이학박사 학위논문

Spectral Analysis of Graph Neural Networks and Its Applications

그래프 신경망의 스펙트럴 해석과 그 응용

2023년 8월

서울대학교 대학원

수리과학부

조 현 수

Spectral Analysis of Graph Neural Networks and Its Applications

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2023년 6월

서울대학교 대학원

수리과학부

조 현 수

조 현 수의 이학박사 학위논문을 인준함

2023년 6월

위 원 장 _____ **(인)**

부 위 원 장 _____ **(인)**

위 원 _____ **(인)**

위 원 _____ **(인)**

위 원 _____ **(인)**

Spectral Analysis of Graph Neural Networks and Its Applications

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Hyunsoo Cho

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

June 2023

Abstract

Spectral Analysis of Graph Neural Networks and Its Applications

Hyunsoo Cho
Department of Mathematical Sciences
The Graduate School
Seoul National University

In this dissertation, we present a theoretical analysis of spectral-based graph neural networks and their practical performance. We analyze how the spectra of a graph Laplacian relates to the convolution operation of a graph neural network, and we discuss how expressive a graph convolutional model can be and how competent expressiveness can be achieved by implementing various convolutions on a graph based on this spectra. The results show that spectral-based graph neural networks can perform well on graph-based tasks, and we discuss what improvements can be made in the future to improve their performance in practice. As an extension, we apply it to traditional computer vision tasks in addition to graph-based tasks and show that it is comparably expressive.

In addition, we present several results of its applications utilizing graphs. Specifically, we conducted experiments on the task of salient object detection using directed acyclic graphs. We also show experimental results of applying the simple model based on the theory of Fourier analysis to practical applications such as the rain removal task. These experiments empirically demonstrate that incorporating the knowledge of graph theory and Fourier analysis into the model helps improve performance.

keywords : Spectral-based Graph Neural Network, Graph Neural Networks, Collaborative Filtering, Salient Object Detection, Deraining
Student Number : 2016–25263

Contents

Chapter 1

Introduction

In recent years, there has been a growing interest in developing advanced techniques for analyzing and processing graph-structured data. And so Graph Neural Networks (GNNs) have emerged as powerful models ([1], [2], [3], [4], [5], [6]) capable of capturing complex relationships and patterns in graph data.

Graphs are a type of data structure that represents a collection of interconnected nodes or vertices. These nodes are connected by edges, which can have different properties such as direction, weight, or label. Graphs are used to model relationships between entities and solve various real-world problems. For instance, there are some common graph data structures such as undirected graphs, directed graphs, weighted graphs, trees, and bipartite graphs.

In an undirected graph, edges have no direction. They represent symmetric relationships between nodes. Social Networks, Web Link Analysis, and Clustering are examples of where these undirected graphs are used.

In a directed graph, edges have a direction from one node to another. They represent asymmetric relationships. These directed graphs are used as the main objects in Web Page Ranking, Routing Algorithms, and Workflow Management. Google's PageRank [7] is the most famous example of an algorithm that builds an directed graph of links between websites.

We can also assign weights to the edges of the graph, representing the

strength or cost of a relationship. There are many applications of weighted graphs, especially in computer algorithms such as Shortest Path Algorithms and Minimum Spanning Tree Algorithms.

Trees are a special type of graph where each node has a specific parent-child relationship. In computer sciences, trees are used to represent hierarchical structures of files and directories and construct syntax trees to analyze and interpret programming languages for parsing and compiler design. Moreover, in the field of deep learning, the tree structure is also used in the decision tree model used by XGBoost [8] and LightGBM [9].

A bipartite graph is a type of graph whose vertex set can be partitioned into two disjoint sets. This graph is commonly used in recommendation systems, where it can be applied especially in online commercial services. In these systems, bipartite graphs can represent relationships between users and items helping to generate personalized recommendations based on connections between users and items. Therefore, methods such as collaborative filtering in recommendation systems use these bipartite graphs to develop models [10].

