \le S, \end{cases} \tag{2.11}$$

where $b_r(\cdot)$ is also a diagonal matrix analogous to $\hat{a}_r(\cdot)$. Ultimately, QUFG [27] implements the quasi-framelet convolution with the quasi-framelet transform matrix $\dot{\mathcal{F}} = \left[\dot{\mathcal{F}}_{0,S}; \dot{\mathcal{F}}_{1,1}; ...; \dot{\mathcal{F}}_{1,S}; ...; \dot{\mathcal{F}}_{R,S}\right]$ as

$$g_\omega * x = \dot{\mathcal{F}}^\intercal \left(g_\omega \left(\dot{\mathcal{F}}x\right)\right). \tag{2.12}$$

Both UFG and QUFG show their superiority over Fourier-based spectral GCNs through empirical results. However, up to this point, our discussion is only for undirected graphs. Since both methods require the eigendecomposition of the graph Laplacian, they cannot be applied to digraphs. As we have discussed in Chapter 1, digraph Laplacian does not support the eigendecomposition required by spectral-based networks, which hinders the extension of spectral GCNs to digraphs. Similarly, framelet and quasi-framelet transform are not adaptable to digraphs either. Motivated by this limitation, Zou et al. [29] proposed SVD-GCN that replaces eigendecomposition with singular value decomposition (SVD), thus enabling the incorporation of framelets in digraph neural networks. Although SVD-GCN provides an unprecedented approach to assembling

framelet-based digraph GCNNs, the theoretical rationale of applying SVD is very vague, for example, how the Laplacian singular values can be linked to the signal frequency in the SVD domain.

To summarize, we have reviewed the development of wavelet/framelet theory, then we focus on some most relevant works and their mathematical foundations. Especially, we have discussed three framelet-based digraph GCNs, including UFG, QUFG, and SVD-GCN. Compared with these models, our model, Framelet-MagNet, is distinctive due to the following reasons. Firstly, UFG and QUFG are only limited to undirected graphs, while Framelet-MagNet can be applied to digraphs. In addition, different from SVD-GCN, our model incorporates framelets in digraph GCNs without discarding the role of eigendecomposition of the digraph Laplacian. Furthermore, our approach uniquely employs signal processing in the complex domain. By contrast, although relevant works have proven the validity of using complex eigenvectors in constructing graph framelets [44, 48], none of them implement signal processing in the complex domain in their experiments.

Chapter 3

Framelet-MagNet

In this Chapter, we will introduce our model, Framelet-MagNet, and explain how to apply it to digraph tasks. We will firstly introduce the magnetic Laplacian in Section 3.1. Then, in Section 3.2, we will develop the Magnetic Graph Framelet System (MGFS) based on the magnetic Laplacian. We will show how Magnetic Graph Framelet Transform (MGFT) works in Section 3.3. Moreover, in Section 3.4, we will propose Fast Magnetic Framelet Transform (FMFT), which adopts Chebyshev polynomials for fast computation. FMFT will be exploited in the construction of Framelet-MagNet. The network architecture of Framelet-MagNet will be presented in Section 3.5. Ultimately, in Section 3.6 and Section 3.7, we will show the application of Framelet-MagNet in node classification and link prediction tasks.

3.1 Magnetic Laplacian

Normally, spectral GCNNs define graph convolution in the spectral domain with the support of graph Laplacian eigendecomposition. So, the asymmetric nature of the digraph Laplacian impedes the extension of classic spectral GCNNs to digraphs. To solve this problem, most digraph GCNNs manage to build a symmetric and positive semi-definite digraph Laplacian. Such Laplacian is expected to encode directional information in a symmetric structure. Magnetic Laplacian [21, 23, 53], a symmetric Hermitian matrix, is one of the most successful instances.

Magnetic Laplacian is a representation of digraph whose real part indicates edge existence and imaginary part shows edge directions. From a perspective of physics, it can be interpreted as a discrete quantum mechanical Hamiltonian of a charged particle under the influence of a magnetic flux [54, 55].

By construction, magnetic Laplacian is a complex Hermitian matrix that is equal to its conjugate transpose. Suppose we have an $N \times N$ Hermitian matrix $\mathbf{H}$, then $\mathbf{H}(i,j) = \mathbf{H}^*(j,i)$ for $i, j = 1, ..., N$. As a basic property, any $N \times N$ Hermitian matrix can be eigen-decomposed into N real eigenvalues and N orthonormal eigenvectors. Therefore, magnetic Laplacian can be smoothly adapted into spectral GCNNs.

Now, we will elaborate on how to construct a magnetic Laplacian. Similar to the classic Laplacian, the construction of magnetic Laplacian involves an adjacency matrix based on the graph topological structure and its corresponding degree matrix. For a digraph $\mathcal{G}_d\{\mathcal{V}, \mathcal{E}\}$, we decompose the original adjacency matrix $\mathbf{A}$ into a symmetric matrix $\mathbf{A}_{sym}$ and a skew-symmetric matrix $\mathbf{A}_{skew}$, where

$$\mathbf{A}_{sym}(i,j) = \frac{1}{2}(\mathbf{A}(i,j) + \mathbf{A}(j,i)), \quad 1 \le i, j \le N, \tag{3.1}$$

$$\mathbf{A}_{skew}(i,j) = \mathbf{A}(i,j) - \mathbf{A}(j,i), \quad 1 \le i, j \le N. \tag{3.2}$$

$\mathbf{A}_{sym}$ is the undirected counterpart of the original digraph, and $\mathbf{A}_{skew}$ preserves the directional information lost by the former. In the symmetric matrix, if $\mathbf{A}_{sym} = 0$, then no edge exists between node v_i and v_j, otherwise, v_i and v_j are linked. Then, the edge directions are indicated by the skew-symmetric matrix. Suppose we have a digraph with unweighted edges, then $\mathbf{A}_{skew}(i,j) = 1$ represents an edge from node v_i to node v_j, and -1 gives the other direction. If $\mathbf{A}_{skew}(i,j) = 0$, then it may indicate either an undirected edge or no edge, which means we cannot distinguish these two circumstances with $\mathbf{A}_{skew}$. However, by combining the information in $\mathbf{A}_{sym}$, whether an edge is undirected or absent can be easily identified. To be more specific, when there is an undirected edge between v_i and v_j, we will have $\mathbf{A}_{skew}(i,j) = 0$ and $\mathbf{A}_{sym} = 1$. If v_i and v_j are not related, we will have $\mathbf{A}_{skew}(i,j) = 0$ and $\mathbf{A}_{sym} = 0$. It is worth noting that the argument here does not hold for weighted mixed graphs. We will discuss this issue later in Chapter 5.

Next, for each element in $\mathbf{A}_{skew}$, we compute

$$\mathbf{\Psi}^{(q)}(i,j) = \exp\left(2\pi \mathrm{i} q \mathbf{A}_{skew}(i,j)\right), \quad 1 \le i,j \le N, \tag{3.3}$$

where i is the imaginary unit, and q is a non-negative electric charge parameter. It has been demonstrated that we can highlight different directional graph patterns (i.e., reciprocity and directed circles) by altering the value of q [23, 53], so the selection of q is crucial.

Generally, we have two methods to select q. In the first method, $q = \frac{1}{m}$ if an m-cycle exists in the graph [23]. The specific value of m is determined by Johnson's algorithm [56, 57]. Another method, in [21], treats q as a hyperparameter to be tuned in the learning process. We prefer the second approach because it is more computationally efficient, and the optimal value of q will be an informative measurement of the significance of directional information in the model. As $q = 0$ means no directional information is incorporated, and $q = \frac{1}{4}$ maximizes the utility of the imaginary part, we assume $0 \le q \le \frac{1}{4}$ such that larger q allows the model to encode more directional information [21, 23, 53].

Now, we can define the magnetic adjacency matrix and its associated degree matrix as

$$\begin{aligned} \mathbf{A}^{(q)} &= \mathbf{\Psi}^{(q)} \odot \mathbf{A}_{sym}, \\ \mathbf{D}_{sym}(i,i) &= \sum_{v_j \in \mathcal{V}} \mathbf{A}_{sym}(i,j), \quad 1 \le i \le N, \end{aligned} \tag{3.4}$$

where $\odot$ is the sign of component-wise matrix multiplication. Finally, the normalized magnetic Laplacian can be defined as

$$\mathcal{L}^{(q)} = \mathbf{I} - \mathbf{\Psi}^{(q)} \odot \left(\mathbf{D}_{sym}^{-\frac{1}{2}} \mathbf{A}_{sym} \mathbf{D}_{sym}^{-\frac{1}{2}} \right). \tag{3.5}$$

By construction, $\mathbf{\Psi}^{(q)}$ is Hermitian, and $\mathbf{A}_{sym}, \mathbf{D}_{sym}$ are symmetric. Therefore, $\mathcal{L}^{(q)}$ is a Hermitian matrix. According to the proof in [21], Appendix E, $\mathcal{L}^{(q)}$ enjoys the desirable property of being positive semi-definite, which means it can be eigen-decomposed as $\mathcal{L}^{(q)} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{*}$. Here we use the conjugate transpose because the eigenvectors of $\mathcal{L}^{(q)}$ are complex-valued.

Let $\{u_k, \lambda_k\}_{k=0}^{N-1}$ be the eigenvector and eigenvalue pairs of $\mathcal{L}^{(q)}$. According to [21], we have $\lambda_k \in [0, 2]$ for $k = 0, ..., N-1$.

3.2 Magnetic Graph Framelet System

The most important component in framelet-based spectral GCNNs is the framelet convolution. We firstly transform graph signals to the framelet frequency domain for filtering, then convert the processed data back to the spatial domain with the framelet reconstruction function. Intuitively, we desire no information loss during the whole process, which means we expect the framelet transform to be "tight". Accordingly, we design the Magnetic Graph Framelet System (MGFS), which is a tight framelet system defined on digraphs, as the basis of transform. The fundamental principle of MGFS is to incorporate magnetic Laplacian in the traditional undecimated tight framelet system on undirected graphs. For a review of the traditional approaches and their theoretical background, we refer to Chapter 2, Section 2.2.

The construction of MGFS is based on a set of scaling functions $Z = \{\zeta_0, ..., \zeta_R\}$. With the transition position n and the dilation level $s = 1, ..., S$, we define the low-pass and high-pass magnetic graph framelets $\rho_{n,s}^{(q)}$ and $\varrho_{n,s,r}^{(q)}$ as

$$\begin{aligned} \rho_{n,s}^{(q)}(m) &= \sum_{k=0}^{N-1} u_k(m) \hat{\zeta}_0 \left(2^{-s}\lambda_k\right) u_k^*(n), \\ \varrho_{n,s,r}^{(q)}(m) &= \sum_{k=0}^{N-1} u_k(m) \hat{\zeta}_r \left(2^{-s}\lambda_k\right) u_k^*(n), \ 1 \le r \le R. \end{aligned} \tag{3.6}$$

The low-pass framelet can detect low frequency components in graph signals, while the high-pass framelets are measurements of high frequency signal components. Then we define the transform basis MGFS as

$$MGFS(Z, z; \mathcal{G}_d) \coloneqq \{\rho_{n,S}^{(q)} : v_n \in \mathcal{V}\} \cup \{\varrho_{n,s,r}^{(q)} : v_n \in \mathcal{V}, s = S_1, ..., S\}_{r=1}^{R}. \tag{3.7}$$

Now, the problem is how to find the scaling functions in $Z = \{\zeta_0, ..., \zeta_R\}$. According to [48],

we can find the appropriate set of scaling functions via Multiresolution Analysis. Basically, we derive scaling functions from a filter bank $a = \{a_0, ..., a_R\}$ defined in the spatial domain with the following relationship $\hat{\zeta}_r(2\delta) = \hat{a}_r(\delta)\hat{\zeta}_0(\delta)$, for $r = 1, ..., R$ and any $\delta \in \mathbb{R}$. For tight transform, the filter bank a should satisfy $\sum_{r=0}^{R} \hat{a}_r(\delta)^2 \equiv 1$ for any $\delta \in [0, \pi]$. Then, "quasi-framelet" proposed by Yang et al. [27] relaxes the requirement of Multiresolution Analysis by straightforwardly constructing a quasi-filter bank $b = \{b_0, ..., b_R\}$ in the Fourier domain. By definition, b should satisfy the identity condition $\sum_{r=0}^{R} b_r(\delta)^2 \equiv 1$ for any $\delta \in [0, \pi]$, such that the value of b_0 decreases from 1 to 0 while the value of b_R increases from 0 to 1 over the Fourier domain $[0, \pi]$. This will allow framelet convolution to impose "double regulation" on the graph signals. More specifically, graph signals are regulated by not only the learnable filter in traditional convolutions, but also the modulation functions $b_0, ..., b_R$, where b_0 and b_R attenuate high and low frequency components, and the rest regulates frequency in between. In the following discussions, we denote $\hat{a} = \{\hat{a}_0, ..., \hat{a}_R\}$ and $b = \{b_0, ..., b_R\}$ collectively as $z = \{z_0, ..., z_R\}$ for simplicity.

The tightness of the magnetic framelet system is closely relevant with the choice of the filter bank $z = \{z_0, ..., z_R\}$. Existing examples of appropriate filter banks include `Haar` [48], `Linear` [48], `Quadratic` [48], `Sigmoid` [27], and `Entropy` [27]:

i) `Haar`

$$z_0(\delta) = \cos\left(\frac{\delta}{2}\right), \quad z_1(\delta) = \sin\left(\frac{\delta}{2}\right);$$

ii) `Linear`

$$z_0(\delta) = \cos^2\left(\frac{\delta}{2}\right), \quad z_1(\delta) = \frac{1}{\sqrt{2}}\sin(\delta), \quad z_2(\delta) = \sin^2\left(\frac{\delta}{2}\right);$$

iii) `Quadratic`

$$z_0(\delta) = \cos^3\left(\frac{\delta}{2}\right), \quad z_1(\delta) = \sqrt{3}\sin\left(\frac{\delta}{2}\right)\cos^2\left(\frac{\delta}{2}\right),$$

$$z_2(\delta) = \sqrt{3}\sin^2\left(\frac{\delta}{2}\right)\cos\left(\frac{\delta}{2}\right), \quad z_3(\delta) = \sin^3\left(\frac{\delta}{2}\right).$$

iv) `Sigmoid`

$$z_0(\delta) = \sqrt{1 - (1 + \exp(-\alpha(\delta/\pi - 0.5)))^{-1}}$$
$$z_1(\delta) = \sqrt{(1 + \exp(-\alpha(\delta/\pi - 0.5)))^{-1}},$$

where $\alpha > 0$. For `Sigmoid`, the default value of α is 20, which promises the power of modulation on both high and low frequency components [27].

v) `Entropy`

$$z_0(\delta) = \begin{cases} \sqrt{1 - a_1^2(\delta)}, & \delta \leq \pi/2 \\ 0, & \text{otherwise} \end{cases}$$
$$z_1(\delta) = \sqrt{4\alpha\delta/\pi - 4\alpha(\delta/\pi)^2}$$
$$z_2(\delta) = \begin{cases} \sqrt{1 - a_1^2(\delta)}, & \delta > \pi/2 \\ 0, & \text{otherwise} \end{cases},$$

where $0 < \alpha \leq 1$. For `Entropy`, the default value of α is 0.5 [27].

3.3 Magnetic Graph Framelet Transform

In this section, we will discuss how to conduct graph signal decomposition with the MGFS defined in Section 3.2. This process is known as Magnetic Graph Framelet Transform (MGFT). We will also introduce how to reconstruct the original graph signals from the spectral domain with the magnetic framelet inversion function.

With the MGFS defined in Section 3.2, we define MGFT as the inner product between graph signals and magnetic framelets $\rho_{n,S}^{(q)}$ and $\varrho_{n,s,r}^{(q)}$. With a single graph signal x, MGFT can be

expressed as

$$\begin{aligned} \left\{\langle x, \rho_{n,S}^{(q)}\rangle\right\}_{v_n\in\mathcal{V}} &= \mathcal{F}_{0,S}^{(q)}x, \\ \left\{\langle x, \varrho_{n,s,r}^{(q)}\rangle\right\}_{v_n\in\mathcal{V}} &= \mathcal{F}_{r,s}^{(q)}x, \quad 1 \le r \le R, 1 \le s \le S, \end{aligned} \tag{3.8}$$

where $\mathcal{F}_{r,s}^{(q)}$ is the magnetic framelet transform operator, given by

$$\mathcal{F}_{r,s}^{(q)}x = \begin{cases} \mathbf{U}z_r(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^*x, & s = 1 \\ \mathbf{U}z_r(2^{-\gamma-s+1}\mathbf{\Lambda})z_0(2^{-\gamma-s+2}\mathbf{\Lambda})\cdots z_0(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^*x, & 2 \le s \le S, \end{cases} \tag{3.9}$$

where $(r, s) \in \{(1,1), ..., (1,S), ..., (r,1), ..., (r,S), ..., (R,1), ..., (R,S)\} \cup \{(0,S)\}$. γ is the minimum value such that $\lambda_{max} \le 2^{\gamma}\pi$. For any $\delta \ge 0$ and $r = 0, .., R$, the expression $z_r(2^{-\gamma-\delta}\mathbf{\Lambda}))$ can be expanded as

$$z_r(2^{-\gamma-\delta}\mathbf{\Lambda})) = \text{diag}\left[z_r(2^{-\gamma-\delta}\lambda_0), ..., z_r(2^{-\gamma-\delta}\lambda_{N-1})\right].$$

Now we rewrite the computation of transform coefficients as

$$\begin{aligned} F_x^{(q)}(0,S) &= \left\{\langle x, \rho_{n,S}^{(q)}\rangle\right\}_{v_n\in\mathcal{V}} = \mathcal{F}_{0,S}^{(q)}x, \\ F_x^{(q)}(r,s) &= \left\{\langle x, \varrho_{n,s,r}^{(q)}\rangle\right\}_{v_n\in\mathcal{V}} = \mathcal{F}_{r,s}^{(q)}x, \quad 1 \le r \le R, 1 \le s \le S, \end{aligned} \tag{3.10}$$

Finally, the reconstruction is realized by the magnetic framelet inversion function as following:

$$\tilde{x} = \mathcal{F}_{0,S}^{(q)*}F_x^{(q)}(0,S) + \sum_{r=1}^{R}\sum_{s=1}^{S}\mathcal{F}_{r,s}^{(q)*}F_x^{(q)}(r,s). \tag{3.11}$$

To have a more concise expression, we may use a vertically stacked transform matrix

$$\mathcal{F}^{(q)} = \left[\mathcal{F}_{0,S}^{(q)}; \mathcal{F}_{1,1}^{(q)}; ...; \mathcal{F}_{1,S}^{(q)}; ...; \mathcal{F}_{R,S}^{(q)}\right], \tag{3.12}$$

to write the transform and inversion functions as

$$F_x^{(q)} = \mathcal{F}^{(q)} x, \tag{3.13}$$

$$\tilde{x} = \mathcal{F}^{(q)^*} F_x^{(q)}, \tag{3.14}$$

where the inversion matrix is the conjugate transpose of the transform matrix $\mathcal{F}^{(q)}$. Recall that we design MGFT as a tight transform, which means $x = \tilde{x}$. We can prove this as follows.

Theorem 3.3.1 (**Tightness of Magnetic Graph Framelet Transform**). *Let $\mathcal{F}^{(q)}$ be the magnetic framelet transform matrix defined in equation (3.12). Let the corresponding inversion matrix be $\mathcal{F}^{(q)^*}$. Then for a single digraph signal $x \in \mathbb{R}^N$, we have*

$$x = \mathcal{F}^{(q)^*} \mathcal{F}^{(q)} x. \tag{3.15}$$

Proof. We assume $S > 3$ in this proof for a clear illustration. But the proof for any $S \geq 1$ is analogous. According to equation (3.12), we have

$$\begin{aligned}
\mathcal{F}^{(q)^*} \mathcal{F}^{(q)} &= \mathcal{F}_{0,S}^{(q)*} \mathcal{F}_{0,S}^{(q)} + \sum_{r=1}^{R} \sum_{s=1}^{S} \mathcal{F}_{r,s}^{(q)*} \mathcal{F}_{r,s}^{(q)} \\
&= \mathcal{F}_{0,S}^{(q)*} \mathcal{F}_{0,S}^{(q)} + \sum_{r=1}^{R} \mathcal{F}_{r,S}^{(q)*} \mathcal{F}_{r,S}^{(q)} + \sum_{r=1}^{R} \sum_{s=1}^{S-1} \mathcal{F}_{r,s}^{(q)*} \mathcal{F}_{r,s}^{(q)}.
\end{aligned}$$

Let

$$Y = \mathcal{F}_{0,S}^{(q)*} \mathcal{F}_{0,S}^{(q)} + \sum_{r=1}^{R} \mathcal{F}_{r,S}^{(q)*} \mathcal{F}_{r,S}^{(q)}.$$

Then, with the definition in equation (3.9), we can write

$$\begin{aligned}
Y &= \mathbf{U} z_0^2(2^{-\gamma-s+1}\mathbf{\Lambda}) z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda}) \mathbf{U}^* \\
&\quad + \sum_{r=1}^{R} \mathbf{U} z_r^2(2^{-\gamma-s+1}\mathbf{\Lambda}) z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda}) \mathbf{U}^* \\
&= \mathbf{U} \left(\sum_{r=0}^{R} z_r^2(2^{-\gamma-s+1}\mathbf{\Lambda}) \right) z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda}) \mathbf{U}^*.
\end{aligned}$$

Recall that no matter $z = \{z_0, ..., z_R\} = \{\hat{a}_0, ..., \hat{a}_R\}$ or $z = \{z_0, ..., z_R\} = \{b_0, ..., b_R\}$, we all have

$$\sum_{r=0}^{R} z_r(\delta) = 1, \quad \forall \delta \in [0, \pi]. \tag{3.16}$$

Since γ is the smallest number such that $\lambda_{max} \leq 2^{\gamma}\pi$, we have $2^{-\gamma}\lambda_{max} \leq \pi$. So, for any eigenvalue $\lambda_k \leq \lambda_{max}$ and any $s \in [1, S]$, we have $2^{-\gamma-s+i}\lambda_k \leq \pi$, where $i \leq s$. In addition, since $\lambda_k \in [0, 2]$ for $k = 0, ..., N-1$, we have $2^{-\gamma-s+i}\lambda_k \geq 0$. Therefore,

$$\begin{aligned} Y &= \mathbf{U}\mathbf{I}z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^*, \\ &= \mathbf{U}z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^*. \end{aligned}$$

Now, with the condition in equation (3.16), we can conduct an iterative calculation as follows. The last step is based on the fact that eigenvectors of $\mathcal{L}^{(q)}$ are orthonormal.

$$\begin{aligned} \mathcal{F}^{(q)^*}\mathcal{F}^{(q)} &= Y + \sum_{r=1}^{R}\sum_{s=1}^{S-1} \mathcal{F}_{r,s}^{(q)*}\mathcal{F}_{r,s}^{(q)} \\ &= \mathbf{U}z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^* + \sum_{r=1}^{R}\sum_{s=1}^{S-1} \mathcal{F}_{r,s}^{(q)*}\mathcal{F}_{r,s}^{(q)} \\ &= \mathbf{U}z_0^2(2^{-\gamma-s+2}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^* \\ &\quad + \sum_{r=1}^{R} \mathbf{U}z_r^2(2^{-\gamma-s+2}\mathbf{\Lambda})z_0^2(2^{-\gamma-s+3}\mathbf{\Lambda}) \cdots z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^* + \sum_{r=1}^{R}\sum_{s=1}^{S-2} \mathcal{F}_{r,s}^{(q)*}\mathcal{F}_{r,s}^{(q)} \\ &\quad \vdots \\ &= \mathbf{U}z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^* + \sum_{r=1}^{R} \mathcal{F}_{r,1}^{(q)*}\mathcal{F}_{r,1}^{(q)} \\ &= \mathbf{U}z_0^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^* + \sum_{r=1}^{R} \mathbf{U}z_r^2(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^* \\ &= \mathbf{U}\left(\sum_{r=0}^{R} z_r^2(2^{-\gamma}\mathbf{\Lambda})\right)\mathbf{U}^* \\ &= \mathbf{U}\mathbf{U}^* \\ &= \mathbf{I} \end{aligned}$$

Therefore, $\mathcal{F}^{(q)^*}\mathcal{F}^{(q)}x = \mathbf{I}x = x$.

This completes the proof. □

3.4 Fast Magnetic Framelet Transform

The fundamental idea of magnetic framelet convolution is to replace the Fourier operator (i.e., the eigenvector matrix $\mathbf{X}$) with the MGFT operator $\mathcal{F}^{(q)}$. As we can see in equation (3.9), MGFT relies on the eigendecomposition of magnetic graph Laplacian. Nonetheless, this is very computationally expensive, especially for large graphs.

Chebyshev polynomial approximation is widely used in traditional graph neural networks for fast computation [7, 24, 25, 28, 48]. Hammond et al. [25] applied Chebyshev polynomials to approximate graph wavelets. Inspired by this, Dong [48] exploited Chebyshev polynomials in approximating graph framelets. Nonetheless, while Hammond et al. [25] attempted to approximate the wavelet functions directly, Dong [48] only focused on the approximation of the filter bank. The latter is comparably faster than the former because it only requires low-degree Chebyshev polynomials. By contrast, the first methods need high-degree Chebyshev polynomials for better accuracy.