In addition to the above versatility of graph data, the recent success of large language models such as GPT-4 [11] has brought more focus to the potential of graphs. The basic module of a transformer [12] is a *Scaled Dot-Product Attention* of query Q, key K, and values V, and the formula of this module is as follows, where $Q, K, V \in \mathbb{R}^{n \times d}$

$$Attention(Q, K, V) = softmax\Big(\frac{QK^\top}{\sqrt{d}}\Big)V \tag{1.1}$$

Here, the n by n matrix $A := softmax\Big(\frac{QK^\top}{\sqrt{d}}\Big)$ can be understood as a weighted adjacent matrix of some graph, and then A is multiplied by V, which is the message passing process in a typical Graph Neural Network architecture. This messaging passing process will be briefly discussed in 2.1.2.

In fact, in recent papers [13], methods have been developed that go beyond applying a transformer by splitting the image into patches in a Vision Transformer [14] and apply the idea of a graph neural network. Figure 1.1 is an illustrative example of applying these methods to an Image to create a graph structure.

(a) a sample figure (b) grid structure (c) graph structure

Figure 1.1: Images and its graph structure (Han, Kai, et al. [13])

In this context of the importance of these graphs, this dissertation aims to analyze the currently used Graph Neural Networks from the perspective of spectral analysis with mathematical theories. Also, we will discuss the improvement methods of the model based on these theories.

Specifically, this dissertation begins in **Chapter 2** with an explanation of the basic terminology and concepts needed to analyze GNNs. It also includes an introduction to Collaborative Filtering and Directed Acyclic Graphs Leaning. **Chapter 3** introduces related works on the topics covered in this dissertation, which these works include spectral-based GNN models like Spectral Network, ChebNet, and Graph Convolutional Networks.

Chapter 4 is devoted to the analysis of graph neural networks in terms of spectral analysis. Starting with the question of how to define the convolution operation for general graphs, we observe how convolution is defined in a typical Schwartz space or ring graphs and what relations the convolution operator has in terms of the Laplacian and Fourier transform. Based on these observations, we propose the plausible assumptions for defining convolution in general graphs and present a theorem that an arbitrary convolution matrix can be represented exactly under these assumptions.

Chapter 5 and 6 propose models for solving practical problems such as collaborative filtering, salient object detection and rain removal tasks based on spectral-based GNNs or Fourier analysis and present their results. Finally, **Chapter 7** concludes with concluding remarks.

Chapter 2

Preliminaries

2.1 Graph Neural Networks

In this section, we briefly review the two typical types of graph neural networks. One is the Spatial-based graph neural network and the other is the Spectral-based graph neural network. First, we'll cover some mathematical terminology and definitions to get started with these methods.

2.1.1 Mathematical Terminologies

First of all, $|A|$ represents the cardinality of the set A.

Definition 2.1.1. For a finite V, $\mathcal{G} = (V, E)$ is called a directed graph if E is a subset of $V \times V$. We call V the set of vertices (or nodes) and E the set of edges (or relations), respectively.

Definition 2.1.2. A graph $\mathcal{G} = (V, E)$ is undirected if every $(x, y) \in E$ implies $(y, x) \in E$.

Definition 2.1.3. For a graph $\mathcal{G} = (V, E)$ on $N = |V|$ for some $N \in \mathbb{N}$, the

adjacency matrix $A = A(\mathcal{G})$ in $\mathbb{R}^{N\times N}$ is defined by

$$A = (a_{ij}), \quad \text{where} \quad a_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1 \le i, j \le N). \tag{2.1}$$

Definition 2.1.4. For a graph $\mathcal{G} = (V, E), A = A(\mathcal{G}) = (a_{ij})$, a degree matrix D of $\mathcal{G}$ is a $|V| \times |V|$ diagonal matrix defined as

$$D = (d_{i,j}), \quad \text{where} \quad d_{ij} = \begin{cases} \sum_j a_{ij} & i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Definition 2.1.5. A graph $\mathcal{G} = (V, E)$ is a bipartite graph if two disjoint subsets V_1 and V_2 of V which partition V such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ and there only exist edges between V_1 and V_2. In case of a bipartite graph, we also denote $\mathcal{G}$ as (V_1, V_2, E).