For fast computation, we propose the Fast Magnetic Framelet Transform (FMFT). We use Chebyshev approximation to approximate the filter bank $z = \{z_0, ..., z_R\}$ following [48]. As suggested by Yang et al. [27], normally $k = 3$ is a sufficient choice for attaining a highly precise approximation. Let $\mathcal{T}_r(\cdot), r = 0, ..., R$ denotes the Chebyshev polynomial approximation of filters $z_0, ..., z_R$. Then, we can write FMFT as

$$\mathcal{F}_{r,s}^{(q)}x \approx \tilde{\mathcal{F}}_{r,s}^{(q)}x := \begin{cases} \mathbf{U}\mathcal{T}_r(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^*x, & s = 1 \\ \mathbf{U}\mathcal{T}_r(2^{-\gamma-s+1}\mathbf{\Lambda})\mathcal{T}_0(2^{-\gamma-s+2}\mathbf{\Lambda})\cdots\mathcal{T}_0(2^{-\gamma}\mathbf{\Lambda})\mathbf{U}^*x, & 2 \le s \le S, \end{cases} \tag{3.17}$$

with $(r, s) \in \{(1, 1), ..., (1, S), ..., (R, 1), ..., (R, S)\} \cup \{(0, S)\}$. Recall that $\mathcal{L}^{(q)} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$. Therefore, equation (3.17) is equivalent with

$$\tilde{\mathcal{F}}_{r,s}^{(q)} x := \begin{cases} \mathcal{T}_r(2^{-\gamma}\mathcal{L}^{(q)})x, & s = 1 \\ \mathcal{T}_r(2^{-\gamma-s+1}\mathcal{L}^{(q)})\mathcal{T}_0(2^{-\gamma-s+2}\mathcal{L}^{(q)})\cdots\mathcal{T}_0(2^{-\gamma}\mathcal{L}^{(q)})x, & 2 \le s \le S. \end{cases} \tag{3.18}$$

We define the transform operator for FMFT as

$$\tilde{\mathcal{F}}^{(q)} = \left[\tilde{\mathcal{F}}_{0,S}^{(q)}; \tilde{\mathcal{F}}_{1,1}^{(q)}; \ldots; \tilde{\mathcal{F}}_{1,S}^{(q)}; \ldots; \tilde{\mathcal{F}}_{R,S}^{(q)}\right].$$

Finally, the fast computations of magnetic framelet signal decomposition and reconstruction are denoted as

$$\begin{aligned} F_x^{(q)} &\approx \tilde{\mathcal{F}}^{(q)} x \\ x &\approx \tilde{\mathcal{F}}^{(q)^*} F_x^{(q)}. \end{aligned} \tag{3.19}$$

3.5 Framelet-MagNet Network Architecture

Framelet-MagNet is composed of one or multiple magnetic framelet-based convolutional layer(s), an unwind operator and a fully connected linear layer as the output layer. In the output layer, we will apply the Softmax activation function for classification tasks. We will present more details relevant to different tasks in the following Sections. Here we will focus on the magnetic framelet-based convolutional layer and the unwind operator.

Let x be a single digraph signal. The graph convolution defined with MGFT is denoted as

$$g_\omega * x = \mathcal{F}^{(q)^*}\left(g_\omega\left(\mathcal{F}^{(q)} x\right)\right), \tag{3.20}$$

where $g_\omega = \mathrm{diag}(\omega)$ is a learnable filter. **Figure 3.1** is a demonstration of MGFT-based graph convolution. Then, applying the method for fast computation in Section 3.4, we have

$$g_\omega * x \approx \tilde{\mathcal{F}}^{(q)^*}\left(g_\omega\left(\tilde{\mathcal{F}}^{(q)} x\right)\right). \tag{3.21}$$

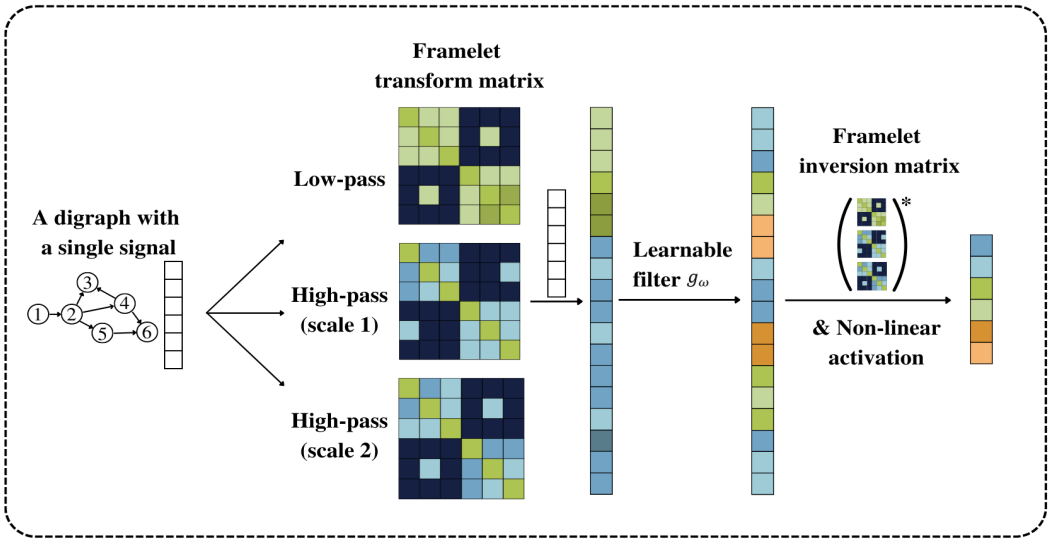

Figure 3.1: An example of Framelet-MagNet graph convolution. The input graph signal is transformed to the spectral domain by a left multiplication with the magnetic framelet transform matrix, which consists of 1 low-pass transform matrix $\mathcal{F}_{0,2}^{(q)}$ and 2 high-pass transform matrices $\mathcal{F}_{1,1}^{(q)}, \mathcal{F}_{1,2}^{(q)}$. Then, the spectral representation is filtered by a learnable filter. Lastly, it is converted back to the graph domain by the inversion matrix. After applying the non-linear activation. we will obtain a new representation of the original graph data.

Now, we construct convolutional layers for Framelet-MagNet while extending to multiple graph signals. Let $\mathbf{X} \in \mathbb{R}^{N\times d}$ be a matrix of d digraph signals. Then, the i^{th} magnetic framelet-based convolutional layer is defined as

$$\sigma(g_{\omega_i} * X_{i-1}) = \sigma\left(\tilde{\mathcal{F}}^{(q)^*}\left(\text{diag}(\omega_i)\left(\tilde{\mathcal{F}}^{(q)}(\mathbf{X}_{i-1}\mathbf{W}_i)\right)\right)\right), \tag{3.22}$$

where σ is a non-linear activation function, $g_{\omega_i} = \text{diag}(\omega_i)$ is a learnable filter, $\mathbf{X}_{i-1}$ is an $N \times D_{i-1}$ feature matrix with $\mathbf{X}_0$ being the original graph feature matrix, and $\mathbf{W}_i$ is a $D_{i-1} \times D_i$ matrix, where D_{i-1} and D_i are dimensions of the input and output channels.

Since the magnetic Laplacian is a complex-valued matrix, the new digraph representation produced by the convolutional layer in equation (3.22) is also complex-valued. For the final prediction, we need to unwind the complex representation to a real representation. Suppose we have C convolutional layers, then we will obtain an $N \times D_C$ complex-valued graph representation. The purpose of the unwind operator is to unwind this representation to an $N \times 2D_C$ real-valued

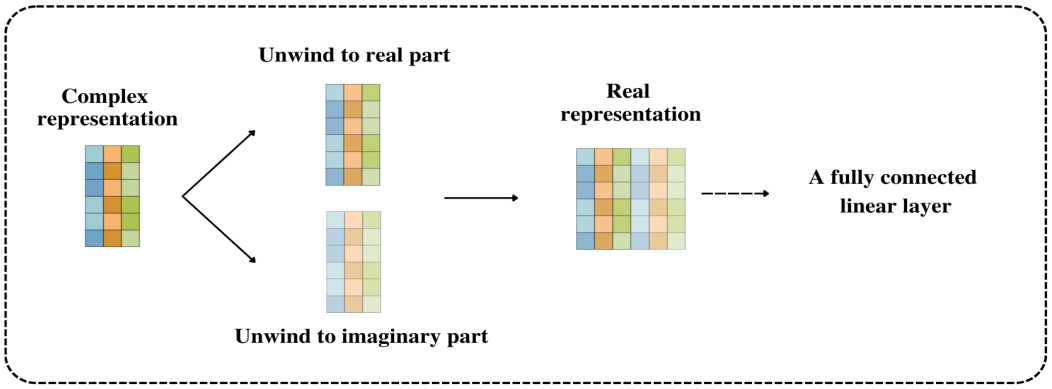

Figure 3.2: The unwind operation. Suppose we have a 6×3 complex representation generated from the convolutional layer(s). The unwind operator will first separate the real and imaginary parts, then concatenate them horizontally to obtain a 6×6 real representation.

representation before using it for prediction (see **Figure 3.2** as an example).

3.6 Framelet-MagNet Node Classification

The objective of graph node classification is to categorize nodes into several classes. For example, the node classification task based on dataset CORA_ML [58] aims to classify published papers into 7 classes, where each class represents a research topic (see more details in Chapter 4, Section 4.1). According to [2], the cross-entropy loss is a typical choice of the loss function for node classification tasks. Let $\mathcal{G}_d\{\mathcal{V}, \mathcal{E}\}$ be a digraph. Suppose that y_i is the true class that node v_i belongs to, and $\hat{y}_i$ is the predicted outcome. Then, the cross-entropy loss is given by

$$\text{Loss}(y_i, \hat{y}_i) = -\sum_{c=1}^{C} y_{v_i,c} \log(p_{v_i,c}),$$

where C is the number of classes, $y_{v_i,c}$ is a binary indicator if v_i belongs to class c, and $p_{v_i,c}$ is the predicted probability that v_i is in class c.

Because we will use the semi-supervised setting for our node-level experiment, the loss function is only computed over labelled nodes in $Label(\mathcal{V})$ as follows:

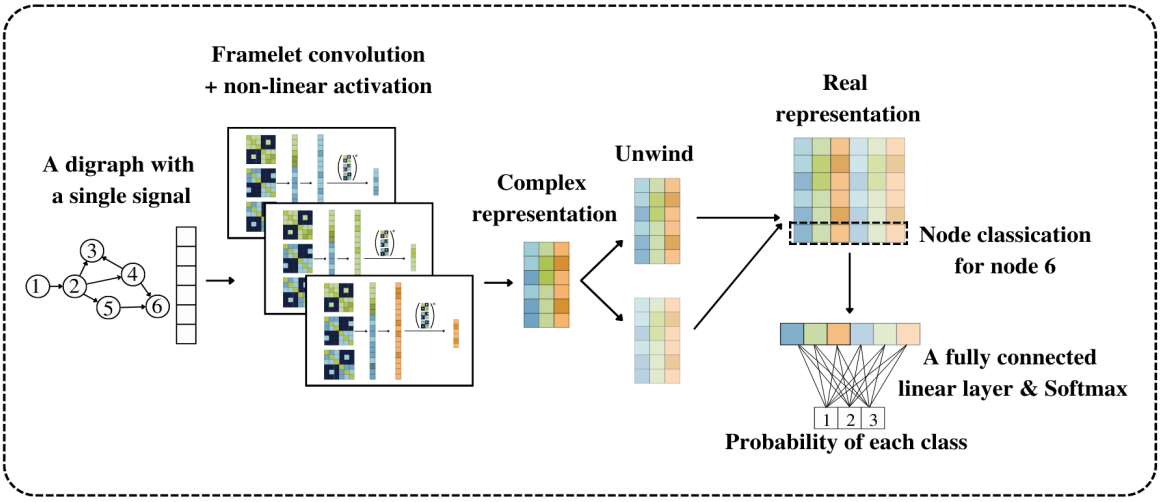

Figure 3.3: An example of Framelet-MagNet node classification (3 classes). We use the graph convolution in Figure 3.1 and three learnable filters. The input signal becomes a 6×3 complex representation after the convolutional layer. Then, we unwind the complex representation and use the real representation directly for node classification. Each row is corresponding to a node. For example, for node 6. we use the last row of the matrix to predict. Finally, we apply the fully connected layer and softmax to generate the probability of each class.

$$\text{Loss}_{\mathcal{V}} = \frac{\sum_{v_i \in Label(\mathcal{V})} \text{Loss}(y_i, \hat{y}_i)}{|Label(\mathcal{V})|},$$

where $|Label(\mathcal{V})|$ is the number of nodes in $Label(\mathcal{V})$.

For node classification tasks, we will use the real representation after the unwind operation directly for prediction. In **Figure 3.3**, we illustrate the whole node classification process with a 6-node digraph as an example.

3.7 Framelet-MagNet Link Prediction

Inspired by [21], we design four link-level tasks for Framelet-MagNet. Here are the descriptions of each task.

i) **Two-class link existence prediction**

This task is to predict whether a directed edge exists between two specified nodes where

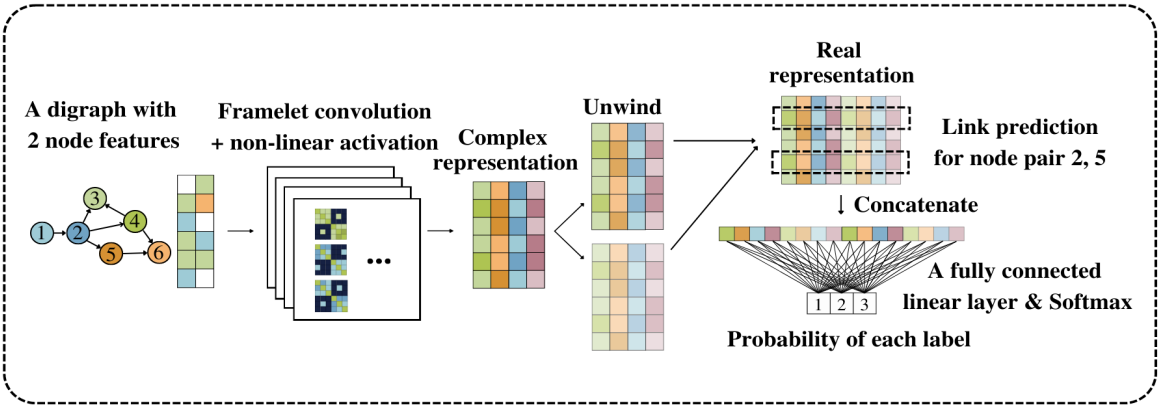

Figure 3.4: An example of Framelet-MagNet link prediction (3-class based task). We use the graph convolution in Figure 3.1 and set the output dimension of the convolutional layer to be 4. The input signal becomes a 6×4 complex representation after the convolutional layer. Then, we unwind the complex representation. Each row is corresponding to a node. To obtain the edge feature, we concatenate the rows associated with each node pair. For example, for node pair 2,5, we concatenate the second and fifth rows. Finally, we apply the fully connected layer and softmax to generate the probability of each label.

the edge direction is pre-specified.

Given a digraph $\mathcal{G}_d\{\mathcal{V}, \mathcal{E}\}$, we consider any two ordered nodes $v_i, v_j \in \mathcal{V}$. If $(v_i, v_j) \in \mathcal{E}$, then we label this edge as 0, otherwise we label it as 1. Therefore, when we refer to the adjacency matrix, if $\mathbf{A}(i, j) = 1$, edge (v_i, v_j) is labelled as 0, otherwise it is labelled as 1. Then, we try to predict the edge label of two given nodes.

In this task, the order of nodes matters. For example, if we predict that the label of $(v_i, v_j) = 1$, we can only conclude that the edge of this specific direction does not exist. We cannot conclude that there is no linkage between node v_i and node v_j, since it is possible that $(v_j, v_i) \in \mathcal{E}$.

ii) **Three-class link existence prediction**

Similar to two-class link existence prediction, our target in this task is to predict whether an edge exists between two given nodes. However, rather than having one single category for both "non-existence" and "existence of an edge in another direction", we have two separate categories instead.

Consider a digraph $\mathcal{G}_d\{\mathcal{V}, \mathcal{E}\}$ and any ordered node pair $v_i, v_j \in \mathcal{V}$. The edge can fall

into three potential categories. Firstly, if $(v_i, v_j) \in \mathcal{E}$, we label the edge as 0. Next, if $(v_j, v_i) \in \mathcal{E}$ instead, we label it as 1. Moreover, if there is no edge between v_i and v_j, we label the edge as 2, implying non-existence. Although we can distinguish non-existence and specific directions based on the labels defined in this task, the order of nodes still matters. Because we shall identify the direction of predicted edges through node order (i.e., distinguish the meanings of label 0 and label 1).

iii) **Two-class link direction prediction**

This task is to predict the direction of an edge conditional on its existence. For train, validation, and test sets, we only use linked node pairs.

Given a graph $\mathcal{G}_d\{\mathcal{V}, \mathcal{E}\}$, we consider any two ordered nodes $v_i, v_j \in \mathcal{V}$. The selection of node pairs is conditional on either $(v_i, v_j) \in \mathcal{E}$ or $(v_j, v_i) \in \mathcal{E}$. If $(v_i, v_j) \in \mathcal{E}$, we label it as 0. Or, if $(v_j, v_i) \in \mathcal{E}$, we label it as 1. Then, our objective is to predict the label of each edge.

iv) **three-class link direction prediction**

Akin to the two-class link direction prediction, we are still interested in the direction of an edge. Nevertheless, we abandon the requirement for strict edge existence in constructing train, validation, and test sets. Here we duplicate the label setting in the three-class link existence prediction, where we have a unique class for edge non-existence. Therefore, different from the two-class setting, we do not specially choose linked node pairs. Besides, this task is different from three-class link existence prediction because we evaluate the model performance with only linked node pairs.

One may notice that undirected edges are not clearly labelled in our task settings. So, as suggested by [21], we can discard all the undirected edges such that the digraphs are "noiseless". This is an expedient way to preclude the perturbation caused by undirected edges. Most digraphs only have a small proportion of undirected edges, for example, CORA_ML [58] only have 7% edges being undirected. So, removing undirected edges will not change the graph structure significantly.

Link prediction tasks are also classification tasks. So, we still use cross-entropy loss as the loss

function. In this case, classes are edge labels. Rather than calculating the cross-entropy loss for all graph nodes, we shall calculate based on node pairs for link tasks. If a task requires linked node pairs, then the loss is calculated based on only linked node pairs. Otherwise, it is based on all ordered node pairs.

Moreover, we shall pay attention to the unwind operation. Rather than feeding the unwound matrix directly into the final layers, we should concatenate the rows according to the node pairs. An example is shown in **Figure 3.4**. In this example, we apply Framelet-MagNet to a digraph with two node features to show the application on multiple graph signals.

Chapter 4

Experiments

To validate the power of our model, Framelet-MagNet, we will conduct three experiments, including node classification, link prediction, and denoising. Link prediction consists of four separate tasks: two-class link existence prediction, three-class link existence prediction, two-class link direction prediction, and three-class link direction prediction. Besides, denoising is designed based on the node classification experiment. In each experiment, we will compare Framelet-MagNet with 10 baseline models across several benchmark datasets. Particularly, we are interested in the comparison between Framelet-MagNet and MagNet [21]. Since MagNet adopts magnetic Laplacian-based Fourier transform in its convolutional layer, this comparison will clearly show the advantage of MGFT over the Fourier transform in enhancing network performance.

In Section 4.1, we will describe the characteristics of datasets used for our experiments. Then, in Section 4.2, we will have a brief introduction to the baseline models. In Section 4.3, Section 4.4, and Section 4.5, we will present the implementation details and results [1] of each experiment.

[1] Although our experimental settings are similar to the experiments in [21], our experiment results are not necessarily similar. Based on a conversation with the author of [21], this is mainly due to the update of datasets.

4.1 Benchmark Datasets

We will use five, seven, and two publicly available datasets for node classification, link prediction, and denoising experiments, respectively. The datasets for the node classification experiment are CORA_ML, CITESEER, CORNELL, TEXAS and WISCONSIN. These five datasets will also be used in the link prediction experiment. Besides, we include two Wikipedia datasets, CHAMELEON and SQUIRREL in link prediction tasks. It is worth noting that they are not included in the node classification experiment because their nodes do not have classes. For denoising, we will use CORA_ML and TEXAS. Now we will briefly describe the characteristics of each dataset.

i) **Citation Datasets — CORA_ML and CITESEER**

CORA is a widely-used citation graph in GNN experiments introduced by [59]. Nodes in Cora are published papers. Node attributes are binary-valued vectors indicating the existence or absence of a specific word in a pre-specified dictionary. CORA_ML is a relatively smaller subset sampled from CORA by [58]. There are 2,995 nodes, 8,416 edges and 2,879 node attributes in CORA_ML. Different from CORA, node features in CORA_ML are normalized. The nodes are classified into 7 classes based on their paper topics. The existence of an edge is determined by whether a paper cites another paper. To illustrate, if paper A cites another paper B, then we will have an edge starting from A and pointing at B. Among all the edges, 93.9% are directed, so this graph is nearly oriented.

CITESEER is also a citation graph whose nodes are published papers, and edges denote citation relationships constructed following the same rule as CORA_ML. These papers are classified into 6 classes according to the scientific subareas they belong to. The version of the CITESEER dataset in our experiment is also from [58]. In this version, we have 3,312 nodes, 4,715 edges, and 3,703 node features. The proportion of directed edges is also very high in CITESEER, which is 95%.

ii) **WebKB — CORNELL, TEXAS, and WISCONSIN**

The WebKB datasets are a collection of webpage graphs introduced by Carnegie Mellon University [60]. Each dataset contains information from the computer science department of a University. The nodes represent websites while the edges are hyperlinks between them. In addition, the node attributes are bag-of-words (binary) extracted from each website. All the nodes are manually classified into five categories: *student, project, course, staff, and faculty.* In webpage datasets, if webpage B is the destination of a hyperlink in webpage A, then we have a directed edge from node A to node B. Usually, webpage graphs are directed because most hyperlinks are uni-directional.

Three WebKB datasets, CORNELL, TEXAS, and WISCONSIN, were selected by [60] for GNN experiments. For our experiments, we will use the same version as in [60]. In CORNELL, we have 183 nodes, 295 edges, and 1,703 features, where 86.9% edges are directed. In TEXAS, we have 183 nodes, 309 edges, and 1,703 features, where 76.6% edges are directed. Lastly, in WISCONSIN, we have 251 nodes, 499 edges, and 1,703 features, where 77.9% edges are directed. One may notice that the number of node features in them is identical. This is because of their use of the same bag-of-words representation.

iii) **WikipediaNetwork — CHAMELEON and SQUIRREL**

WikipediaNetwork datasets [61] are page-to-page graphs, including CHAMELEON, CROCODILE, and SQUIRREL. Each node refers to articles related to the dataset name. For example, in SQUIRREL, the nodes are articles talking about squirrels. These articles were collected from English Wikipedia in December 2018. Then edges were constructed reflecting the hyperlinks between articles, akin to webpage graphs. Moreover, the node features were designed to denote the presence of specific nouns and the mean monthly page views of each article between October 2017 and November 2018. The original purpose of WikipediaNetwork datasets was for a node regression task, in which the response variable is mean monthly page views. Consequently, the nodes were not clearly classified. Therefore, we only use them for link prediction experiments. Furthermore, since CROCODILE is a large graph with 11,631 nodes, we will not use it due to a consideration of time budget.

In our link prediction experiments, we will use the same version of datasets as in [61]. We

have 2,277 nodes with 36,101 edges in CHAMELEON and 5,201 nodes with 217,073 edges in SQUIRREL. Besides, 73.9% edges in CHAMELEON are directed, and 82.8% edges in SQUIRREL are directed. Although we have 2,325 and 2,089 node features in CHAMELEON and SQUIRREL, respectively, we will not use them for link prediction. Alternatively, we will use node in-degrees and node out-degrees following the practice in [21]. More details will be discussed in Section 4.4.

Our choice of datasets is driven by three reasons. Firstly, all of these datasets were adopted in the MagNet experiments. We would like to follow most of the original experimental settings such that the results are more illustrative. Next, due to the limitation of magnetic Laplacian-based neural networks (i.e., not applicable to weighted mixed graphs), we require the datasets to be unweighted. This is satisfied by all seven datasets. Lastly, these datasets have been well-studied in the relevant literature, and they are structured and clean. So, we can skip data pre-processing before our experiments.

4.2 Baseline Models

Baseline models in our experiments are used as a comparison to evaluate the performance of Framelet-MagNet. We have two classic spectral GCNNs: ChebNet [24] and GCN [7]. We also have four typical spatial GCNNs: APPNP [18], GraphSAGE [16], GIN [19] and GAT [17]. Furthermore, to investigate the state-of-the-art digraph networks, we also compare with DGCN [36], Digraph [20], DiGCN [20], and MagNet [21]. Note that the DGCN in our experiments is identical to the DGCN-Tong in Chapter 2.

In link prediction tasks, we adopt a GCN architecture that is slightly different from the one in Section 1.1 of Chapter 1. This architecture can accept the asymmetric digraph Laplacian as input. Basically, we replace $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ with $\mathbf{D}^{-1}\mathbf{A}$. For more details, we refer to [62].

We use all 10 baseline models in node classification and link prediction. Besides, we only use MagNet and GCN as baseline models in denoising.

4.3 Node Classification

4.3.1 Implementation Details

The node classification experiment will be carried out over 5 publicly available datasets. We will use the original values of node features as the model input. Moreover, the node classification experiment is based on a semi-supervised task (i.e., test data are included in training with their labels being removed).

To alleviate the effect of randomness and measure the robustness of each neural network, we randomly generate 10 subsets from each dataset following the experiments in [21]. Then the models are trained, validated, and tested with these 10 subsets. For WebKB datasets, following the rule in [21], we use 60%, 20%, and 20% data for training, validation, and testing, respectively. For citation datasets, CORA_ML and CITESEER, following the experiments in [7], we use 20 labels from each class for training, 500 labels for validation, and the rest for testing.

We will obtain a classification test accuracy of each model on each subset. The results of our interests are the average test accuracy across 10 subsets and the associated standard deviation. Both of them will be expressed in the form of a percentage (%). In addition, for magnetic Laplacian-based approaches, MagNet and Framelet-MagNet, we will also report the optimal value of hyperparameter q. Moreover, we will provide the framelet type that contributes to the best performance of Framelet-MagNet.

The models for testing are selected based on their validation accuracy. We train all the models with 200 epochs since the increase of validation accuracy terminates before 200 epochs based on our observation. At each epoch, models are kept if the current validation accuracy is higher than the best recorded accuracy in previous epochs. Finally, after 200 epochs, the latest kept model will be used for testing.

Here are the basic settings of hyperparameters shared by all models. According to [21], the number of filters in the convolutional layer and the learning rate are essential for the model performance. So, we tune the number of filters in $(16, 32, 64)$ ($(6, 11, 21)$ for DiGCN) and

the learning rate in $(0.001, 0.005, 0.01)$. We choose Adam as the optimizer for all the models. We include weight decay with $\lambda = 5e^{-4}$. In general, we set two convolutional layers for every graph neural network except for Framelet-MagNet. As we will see later in the results, one convolutional layer is sufficient for Framelet-MagNet to achieve outstanding performance in the node classification task. So, we implement only one layer to mitigate the computational overhead. Then, we include a dropout layer with 50% dropout probability before the final output layer. The output layer for the node classification task is based on the softmax function.

Now, we will concisely introduce some model-specific hyperparameters and settings. Most baseline models are introduced in Chapter 1 or Chapter 2, but spatial methods are less relevant and have no detailed descriptions. So, for spatial methods, we refer the definition of their hyperparameters to the original papers [16, 17, 18, 19].