From these definitions, we can readily know that the adjacency matrix the adjacency matrix of a (undirected) bipartite graph $\mathcal{G} = (V_1, V_2, E)$ is

$$A(\mathcal{G}) = \begin{pmatrix} 0 & B \\ B^t & 0 \end{pmatrix} \in \mathbb{R}^{(|V_1|+|V_2|)\times(|V_1|+|V_2|)} \tag{2.3}$$

for some $B \in \mathbb{R}^{|V_1|\times|V_2|}$ and elements of B only consist of 0 or 1.

2.1.2 Graph Message Passing

There are several approaches to the basic Graph Neural Network(GNN) model [15, 16, 17]. They commonly used the form of neural message passing, which aggregates messages from neighbor nodes and updates their embeddings through neural networks. From now on, we will focus on how to generate the message and update the kth hidden embedding $E^{(k)}$.

The basic message passing framework in the node level can be expressed as follows

$$e_u^{(k)} = UPDATE^{(k)}(e_u^{(k-1)}, AGGREGATE^{(k)}(\{e_v^{(k-1)}, \forall v \in \mathcal{N}_u\})) \tag{2.4}$$

where $e_u^{(0)}$ is an initial embedding of node u and $e_u^{(k)}$ is the hidden embedding

of node u generated from the kth iteration [18]. AGGREGATE function generates a message from hidden embeddings of adjacent nodes, and UPDATE function generates the next hidden embedding $e_u^{(k)}$ from the message and previous hidden embedding $e_u^{(k-1)}$.

Merkwirth *et al.* [19] and Scarselli *et al.* [20] proposed a basic GNN model that uses aggregate function and update function as follows

$$e_u^{(k)} = \sigma(W_{self}^{(k)} e_u^{(k-1)} + W_{neigh}^{k} \sum_{v \in \mathcal{N}_u} e_v^{k-1} + b^{(k)}) \tag{2.5}$$

where $W_{self}^{(k)}, W_{neigh}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ and $b \in \mathbb{R}^{d^{(k)}}$ are trainable parameters of kth layer of GNN, and σ is non-linear activation function. The aggregate function of this model generates message from neighbors simply by adding hidden embeddings of neighboring nodes, and the update function calculates message from neighbors and their previous hidden embeddings using linear combination, and finally applies elementwise non-linearity.

However, these methods are not the only AGGREGATE and UPDATE methods that exist. In Graph Neural Networks literature, these methods can be roughly categorized based on two message passing paradigms: Spatial-based GNNs and Spectral-based GNNs.

2.1.3 Spatial-based Graph Neural Networks

Spatial-based Graph Neural Networks aggregate a node's neighbors, usually by a fixed size, when passing messages. This fixed-size aggregation of neighbors makes it possible to apply traditional 1d convolution (1D-CNN) in GNN models such as Patchy-SAN [21] and LGCN [22]. Or, in the case of GraphSAGE [23], they tried to randomly select a fixed size of neighbors and use Max pooling, Mean Pooling, LSTM aggregator. However, since LSTM aggregator takes sequential input, it introduces the risk of breaking the model's permutation invariance.