For MagNet and Framelet-MagNet, we will use the normalized magnetic Laplacian. The charge parameter q will be tuned in a searching space of 4 values, $(0.00, 0.05, 0.15, 0.25)$. If the optimal value of q is 0.00, then no directional information is encoded by the trained model. On the contrary, the larger the value of q, the more directed information is encoded. As we have discussed before, MagNet is constructed based on a traditional GCNN architecture. In our experiment, we choose the ChebNet architecture with $k = 1$, since experiments in [21] have shown the effectiveness of adopting this particular architecture. For Framelet-MagNet, available framelet types include `Haar, Linear, Quadratic, Sigmoid`, and `Entropy`. We set $\alpha = 20$ for `Sigmoid` and $\alpha = 0.5$ for `Entropy`. The dilation level $s = 2$. To enable fast computation, we set Chebyshev polynomial degree to either 2 or 4. From the perspective of graph spectral theory, a degree of 2 emphasizes the significance of the closest neighbor of each node, while a degree of 4 highlights the importance of the 3-hop neighborhood (i.e., neighbors that can be reached within 3 steps).

For classic spectral approaches, GCN and ChebNet ($k = 1$), we use symmetrized adjacency matrices as suggested by [7] and [24]. To symmetrize adjacency matrices, we replace all directed edges straightforwardly with undirected edges. For spatial methods, APPNP, GraphSAGE, GIN, and GAT, we will try both symmetric and asymmetric adjacency matrices and record the better

results. For APPNP, we set the number of power iteration steps $K = 10$ as suggested by [18]. Since APPNP is a Paper-Rank based model, we tune the teleport probability η in the searching space $(0.05, 0.10, 0.15, 0.20)$. For GIN, we use the simplified architecture by setting $\epsilon = 0$ [19]. In addition, for GAT, we tune the number of heads among three values $(2, 4, 8)$. As for Digraph and DiGCN, we also need to tune the teleport probability in $(0.05, 0.10, 0.15, 0.20)$. Moreover, we tune the number of filters in the DiGCN convolutional layer in $[6, 11, 21]$ because this model constructs a three-channel Laplacian tensor, which means the number of filters is tripled.

To summarize, there are two types of hyperparameters to be tuned in the model training process. Some are mutually shared across all models, such as the number of filters and the learning rate. Others are model-specific, such as the Paper-Rank teleport probability. Since neural networks are heavily parameterized, we only tune the hyperparameters that have influential impacts on the classification performance. The hyperparameter tuning process will be guided by grid search as common practice.

4.3.2 Experiment Results

The experiment results of the node classification task are exhibited in **Table 4.1** [2]. Overall, our model, Framelet-MagNet, presents a good performance across all five digraph datasets. In four out of five datasets, Framelet-MagNet achieves the best prediction accuracy. For WISCONSIN, both ChebNet and Framelet-MagNet achieve the highest classification accuracy of 85.1%. However, our model is only the second best in the node classification task for TEXAS, while MagNet performs the best with an accuracy of 82.4%. In WebKB datasets, magnetic Laplacian-based models show an absolute advantage over most baseline models. A potential explanation is that these baseline models need more training data to express their power, but WebKB datasets are quite small (see details in Section 4.1).

Noticeably, Framelet-MagNet outperforms MagNet on four datasets, CORA_ML, CITESEER, CORNELL, and WISCONSIN, by 5.1%, 3.2%, 2.4%, and 1.8%, respectively. This demonstrates the power of framelet transform in digraph node classification tasks. Nevertheless, we also

[2] For all the tables in this thesis, we present the results in the from of $A \pm B$, where A is the average accuracy, and B is the standard deviation of accuracy. The results in **bold** are the best results among all models.

Table 4.1: Experiment Results: Node Classification Accuracy (%)

Models	CORA_ML	CITESEER	CORNELL	TEXAS	WISCONSIN
ChebNet	60.8 ± 3.3	53.3 ± 2.6	74.1 ± 2.8	78.4 ± 5.8	**85.1 ± 3.8**
GCN	69.7 ± 2.0	60.1 ± 2.6	42.4 ± 5.7	58.4 ± 3.9	51.0 ± 6.3
APPNP	79.4 ± 2.7	66.7 ± 2.0	42.7 ± 5.5	58.9 ± 4.3	53.1 ± 6.9
GraphSAGE	78.7 ± 1.1	66.4 ± 1.3	69.2 ± 3.5	79.7 ± 6.1	76.5 ± 5.6
GIN	78.7 ± 1.8	63.9 ± 2.2	48.1 ± 5.0	62.7 ± 7.2	57.1 ± 7.5
GAT	81.2 ± 2.0	66.2 ± 1.7	45.4 ± 10.4	58.1 ± 5.0	57.6 ± 5.5
DGCN	79.8 ± 1.5	65.9 ± 1.4	65.1 ± 6.1	71.8 ± 7.0	66.3 ± 6.8
Digraph	76.7 ± 1.9	62.9 ± 1.8	54.6 ± 6.8	63.2 ± 6.1	58.8 ± 3.6
DiGCN	76.5 ± 1.6	61.6 ± 1.9	54.3 ± 7.5	60.3 ± 5.3	62.4 ± 5.8
MagNet	78.7 ± 2.2	64.6 ± 2.2	74.6 ± 4.4	**82.4 ± 6.1**	83.3 ± 3.3
q	0.00	0.05	0.15	0.05	0.15
Framelet-MagNet	**83.8 ±1.4**	**67.8 ±1.5**	**77.0 ± 3.5**	81.9 ± 3.8	**85.1 ± 5.2**
q	0.00	0.05	0.25	0.25	0.25
Framelet type	`Sigmoid`	`Sigmoid`	`Quadratic`	`Quadratic`	`Quadratic`

notice that Framelet-MagNet fails to outperform MagNet for TEXAS. Our model generates an accuracy of 81.9%, which is 0.5% lower than MagNet. Although the average accuracy of MagNet is slightly higher, its standard deviation of accuracy across ten subsets is 6.1%, implying that the 0.5% superiority may not be robust. Namely, if we try another ten randomly selected subsets, it is possible that MagNet cannot sustainably outperform Framelet-MagNet.

In terms of the variation in classification accuracy, Framelet-MagNet has a lower standard deviation compared to MagNet in four out of five datasets, including CORA_ML, CITESEER, CORNELL, and TEXAS. Meanwhile, compared with other benchmark models, Framelet-MagNet also enjoys a comparably lower standard deviation in its results, showing strong robustness in its node classification performance. Accordingly, the performance of Framelet-MagNet will be more stable in future node classification tasks. This is desirable in the context of graph-based tasks, because graphs are frequently updated in dynamic real-life practices.

For citation datasets, Framelet-MagNet selects $q = 0.00$ for CORA_ML and $q = 0.05$ for CITESEER. Likewise, MagNet also chooses the same q values for these two datasets. According to [21], this means directional information is not useful for these models to achieve the best performance, implying that symmetrizing adjacency matrices are more suitable in the node classification tasks for citation graphs. Intuitively, in citation analysis, if a paper cites or is cited by another paper, it suggests that these two papers are based on similar topics. So, whether the paper cites or is cited is not important in deciding the category it belongs to. For example, in

CITESEER where all nodes are scientific publications, if a paper cites a machine learning paper, then it is likely that this paper is related to machine learning as well. Similarly, if this paper is cited by another machine learning paper, we may also conclude that it is relevant to machine learning. So, the directional information does not impact our judgement. This explains our observation.

On the contrary, Framelet-MagNet adopts $q = 0.25$ for three WebKB graphs while MagNet also use high q values for CORNELL and WISCONSIN. Moreover, we also observe that Framelet-MagNet always picks higher q values than MagNet for WebKB graphs. While MagNet chooses $q = 0.15$, $q = 0.05$, and $q = 0.15$ for CORNELL, TEXAS, and WISCONSIN, Framelet-Magnet adopts $q = 0.25$ for all these datasets. This means Framelet-MagNet can encode more directional information than MagNet in its training process. Especially for CORNELL and WISCONSIN, our model compares favourably to MagNet by incorporating more directional information for classification. Since the edges of WebKB datasets represent hyperlinks between webpages, the direction of edges matters. For instance, we may find a hyperlink on a faculty page leading to its teaching staff, but we may not find a way back to the faculty page on the staff page. So, leveraging the directional information helps to enhance the performance of magnetic Laplacian-based neural networks. However, this is not the case in the node classification task for TEXAS. Since MagNet with $q = 0.05$ performs better than Framelet-MagNet with $q = 0.25$, we conclude that directional information may be less important for TEXAS node classification.

4.4 Link Prediction

4.4.1 Implementation Details

According to Section 3.7 of Chapter 3, we have four different link prediction tasks, including two-class link existence prediction, three-class link existence prediction, two-class link direction prediction, and three-class link direction prediction.

Akin to the node classification experiment, we also use 10 subsets for link prediction. Following [21], we remove 5% edges for validation, 15% edges for testing, and we keep the rest of edges for

training. Thus, the number of nodes in each graph remains constant after the train/validation/test split. Although our method does not require the digraphs to be strongly connected, we maintain the connectivity of the training sets to preserve more graph structural information. This is accomplished by calculating the minimum spanning tree. This spanning tree contains all the vertices in the graph and the minimum number of edges that maintain the connectivity between nodes. Therefore, we can avoid destroying the graph connectivity by removing edges outside of the tree for validation and test sets. The edges to be removed are selected randomly.

Different from the node classification experiment, we will not use the original node features in the link prediction experiment. Alternatively, we conduct each task with node in-degree and node out-degree as features such that the model can learn directly from the graph structure via the adjacency matrix [21]. Our choice is based on the assumption that structural information is more useful than node attributes in link prediction tasks, because linkage implies the underlying relationship between nodes, and the relationship is encoded in the graph structure.

For an ordered node pair, the response variable is the edge label while the features are the in-degrees and out-degrees of these two nodes. More specifically, the in-degree is the number of edges pointed into a node, and the out-degree is the number of edges pointing out of a node. Mathematically, with the adjacency matrix $\mathbf{A}$, the in-degree of node v_i is calculated as

$$v_i^{(in)} = \sum_{v_j \in \mathcal{V}} \mathbf{A}(j, i),$$

and the out-degree of this node is calculated as

$$v_i^{(out)} = \sum_{v_j \in \mathcal{V}} \mathbf{A}(i, j).$$

Finally, other implementation details such as hyperparameters are almost identical to the node classification experiment. The only difference is that we set a learning rate of 0.001 for all the models, because the number of available samples for link prediction tasks is much larger than for node classification.

4.4.2 Experiment Results of Link Existence Prediction

In this section, we will discuss the results from link existence tasks (see **Table 4.2** and **Table 4.3**). The metric of our primary interest is prediction accuracy, measured by the proportion of correctly categorized edges in the test set. Besides, we will analyze model robustness, the optimal q values for magnetic Laplacian-based models, and the best framelet type for Framelet-MagNet.

Table 4.2: Experiment Results: Two-Class Link Existence Prediction Accuracy (%)

Models	CORA_ML	CITESEER	CORNELL	TEXAS	WISCONSIN	CHAMELEON	SQUIRREL
ChebNet	50.1 ± 0.1	70.9 ± 15.7	49.7 ± 1.4	50.6 ± 1.2	50.0 ± 0.0	50.1 ± 0.0	50.2 ± 0.0
GCN	73.1 ± 5.3	76.1 ± 0.0	51.1 ± 3.7	51.5 ± 4.0	53.4 ± 6.0	**89.8 ± 0.5**	90.3 ± 1.5
APPNP	69.5 ± 3.9	76.2 ± 0.2	61.4 ± 8.0	69.5 ± 9.8	65.2 ± 3.9	87.1 ± 4.9	88.8 ± 0.4
GraphSAGE	67.2 ± 3.7	76.1 ± 0.1	63.5 ± 9.4	72.3 ± 4.2	70.6 ± 4.4	86.0 ± 0.5	83.6 ± 14.3
GIN	75.0 ± 3.4	76.1 ± 0.0	65.5 ± 8.5	69.4 ± 5.9	69.5 ± 5.2	83.9 ± 7.1	84.0 ± 5.9
GAT	50.0 ± 0.2	76.1 ± 0.1	51.4 ± 3.5	50.3 ± 1.6	51.5 ± 1.6	50.4 ± 1.0	61.9 ± 14.6
DGCN	60.6 ± 7.6	76.1 ± 0.1	60.8 ± 10.1	65.3 ± 7.5	61.3 ± 6.8	86.3 ± 1.4	76.1 ± 8.5
Digraph	76.7 ± 1.9	62.9 ± 1.8	54.6 ± 6.8	63.2 ± 6.1	58.8 ± 3.6	83.9 ± 11.4	85.3 ± 12.0
DiGCN	72.8 ± 7.7	76.1 ± 0.0	65.6 ± 12.1	74.0 ± 14.2	69.0 ± 10.4	88.9 ± 0.6	89.7 ± 0.3
MagNet	77.1 ± 1.4	75.5 ± 1.4	68.2 ± 7.0	77.7 ± 9.2	76.0 ± 7.7	**89.8 ± 0.5**	90.3 ± 0.3
q	0.15	0.25	0.15	0.15	0.25	0.15	0.15
Framelet-MagNet	**78.1 ± 1.2**	**77.0 ± 1.4**	**73.8 ± 6.0**	**83.0 ± 7.2**	**79.0 ± 4.6**	89.7 ± 0.4	**90.7 ± 0.1**
q	0.15	0.15	0.25	0.25	0.25	0.25	0.25
Framelet type	`Haar`	`Quadratic`	`Haar`	`Haar`	`Sigmoid`	`Linear`	`Sigmoid`