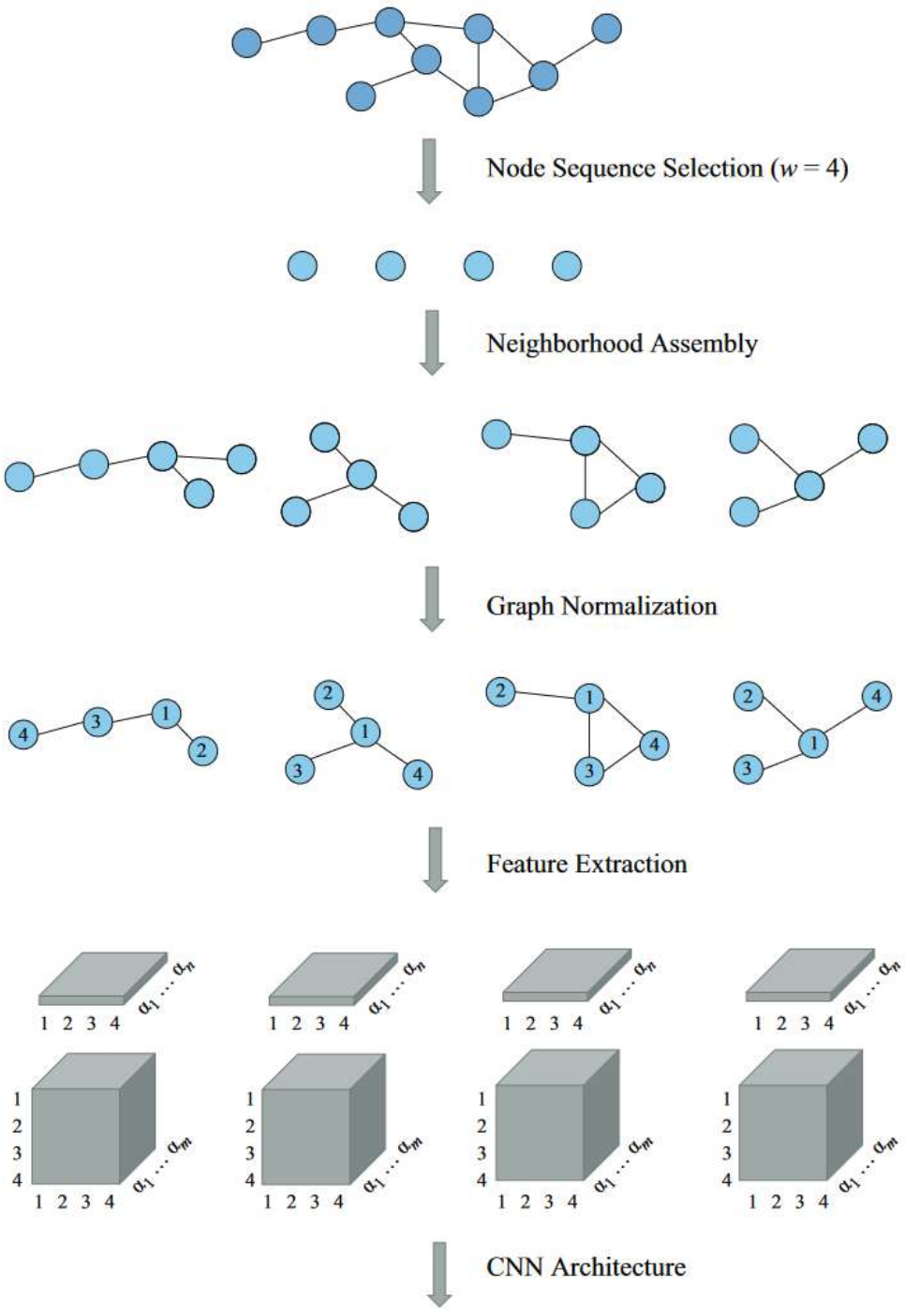


Figure 2.1: Spatial methods (Patch-SAN, Liu et al., 2020, p.27 [24])

2.1.4 Spectral-based Graph Neural Networks

Unlike Spatial-based GNNs, Spectral-based GNNs ([1], [25], [3]) utilize all neighbors of each node for message passing. In addition, Spatial-based applies a general 1D-CNN or 2D-CNN with a fixed number of neighbors, and these convolutional operations are not particularly considered to reflect the structure of the graph. Spectral-based GNNs, on the other hand, can define a *convolution on the graph* by utilizing the structure of the graph $\mathcal{G} = (V, E)$, i.e., the information of vertices V and edges E. The theoretical analysis is further elaborated in Section 4.

2.2 Collaborative Filtering

Figure 2.2: User-Item interaction

With the prevalence of the Internet and the abundance of information available, there is a growing demand for recommendation systems that can provide personalized item suggestions to individuals. As a result, extensive research is being conducted to enhance recommendation systems from various angles, and two prominent approaches are content-based filtering (CBF) and collaborative filtering (CF).

Content-based filtering (CBF) is an algorithm that examines the characteristics of items that users have positively engaged with and suggests items with similar attributes. For instance, it suggests another thriller genre movie

to users who have rated various thriller movies highly. CBF offers the benefit of generating recommendations based solely on the purchase history of a single user and the feature data of items. However, a drawback is that it does not provide recommendations for items lacking a history of interaction based on their features.

In contrast to CBF, the collaborative filtering (CF) is a technique used in recommendation systems to generate personalized recommendations by leveraging the preferences and behaviors of a group of users. It predicts a user's interests based on the interests and behaviors of similar users or items. CF does not rely on explicit item features but instead focuses on the collective wisdom of users.

Because of these properties, CF can provide recommendations for items that users might not have discovered on their own but are likely to enjoy based on the preferences of similar users. In other words, CF can provide diverse recommendations by considering the preferences and behaviors of a group of users. It can suggest items that might not be directly related but are liked by users with similar tastes.

The best advantage of CF is that it can handle the *"cold start"* problem, where there is a lack of initial data or information about a user or item. By leveraging similarities between users or items, CF can still make recommendations even for new or less-known entities.

Within the realm of CF models, matrix factorization techniques are commonly employed [26]. These methods represent collaborative signals or user-item interactions through linear combinations, which can limit their ability to capture complex structures. To tackle this challenge, neural collaborative filtering models [27] have emerged as an alternative. These models replace the inner product interaction function of matrix factorization with nonlinear neural networks, enabling them to capture more intricate patterns and relationships in the data.

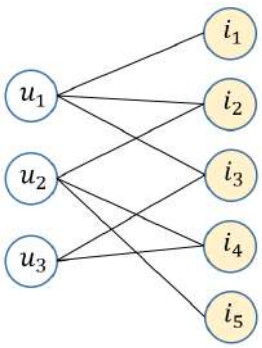


Figure 2.3: User-Item interaction bipartite graph

However, user-item interactions can be easily understood as edges of a

bipartite graph as shown in Figure 2.3. Therefore, it is a reasonable approach to introduce graph neural networks [28] in this CF models.

2.3 Directed Acyclic Graphs Learning

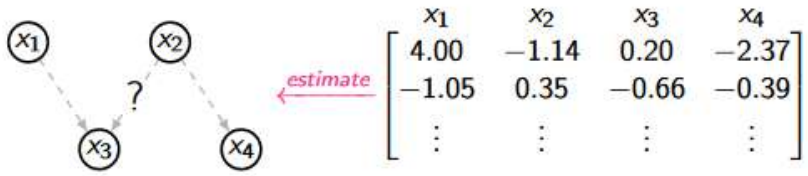


Figure 2.4: An illustration of DAG Structure Learning

Structure learning, in the context of machine learning and graphical models, refers to the process of inferring the underlying structure or topology of a graphical model from observed data. It involves determining the dependencies or relationships between variables in the data and constructing an appropriate graphical model that represents these relationships.