Table 4.3: Experiment Results: Three-Class Link Existence Prediction Accuracy (%)

Models	CORA_ML	CITESEER	CORNELL	TEXAS	WISCONSIN	CHAMELEON	SQUIRREL
ChebNet	47.1 ± 10.3	48.0 ± 10.5	48.3 ± 10.1	50.7 ± 11.1	54.1 ± 1.9	52.5 ± 0.2	51.6 ± 0.1
GCN	50.6 ± 0.1	44.6 ± 13.9	52.0 ± 1.6	49.7 ± 13.1	50.5 ± 9.9	53.9 ± 3.2	63.8 ± 4.1
APPNP	51.5 ± 1.5	51.5 ± 0.4	52.3 ± 1.2	54.1 ± 3.1	50.0 ± 11.1	65.1 ± 0.4	65.4 ± 0.2
GraphSAGE	50.6 ± 0.1	51.5 ± 0.4	52.3 ± 1.2	54.1 ± 3.1	54.1 ± 1.9	54.4 ± 4.1	53.5 ± 4.1
GIN	58.4 ± 0.6	52.5 ± 1.5	53.1 ± 2.2	55.1 ± 4.7	55.1 ± 2.8	**65.1 ± 0.4**	**65.5 ± 0.2**
GAT	50.6 ± 0.0	51.5 ± 0.4	52.0 ± 1.6	54.1 ± 3.1	54.1 ± 1.9	52.5 ± 0.0	51.6 ± 0.1
DGCN	50.6 ± 0.1	51.5 ± 0.4	52.3 ± 1.2	53.8 ± 3.4	54.1 ± 1.9	52.5 ± 0.2	51.6 ± 0.1
Digraph	50.6 ± 0.1	47.9 ± 10.8	52.5 ± 2.0	50.3 ± 11.8	53.8 ± 2.4	52.5 ± 0.2	51.6 ± 0.1
DiGCN	50.7 ± 0.3	51.5 ± 0.4	52.0 ± 1.4	53.8 ± 3.4	53.9 ± 2.0	60.0 ± 5.6	62.5 ± 5.5
MagNet	50.6 ± 0.2	51.5 ± 0.4	52.5 ± 0.9	56.4 ± 4.9	54.7 ± 1.8	52.8 ± 0.5	52.0 ± 1.0
q	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Framelet-MagNet	**58.5 ± 3.5**	**53.1 ± 2.4**	**53.4 ± 1.5**	**56.7 ± 4.9**	**55.2 ± 0.3**	64.3 ± 1.8	57.2 ± 6.9
q	0.15	0.25	0.15	0.25	0.25	0.25	0.25
Framelet type	`Haar`	`Linear`	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Sigmoid`

In two-class link existence prediction, Framelet-MagNet achieves the best accuracy for all datasets except for CHAMELEON. The models that perform the best on CHAMELEON are MagNet and GCN. Yet, the accuracy of Framelet-MagNet is only 0.1% lower than MagNet and GCN, which is not very significant. For other datasets, Framelet-MagNet improves state-of-the-art performance by 0.4% $\sim$ 5.6%. In general, magnetic Laplacian-based methods perform better

than other models for CORA_ML, CORNELL, TEXAS, and WISCONSIN. For CITESEER, most models achieve very similar results, while Framelet-MagNet obtains the highest accuracy, 77.0%. In addition, for CHAMELEON and SQUIRREL, all models perform quite well except for ChebNet and GAT. Since ChebNet cannot process with the asymmetric adjacency matrix, we train it with symmetrized adjacency matrix, which means no directional information is included in ChebNet training. So, it is doomed that ChebNet will not perform well for all link related tasks.

In three-class link existence prediction, the prediction accuracy is comparably low across all models. For citation and WebKb datasets, no model can achieve prediction accuracy of more than 60%. For CHAMELEON and SQUIRREL, the results are slightly better, but they are still very undesirable compared with the two-class results. This may be due to the difficulty in identifying edges of different directions. Later, we will observe the same performance deterioration in three-class link direction prediction.

In two-class existence predictions, Framelet-MagNet has standard deviations that are $0.1\% \sim 3.1\%$ lower than MagNet, showing its superiority in robustness over the Fourier-based model. Compared with the other nine models, Framelet-MagNet presents good stability in CORA_ML, CORNELL, CHAMELEON, and SQUIRREL.

Compared with the node classification results, MagNet and Framelet-MagNet tend to choose larger q values for link predictions, even with the citation datasets. This highlights the importance of directional information in edge existence prediction.

Framelet-MagNet tends to choose quasi-framelet, `Sigmoid`, for three-class tasks. As we have discussed before, `Sigmoid` allows the algorithm to conduct double regulation on graph signals, enabling more sophisticated frequency filtering. Since the complexity of three-class link existence prediction is higher, we suggest that we should implement more sophisticated tools for complicated tasks. By contrast, there is no obvious preference for framelet type in two-class tasks. So, in two-class tasks, the choice of framelet type is more data-driven, mainly depending on the characteristics of each dataset.

4.4.3 Experiment Results of Link Direction Prediction

Results of the link direction prediction experiment is exhibited in **Table 4.4** and **Table 4.5**. To interpret the results, we will mainly focus on prediction accuracy. In addition, we will also pay attention to the optimal q values for magnetic Laplacian-based models, and any other intriguing findings.

Table 4.4: Experiment Results: Two-Class Link Direction Prediction Accuracy (%)

Models	CORA_ML	CITESEER	CORNELL	TEXAS	WISCONSIN	CHAMELEON	SQUIRREL
ChebNet	50.1 ± 0.2	50.0 ± 0.1	49.6 ± 10.3	50.3 ± 1.3	50.3 ± 0.5	50.0 ± 0.0	50.0 ± 0.0
GCN	79.1 ± 1.5	56.9 ± 8.7	52.0 ± 2.8	53.7 ± 8.9	55.6 ± 7.1	96.8 ± 0.6	97.2 ± 0.1
APPNP	81.9 ± 0.9	72.3 ± 2.1	70.3 ± 10.9	81.0 ± 4.7	73.7 ± 5.3	97.4 ± 0.2	97.5 ± 0.1
GraphSAGE	69.1 ± 0.5	70.6 ± 2.0	69.0 ± 7.4	77.0 ± 5.7	73.9 ± 4.4	94.2 ± 0.3	92.2 ± 0.1
GIN	84.2 ± 0.9	70.6 ± 2.0	77.0 ± 7.1	85.0 ± 4.0	79.0 ± 4.6	97.6 ± 0.2	97.8 ± 0.2
GAT	50.0 ± 0.6	50.1 ± 0.3	50.7 ± 3.1	50.9 ± 1.9	52.8 ± 1.9	51.6 ± 2.2	59.0 ± 13.8
DGCN	70.9 ± 1.4	65.7 ± 4.8	58.7 ± 5.1	65.9 ± 7.1	68.0 ± 6.1	93.7 ± 6.4	94.9 ± 0.5
Digraph	73.2 ± 11.7	66.1 ± 16.1	49.1 ± 2.7	50.8 ± 2.1	61.1 ± 12.8	92.2 ± 14.1	92.2 ± 14.1
DiGCN	83.4 ± 1.5	84.3 ± 1.7	73.3 ± 15.3	82.7 ± 13.9	80.4 ± 11.1	97.4 ± 0.2	97.1 ± 0.1
MagNet	87.0 ± 0.6	86.8 ± 1.3	78.6 ± 10.6	84.3 ± 9.1	83.9 ± 5.0	97.7 ± 0.2	97.8 ± 0.1
q	0.15	0.05	0.25	0.25	0.25	0.25	0.25
Framelet-MagNet	**88.5 ± 1.0**	**90.4 ± 1.0**	**86.7 ± 5.7**	**93.1 ± 5.5**	**89.4 ± 3.8**	**97.8 ± 0.2**	**97.9 ± 0.1**
q	0.15	0.15	0.25	0.25	0.25	0.25	0.25
Framelet type	`Haar`	`Linear`	`Linear`	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Haar`

Table 4.5: Experiment Results: Three-Class Link Direction Prediction Accuracy (%)

Models	CORA_ML	CITESEER	CORNELL	TEXAS	WISCONSIN	CHAMELEON	SQUIRREL
ChebNet	43.0 ± 12.0	43.7 ± 13.0	43.5 ± 13.8	46.9 ± 15.5	45.0 ± 14.4	44.9 ± 14.3	43.9 ± 13.2
GCN	43.1 ± 12.1	43.8 ± 12.8	43.9 ± 13.2	47.1 ± 15.3	45.1 ± 14.3	70.2 ± 9.7	80.6 ± 11.7
APPNP	61.8 ± 2.2	52.2 ± 0.2	**66.7 ± 7.4**	56.7 ± 1.3	54.4 ± 1.1	**80.2 ± 0.6**	82.1 ± 2.0
GraphSAGE	50.8 ± 0.1	52.2 ± 0.2	52.7 ± 0.8	56.7 ± 1.3	54.4 ± 1.1	55.0 ± 2.4	53.3 ± 2.2
GIN	57.8 ± 7.8	56.2 ± 2.2	52.8 ± 0.6	56.9 ± 1.4	55.3 ± 2.1	69.0 ± 13.1	71.9 ± 9.4
GAT	50.8 ± 0.1	52.2 ± 0.2	52.8 ± 0.6	56.7 ± 1.3	54.4 ± 1.1	54.2 ± 0.1	52.5 ± 0.0
DGCN	50.8 ± 0.1	52.2 ± 0.2	52.8 ± 0.6	56.7 ± 1.3	54.1 ± 0.7	54.2 ± 0.1	53.7 ± 3.5
Digraph	48.2 ± 7.9	49.4 ± 8.5	53.3 ± 10.7	49.8 ± 8.7	50.8 ± 9.7	51.1 ± 9.4	49.6 ± 8.6
DiGCN	54.1 ± 3.6	52.2 ± 0.2	52.7 ± 0.9	56.7 ± 1.3	54.3 ± 1.2	65.4 ± 10.4	75.5 ± 12.9
MagNet	52.5 ± 2.3	53.0 ± 1.6	57.5 ± 4.3	63.1 ± 7.4	60.2 ± 6.8	56.8 ± 3.3	54.4 ± 4.0
q	0.25	0.25	0.25	0.25	0.25	0.25	0.15
Framelet-MagNet	**64.2 ± 0.9**	**64.6 ± 0.8**	61.0 ± 1.6	**70.5 ± 2.8**	**69.7 ± 2.5**	70.7 ± 1.2	**82.3 ± 0.4**
q	0.15	0.15	0.25	0.25	0.25	0.25	0.25
Framelet type	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Sigmoid`	`Sigmoid`

Overall, Framelet-MagNet achieves the highest accuracy across all seven datasets. Especially, the results for WikipediaNetwork datasets, CHAMELEON and SQUIRREL, are extremely high (97.8% and 97.9% respectively). For other datasets, our model also obtains very impressive results, ranging from 86.7% to 93.1%. Compared with MagNet, the accuracy improvements vary from 0.1% to 8.8%. Except for magnetic Laplacian-based networks, DiGCN also shows its

strength in this task, which shows the power of complicated Laplacian design in link direction prediction. Nevertheless, the other two digraph neural networks, Digraph and DGCN, fail to perform well for most datasets. On the contrary, spatial methods, APPNP, GraphSAGE, and GIN, generally produce satisfying results. Although their performance is not as good as magnetic Laplacian-based models, it shows the practical value of these models considering their natural extension to digraphs.

However, there is an overall deterioration in prediction accuracy in the three-class task. We can see that the performance of MagNet is much worse compared to its results in the two-class task. The results of all datasets drop by 31.2% on average. Likewise, the performance of Framelet-MagNet also deteriorates in the three-class task, with its prediction accuracy decreasing by 23.0% on average. Moreover, we can see a pervasive downturn in the accuracy of other models as well. Therefore, we suggest that the state-of-the-art approaches are not sufficiently competent for the design of three-class direction prediction.

Here, we propose two possible solutions to obtain better results. On the one hand, collecting more training data allows graph neural networks to encode more useful information and make better decisions. For citation and WebKB datasets, most models can only achieve less than 60% prediction accuracy. However, the results are comparably better for WikipediaNetwork datasets, CHAMELEON and SQUIRREL. This is mainly due to the larger sample size of WikipediaNetwork datasets, implying that the design of three-class direction prediction requires more training data than the two-class task to obtain a good outcome. On the other hand, sophisticated methods are more suitable for complicated tasks. In general, Framelet-MagNet successfully outperforms other models for most datasets. Compared with MagNet, Framelet-MagNet improves its performance by 3.5% to 27.9%. Equipped with more a sophisticated signal processing system, Framelet-MagNet manages to stay competitive when the task complexity increases (from two classes to three classes). Therefore, we suggest that it is better to conduct the three-class task with more sophisticated methods.

In both tasks, Framelet-MagNet and MagNet select the same optimal q for CORNELL, TEXAS, WISCONSIN, and CHAMELEON, which is 0.25. This indicates full leverage of directional

information. For citation datasets, CORA_ML and CITESEER, Framelet-MagNet adopts $q = 0.15$ in all the cases while MagNet chooses lower q values for two classes prediction and higher q values for three classes prediction. By intuition, higher q allows the algorithm to encode more directional information, providing a more solid basis for direction prediction. However, Framelet-MagNet adopts a lower q to obtain a better prediction accuracy in the three-class task. It may suggest that when we preclude partial directional information, the model will be more accurate at predicting link non-existence (i.e., label 2). Since label 2 is most frequently observed in the citation datasets, this allows the model to improve the overall performance across three labels. In the two-class task, however, encoding more directional information enables the model to distinguish edges in a certain direction from all other possibilities (i.e., distinguish label 0 from label 1).

We notice that quasi-framelet, `Sigmoid`, is most helpful in link direction prediction tasks, especially for three-class tasks. This observation again proves the power of double regulation. Besides, in two-class tasks, Framelet-MagNet also prefer simpler graph framelets, `Haar` and `Linear`, for some datasets. The difference between simple and complex graph framelets is that simpler graph framelets are based on fewer scaling functions. Since the node attributes for link prediction tasks are just node in-degrees and node out-degrees, they are less complicated than those for node classification tasks. Accordingly, we may not need a complicated framelet transform to process graph signals in link tasks. Therefore, the optimal framelet types tend to be simpler ones.

4.5 Denoising

When we construct real-life data as graphs, each observation is modelled as a node, and its corresponding attributes are modelled as graph signals. While the underlying rationale of graph neural networks is to utilize the information contained in graph signals to make predictions, it is inevitable that some information is less useful or even harmful for the prediction. Such information is considered as "noises". For example, graph neural networks are sometimes vulnerable to adversarial examples, which can be regarded as "attackers intentionally designed

to cause the model to make mistakes" [63]. It is shown in [64] that even only a few perturbations in graph signals can cause a significant drop in the node classification accuracy of some state-of-the-art models. Intuitively, removing noises from graph signals or their representations will help models to make more accurate decisions. This procedure is known as "denoising".

In GCNNs, denoising is achieved by filtering the graph signals with learnable filters in the training process. In classic spectral GCNNs, the graph signals are converted to their spectral representations through Fourier transform before filtering. By contrast, in Framelet-MagNet, we obtain more sophisticated representations through Framelet transform. Theoretically, these representations will allow the model to better distinguish different frequency components of graph signals, enabling more effective denoising. In this experiment, we explore the denoising capability of Framelet-MagNet and two other baseline models over two benchmark datasets. We will manually include noises in the original graph signals and then test the performance of each model over the data with perturbation. The design of this experiment is based on [28] and [29].

4.5.1 Implementation Details

This experiment is based on the node classification task, but the same methods can be applied to other tasks. We will use two datasets for this experiment, including CORA_ML and TEXAS. For CORA_ML, magnetic Laplacian-based models tend to choose $q = 0.00$, implying no usage of directional information in prediction. On the contrary, magnetic Laplacian-based models prefer larger q values for TEXAS, incorporating more directional information for better classification outcomes. Thus, we choose these two datasets to investigate two circumstances: (1) directional information is regarded as useless; (2) directional information is important for predictions.

Different from other experiments, we will not split each dataset into 10 subsamples. Alternatively, we use the whole dataset and repeat the training, validation, and testing process 10 times. This will allow us to have more data for training, validation, and testing. Considering that we will introduce noises in graph signals, more training data will generate more promising results. Meanwhile, we can still evaluate the mean and standard deviation of 10 results from the 10 trials to offset the randomness inherent in neural networks. For CORA_ML, 20 random nodes per

class are used as training data with 500 nodes for validation and the rest for testing. For TEXAS, 10 random nodes per class are used as training data with 50 nodes for validation and the rest for testing.

We adopt two different methods to inject noises into the original graph signals. Generally, our choice of methods depends on the data type of input graph signals. CORA_ML has normalized input feature values, while TEXAS has binary input feature values. For normalized data, we add Gaussian distributed values with mean 0 and standard deviation σ to the original graph signals as noises. We will gradually raise the noise level by increasing the standard deviation σ. Specifically, we will implement $\sigma = 0.00, 0.01, 0.05, 0.10, 0.50, 1.00, 5.00, 10.00$. With a larger σ, the noises will fluctuate more dramatically, so the denoising task will be more challenging. For binary data, on the other hand, we add Bernoulli distributed noises. Namely, we randomly change feature values from 0/1 to its opposite 1/0. The noise level r in this case indicates the signal-to-noise ratio, implying the proportion of perturbed data. We set this ratio between 0.00 and 0.20, which means the proportion of altered data will be between 0% and 20% in the experiment. More specifically, we will have $r = 0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20$.

We will compare our model, Framelet-MagNet, with two baseline models, MagNet and GCN. We choose GCN because it is a typical spectral GCNN, which means we can treat it as a benchmark. Framelet-MagNet is composed of one convolutional layer followed by a fully connected linear layer. MagNet and GCN, however, need two convolutional layers to express their power. We include a dropout layer after the convolutional layers with a dropout rate of 0.5 to prevent overfitting. The output layer is a softmax layer for classification. For all three models, we implement the ReLU activation function, number of filters 64, and weight decay with $\lambda = 5e^{-4}$. Besides, we use the Adam optimizer with a learning rate of 0.01 and train each model with a maximum of 200 epochs. In each epoch, the model is kept if the validation accuracy increases. In terms of the adjacency matrix, we use the original matrix for magnetic Laplacian-based models and the symmetrized adjacency matrix for GCN.

For MagNet and GCN, we set $K = 1$. For MagNet and Framelet-MagNet, We tune the q value in the searching space $[0.00, 0.05, 0.15, 0.25]$. Finally, here are some Framelet-MagNet

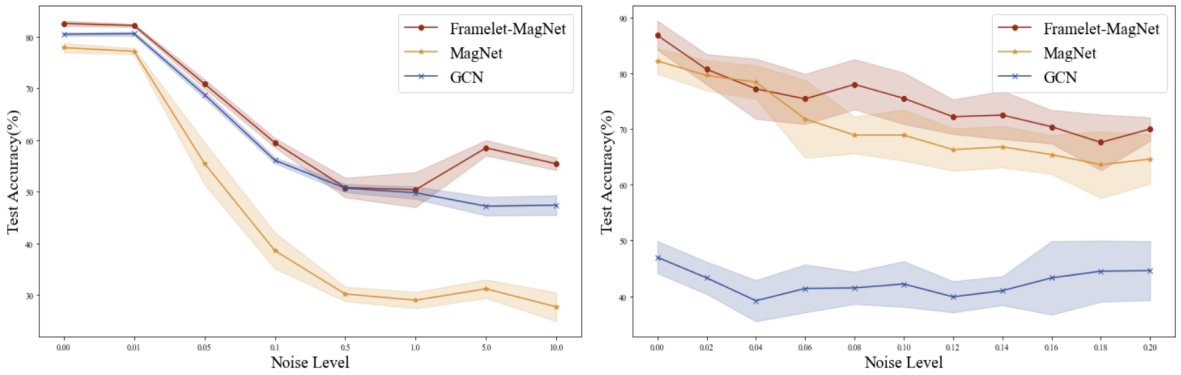

Figure 4.1: Denoising experiment results. For the denoising tasks on CORA_ML (left) and TEXAS (right), Framelet-MagNet achieves better classification accuracy than MagNet and GCN at almost all noise levels.

specific hyperparameter settings. The selection of framelet type is based on the choices in the node classification experiment in **Table 4.1**. For CORA_ML, we use `Sigmoid` quasi-framelet transform, with $\alpha = 20.0$ and a Chebyshev degree of 2. Then, for TEXAS, we adopt `Quadratic` framelet transform with the Chebyshev degree selected among 2, 3, and 4. The dilation level $s = 2$.

4.5.2 Experiment Results

We present the experiment results of the denoising tasks on CORA_ML and TEXAS respectively in **Table 4.6** and **Table 4.7**. We will focus on the average accuracy over 10 trials for evaluation. The model with a strong denoising capacity will achieve better accuracy than others at each noise level. In addition, we also plot the results in **Figure 4.1** to show the changes in prediction accuracy at different noise levels more clearly.

Table 4.6: Denoising Results of Framelet-MagNet and Baseline Models on CORA_ML

Noise Level	Framelet-MagNet	MagNet	GCN
$\sigma = 0.00$	**82.6 ± 0.5**	77.9 ± 0.9	80.5 ± 0.3
$\sigma = 0.01$	**82.2 ± 0.3**	77.2 ± 0.6	80.6 ± 0.4
$\sigma = 0.05$	**70.9 ± 0.9**	55.5 ± 4.1	68.8 ± 0.7
$\sigma = 0.10$	**59.5 ± 0.9**	38.6 ± 3.5	56.1 ± 0.6
$\sigma = 0.50$	**50.8 ± 1.9**	30.2 ± 1.4	50.7 ± 0.8
$\sigma = 1.00$	**50.4 ± 3.4**	29.0 ± 1.6	49.8 ± 1.2
$\sigma = 5.00$	**58.5 ± 1.5**	31.2 ± 1.8	47.2 ± 1.8
$\sigma = 10.00$	**55.4 ± 1.2**	27.7 ± 2.8	47.4 ± 1.9

Table 4.7: Denoising Results of Framelet-MagNet and Baseline Models on TEXAS

Noise Level	Framelet-MagNet	MagNet	GCN
$r = 0.00$	**86.8 ± 2.6**	82.2 ± 2.3	47.0 ± 2.9
$r = 0.02$	**83.8 ± 2.5**	79.6 ± 2.7	43.3 ± 2.9
$r = 0.04$	77.2 ± 5.4	**78.4 ± 3.0**	39.2 ± 3.7
$r = 0.06$	**75.4 ± 4.5**	71.8 ± 7.0	41.4 ± 4.3
$r = 0.08$	**78.0 ± 4.5**	68.9 ± 3.3	41.5 ± 2.9
$r = 0.10$	**75.5 ± 4.6**	68.9 ± 4.6	42.2 ± 4.1
$r = 0.12$	**72.2 ± 3.1**	66.3 ± 3.8	39.9 ± 2.8
$r = 0.14$	**72.5 ± 4.3**	66.8 ± 3.7	41.0 ± 2.6
$r = 0.16$	**70.4 ± 3.0**	65.4 ± 3.5	43.3 ± 6.6
$r = 0.18$	**67.6 ± 5.0**	63.6 ± 6.0	44.5 ± 5.5
$r = 0.20$	**70.0 ± 2.1**	64.6 ± 4.4	44.6 ± 5.3

Table 4.6 shows the denoising capability of Framelet-MagNet, MagNet, and GCN on CORA_ML. With the original graph signals (i.e., $\sigma = 0.00$), the average accuracy given by Framelet-MagNet, MagNet and GCN are 82.6%, 77.9%, and 80.5%. At this stage, the difference in the results is not very significant. Nonetheless, we notice that their performances vary more distinctively when we raise the noise level. When $\sigma = 10.00$, the accuracy of Framelet-MagNet is 8% and 27.7% higher than the accuracy of GCN and MagNet, respectively. Notably, the accuracy of Framelet-MagNet is even twice as much as MagNet. This indicates that Framelet-MagNet is good at handling severely distorted data, showing a strong denoising capability.

Table 4.7 shows the denoising capability of Framelet-MagNet, MagNet, and GCN over TEXAS. Generally, Framelet-MagNet achieves better accuracy than the other two models at nearly all noise levels, demonstrating its superior skill in denoising. Compared with MagNet, Framelet-MagNet's accuracy is 4.6% higher when no noise is introduced. This difference increases to 6.6% when $r = 0.10$, and 5.4% when $r = 0.20$. The accuracy of Framelet-MagNet is only lower than MagNet when $r = 0.04$, but with a small difference, 1.2%. Therefore, the incorporation of framelet transform enhances the overall denoising capability of MagNet. Unlike Framelet-MagNet and MagNet, GCN does not show a successive decrease in prediction accuracy (see **Figure 4.1**). Nevertheless, its accuracy is under 50% at all noise levels, showing a very poor classification ability. Since magnetic Laplacian-based models typically choose $q > 0$ for TEXAS, directional information is regarded as valuable in this classification task. However, GCN adopts symmetrized adjacency matrix, which means directional information is discarded.

Chapter 5

Conclusion

In this Chapter, we will summarize this thesis and highlight the contributions of our research to the discipline and organizational practices. In addition, we will list current limitations and potential future works. Moreover, we will provide a statement of ethical consideration at the end of this Chapter.

5.1 Summary and Contributions

In this thesis, we propose Framelet-MagNet, a magnetic framelet-based spectral GCNN for digraphs, and demonstrate its power over state-of-the-art methods via empirical results. Our work is motivated by the fact that although framelet-based spectral GCNNs have demonstrated their power in undirected graph applications, there is hardly any such GCNN for digraphs. So, we extend the method to digraphs with the assistance of the magnetic Laplacian. We realize magnetic Laplacian-based tight framelet transform on digraphs. In addition, we use Chebyshev polynomial approximation to design a fast computation for magnetic framelet transform and reconstruction. Besides, we use FMFT, the fast magnetic framelet transform, to define a magnetic framelet-based convolutional layer, in which we process digraph signals in a complex frequency domain to achieve effective filtering. Finally, we assemble Framelet-MagNet with the newly defined convolutional layer. Based on the comparison between Framelet-MagNet and 10 baseline models in three experiments, we conclude that Framelet-MagNet successfully improves the state-of-the-

art performance of spectral GCNN for digraphs. More specifically, Framelet-MagNet shows its superiority over various advanced baseline models in node classification, link prediction, and denoising tasks on several benchmark datasets. Compared with MagNet, a magnetic Laplacian-based spectral GCNN whose convolutional layer is assisted by the Fourier transform, Framelet-MagNet has higher prediction accuracy, better robustness, and more excellent denoising capability. This demonstrates the power of the framelet transform over the traditional Fourier transform in the signal processing of digraph GCNNs.

The contributions of our research to the discipline and organizational practices are threefold. Firstly, to our best knowledge, this research is the first attempt to construct a framelet-based digraph GCNN without discarding the role of Laplacian eigendecomposition. So, it is an inspiration for framelet-based digraph neural networks in relevant studies. Next, our work realizes graph framelet transform with a complex-valued Laplacian, which means we can process graph signals in both real and complex domains simultaneously. This opens up a new perspective for framelet-based signal processing in GCNNs. Furthermore, our approach is an effective solution to enhance the prediction accuracy and stability of digraph GCNNs. This strengthens the practical value of neural networks in real-world digraph applications.

5.2 Limitations and Future Works

5.2.1 Limitations in Link Prediction Tasks and Future Works

The design link prediction tasks impose some constraints on link type and graph data, resulting in restricted real-world application values. When assigning labels to graph edges, we did not clearly define undirected edges and self-loops. This is not a huge concern since the majority of edges in our benchmark datasets are directed edges. Nevertheless, given the fact that these two types of edges are frequently observed in real-world datasets, future research shall consider how to include them in prediction. In addition, we used node in-degrees and out-degrees as node features to train our model. Although this allows the model to exploit graph topological information for prediction, it fails to explore the value of original node attributes, which may

also be useful information when determining edge types.

Including self-loops in prediction is not difficult. Recall that we use ordered node pairs to identify edges in digraphs. To apply this methodology to identify self-loops, we may add node pairs with the same node when defining edge labels. For instance, suppose that $v_i \in \mathcal{V}$, then we can use a node pair v_i, v_i to detect whether self-loop $(v_i, v_i) \in \mathcal{E}$. However, there is no good strategy for defining undirected edges currently. Although the ordered node pairs v_i, v_j and v_j, v_i can define an undirected edge between node v_i and v_j collaboratively, the ideal design should be a unique label for undirected edges, which shall be explored in future works.

In terms of node attributes, we may conduct more link-level experiments with original features. Besides, stacking node degree features with the original features may also be a workable solution for more accurate predictions.

5.2.2 Limitations in Framelet-MagNet and Future Works

Although we apply Chebyshev polynomials to accelerate computation, Framelet-MagNet is still slow with large graphs. It is suggested by [21] that we may consider incorporating attention mechanisms for large graphs.

Another concern is relevant to mixed graphs. In Chapter 3, we constructed the magnetic Laplacian with a symmetric adjacency matrix and a skew-symmetric adjacency matrix. According to [65, 66, 67], this skew-symmetric matrix is designed for digraphs with only directed edges (i.e., oriented graphs). So, to identify an undirected edge, we have to combine the information in the symmetric adjacency matrix. What if we prefer to construct a complex-valued Laplacian that represents mixed graphs more straightforwardly?

To address this problem, we propose Hermitian Laplacian-based digraph GCNNs. For a mixed digraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, let node $v_i, v_j \in \mathcal{V}$. Then, the Hermitian adjacency matrix [68, 69] is defined as

$$\mathbf{A}_H(i,j) = \begin{cases} 1, & \text{if} \quad v_i \leftrightarrow v_j \\ \mathrm{i}, & \text{if} \quad v_i \rightarrow v_j \\ -\mathrm{i}, & \text{if} \quad v_i \leftarrow v_j \\ 0, & \text{otherwise} \end{cases} . \tag{5.1}$$

Based on this Hermitian adjacency matrix, [70] constructed Hermitian Laplacian. Let $\mathbf{A}$ be the original adjacency matrix. We start with symmetrizing $\mathbf{A}$ by replacing directed edges with undirected edges to obtain $\mathbf{A}_s$. Then, the degree matrix $\mathbf{D}_s$ is computed based on $\mathbf{A}_s$. Finally, the Hermitian Laplacian is defined as

$$\mathcal{L}_H = \mathbf{D}_s^{-1/2}(\mathbf{D}_s - \mathbf{A}_H)\mathbf{D}_s^{-1/2} = \mathbf{I} - \mathbf{D}_s^{-1/2}\mathbf{A}_H\mathbf{D}_s^{-1/2} \tag{5.2}$$

It has been proved that Hermitian Laplacian is symmetric and positive semi-definite [70], which means it can be applied to GCNN architectures in a manner analogous to the magnetic Laplacian. In addition, since Hermitian Laplacian and magnetic Laplacian share many properties, we suggest that it is possible to extend framelets to Hermitian Laplacian-based neural networks.

5.3 Ethical Considerations

Almost no ethical issues are anticipated from our research. In our experiments, we only use publicly available datasets from secondary sources, and data authority is not relevant. All datasets contain no personally identifiable or harmful information. Moreover, our research methods are commonly used in machine learning research. They have neither more, nor less social influence compared with other machine learning research methods.

References

[1] F. Monti, K. Otness, and M. M. Bronstein, "Motifnet: a motif-based graph convolutional network for directed graphs," in *2018 IEEE Data Science Workshop*. IEEE, 2018, pp. 225–228.

[2] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[3] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[4] A. Daigavane, B. Ravindran, and G. Aggarwal, "Understanding convolutions on graphs," *Distill*, 2021.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[8] B. Abedin and B. Sohrabi, "Graph theory application and web page ranking for website link structure improvement," *Behaviour & Information Technology*, vol. 28, no. 1, pp. 63–72, 2009.

[9] B. Zhou, J. Chen, J. Shi, H. Zhang, and Q. Wu, "Website link structure evaluation and improvement based on user visiting patterns," in *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*, 2001, pp. 241–244.

[10] J. R. Abrams, J. Celaya-Alcalá, D. Baldwin, R. Gonda, and Z. Chen, "Analysis of equity markets: A graph theory approach," *Society for Industrial and Applied Mathematics. doi*, vol. 10, 2016.

[11] Y. An, J. Janssen, and E. E. Milios, "Characterizing and mining the citation graph of the computer science literature," *Knowledge and Information Systems*, vol. 6, no. 6, pp. 664–678, 2004.

[12] M. Boss, H. Elsinger, M. Summer, and S. Thurner 4, "Network topology of the interbank market," *Quantitative Finance*, vol. 4, no. 6, pp. 677–684, 2004.

[13] F. Zhou, Q. Yang, T. Zhong, D. Chen, and N. Zhang, "Variational graph neural networks for road traffic prediction in intelligent transportation systems," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 2802–2812, 2021.

[14] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[15] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li, "Understanding graph sampling algorithms for social network analysis," *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 123–128, 2011.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations*, 2018.

[18] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural

networks meet personalized pagerank," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[20] Z. Tong, Y. Liang, C. Sun, X. Li, D. S. Rosenblum, and A. Lim, "Digraph inception convolutional networks," in *NeurIPS*, 2020.

[21] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. J. Hirn, "Magnet: A neural network for directed graphs," in *NeurIPS*, 2021.

[22] Y. Ma, J. Hao, Y. Yang, H. Li, J. Jin, and G. Chen, "Spectral-based graph convolutional network for directed graphs," *arXiv preprint arXiv:1907.08990*, 2019.

[23] M. Fanuel, C. M. Alaiz, and J. A. Suykens, "Magnetic eigenmaps for community detection in directed networks," *Physical Review E*, vol. 95, no. 2, p. 022302, 2017.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[25] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[26] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," *arXiv preprint arXiv:1904.07785*, 2019.

[27] M. Yang, X. Zheng, J. Yin, and J. Gao, "Quasi-framelets: Another improvement to graphneural networks," *arXiv preprint arXiv:2201.04728*, 2022.

[28] X. Zheng, B. Zhou, J. Gao, Y. Wang, P. Lió, M. Li, and G. Montúfar, "How framelets enhance graph neural networks," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, vol. 139, 2021, pp. 12 761–12 771.

[29] C. Zou, A. Han, L. Lin, and J. Gao, "A simple yet effective svd-gcn for directed graphs," *arXiv preprint arXiv:2205.09335*, 2022.

[30] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[31] S. Ebli, M. Defferrard, and G. Spreemann, "Simplicial neural networks," *arXiv preprint arXiv:2010.03633*, 2020.

[32] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[33] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[34] C. Li, X. Qin, X. Xu, D. Yang, and G. Wei, "Scalable graph convolutional networks with fast localized spectral filter for directed graphs," *IEEE Access*, vol. 8, pp. 105 634–105 644, 2020.

[35] H. Minc, *Nonnegative matrices*. Wiley, 1988.

[36] Z. Tong, Y. Liang, C. Sun, D. S. Rosenblum, and A. Lim, "Directed graph convolutional network," *arXiv preprint arXiv:2004.13970*, 2020.

[37] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[38] E. H. Lieb and M. Loss, "Fluxes, laplacians, and kasteleyn's theorem," in *Statistical Mechanics*. Springer, 1993, pp. 457–483.

[39] A. Ron and Z. Shen, "Affine systems in $\mathbf{L}_2(\mathbb{R}^d)$: the analysis of the analysis operator," *Journal of Functional Analysis*, vol. 148, no. 2, pp. 408–447, 1997.

[40] C. K. Chui, W. He, and J. Stöckler, "Compactly supported tight and sibling frames with maximum vanishing moments," *Applied and Computational Harmonic Analysis*, vol. 13, no. 3, pp. 224–262, 2002.

[41] B. Han, "Framelets and wavelets," *Algorithms, Analysis, and Applications, Applied and Numerical Harmonic Analysis*, 2017.

[42] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 53–94, 2006.

[43] Y. G. Wang and X. Zhuang, "Tight framelets and fast framelet filter bank transforms on manifolds," *Applied and Computational Harmonic Analysis*, vol. 48, no. 1, pp. 64–95, 2020.

[44] X. Zheng, B. Zhou, Y. G. Wang, and X. Zhuang, "Decimated framelet system on graphs and fast g-framelet transforms." *Journal of Machine Learning Research*, vol. 23, pp. 18–1, 2022.

[45] M. Crovella and E. Kolaczyk, "Graph wavelets for spatial traffic analysis," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 3. IEEE, 2003, pp. 1848–1857.

[46] M. Gavish, B. Nadler, and R. R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," in *ICML*, 2010.

[47] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[48] B. Dong, "Sparse representation on graphs by tight wavelet frames and applications," *Applied and Computational Harmonic Analysis*, vol. 42, no. 3, pp. 452–479, 2017.

[49] H. Gao and S. Ji, "Graph u-nets," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2083–2092.

[50] R.-Q. Jia and Z. Shen, "Multiresolution and wavelets," *Proceedings of the Edinburgh Mathematical Society*, vol. 37, no. 2, pp. 271–300, 1994.

[51] S. G. Mallat, "Multiresolution approximations and wavelet orthonormal bases of $l^2(\mathbb{R}$," *Transactions of the American Mathematical Society*, vol. 315, no. 1, pp. 69–87, 1989.

[52] C. de Boor, R. A. DeVore, and A. Ron, "On the construction of multivariate (pre) wavelets," *Constructive approximation*, vol. 9, no. 2, pp. 123–166, 1993.

[53] M. Fanuel, C. M. Alaíz, Á. Fernández, and J. A. Suykens, "Magnetic eigenmaps for the visualization of directed networks," *Applied and Computational Harmonic Analysis*, vol. 44, no. 1, pp. 189–199, 2018.

[54] M. Shubin, "Discrete magnetic laplacian," *Communications in Mathematical Physics*, vol. 164, no. 2, pp. 259–275, 1994.

[55] G. Berkolaiko, "Nodal count of graph eigenfunctions via magnetic perturbation," *Analysis & PDE*, vol. 6, no. 5, pp. 1213–1233, 2013.

[56] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 77–84, 1975.

[57] J. Zhang, B. Hui, P.-W. Harn, M.-T. Sun, and W.-S. Ku, "smgc: A complex-valued graph convolutional network via magnetic laplacian for directed graphs," *arXiv preprint arXiv:2110.07570*, 2021.

[58] A. Bojchevski and S. Günnemann, "Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, 2018.

[59] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[60] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[61] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, 2021.

[62] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web - 15th International Conference, ESWC 2018*, ser. Lecture Notes in Computer Science, vol. 10843, 2018, pp. 593–607.

[63] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020.

[64] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.

[65] M. Cavers, S. Cioabă, S. Fallat, D. Gregory, W. Haemers, S. Kirkland, J. McDonald, and M. Tsatsomeros, "Skew-adjacency matrices of graphs," *Linear Algebra and its Applications*, vol. 436, no. 12, pp. 4512–4529, 2012.

[66] S. Li and Y. Yu, "Hermitian adjacency matrix of the second kind for mixed graphs," *Discrete Mathematics*, vol. 345, no. 5, p. 112798, 2022.

[67] B. Mohar, "A new kind of hermitian matrices for digraphs," *Linear Algebra and its Applications*, vol. 584, pp. 343–352, 2020.

[68] J. Liu and X. Li, "Hermitian-adjacency matrices and hermitian energies of mixed graphs," *Linear Algebra and its Applications*, vol. 466, pp. 182–207, 2015.

[69] K. Guo and B. Mohar, "Hermitian adjacency matrix of digraphs and mixed graphs," *Journal of Graph Theory*, vol. 85, no. 1, pp. 217–248, 2017.

[70] G. Yu and H. Qu, "Hermitian laplacian matrix and positive of mixed graphs," *Applied Mathematics and Computation*, vol. 269, pp. 70–76, 2015.