For example, in a linear structural equation model (linear SEM), we want to learn a directed acyclic graph (DAG) A that satisfies the following equation. Learning such an A is called DAG structure learning.

$$X = XA + Z \quad \text{(Linear SEM)} \tag{2.6}$$

, where $X \in \mathbb{R}^{n \times d}$ is a data matrix consisting of n i.i.d. observations and $Z \in \mathbb{R}^{n \times d}$ is n i.i.d. random Gaussian noise vectors.

Directed Acyclic Graphs (DAGs) are a type of graphical model commonly used in structure learning. In a DAG, nodes represent variables or random variables, and directed edges between nodes indicate direct dependencies or causal relationships between variables. The acyclic property ensures that there are no cycles or feedback loops in the graph, meaning the dependencies form a directed acyclic relationship.

The goal of structure learning with DAGs is to discover the most likely or optimal graph structure that best represents the relationships in the given

data. This typically involves finding the set of directed edges that best explain the dependencies among variables. There are several approaches and algorithms for structure learning in DAGs: [29], [30].

The basic approach to learning these DAGs is to solve the following constrained optimization problem, which is also known as NP-hard problem [31].

$$\begin{aligned} &\text{Minimize:} \quad F(A, X) \\ &\text{Subject to:} \\ &A \in \text{DAGs} \end{aligned}$$

In order to smoothly relax the above constraint that the matrix A must be a DAG in the previous work such as [29], [32] they introduce the following function h, which has the property that $h(A) = 0$ is equivalent to A being a DAG.

$$h(A) = tr[(I + \alpha A \odot A)^d] - d \quad (A \in \mathbb{R}^{d \times d},\ \alpha > 0) \tag{2.7}$$

, where $\odot$ means element-wise multiplication.

The above optimization problem is then rephrased as minimizing the regularized loss $F(A, X) + \lambda h(A)$ with respect to A. Here, λ is a hyperparameter for regularization. As a result, this continuous relaxation allows us to circumvent the combinatorial optimization problem and solve it with backpropagation, which can be utilized in deep learning frameworks.

In summary, structure learning with Directed Acyclic Graphs (DAGs) involves inferring the dependencies and relationships between variables from observed data. It plays a vital role in various domains by providing insights into causal relationships, enabling predictive modeling, and aiding decision-making in complex systems.

Chapter 3

Related Works

3.1 Spectral-based Graph Neural Networks

In historical order, the following three models are the most popular in spectral-based GNNs: Spectral Network [1], ChebNet [2], GCN [3].

3.1.1 Spectral Network

Bruna et al. [1] propose the following graph convolution.

$$g_\theta * x = U g_\theta(\Lambda) U^\top x, \tag{3.1}$$

where U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, with Λ of a diagonal matrix of eigenvalues of L.

3.1.2 ChebNet

Defferrard et al. [2] suggest that the graph convolution $g_\theta * x$ can be approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ up to K-th order. In other words,

$$g_\theta * x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L})x, \tag{3.2}$$

with $\tilde{L} = \frac{2}{\lambda_{max}} L - I_N$. λ_{max} denotes the largest eigenvalue of L. The Chebyshev polynomials are defined as

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \tag{3.3}$$

with $T_0(x) = 1$ and $T_1(x) = x$. The main difference with the above Spectral Networks is that ChebNet does not need to multiplicate with the Fourier basis U. This computation reduction is valid when L is sparse, specifically $\mathcal{O}(K|E|) \ll \mathcal{O}(|V|^2)$

3.1.3 Graph Convolutional Networks

Kipf and Welling [3] proposed Graph Convolutional Network(GCN) model which motivated from a first-order approximation of spectral graph convolutions [33]. Indeed, it is a simplified version of ChebNet. Layer-wise propagation rule fo GCN can be expressed as follows

$$E^{(k)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}E^{(k-1)}W^{(k)}) \tag{3.4}$$

where $\tilde{A} = A + I$ is the adjacency matrix of the undirected graph with self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, and $W^{(k)}$ is a trainable weight matrix of the kth layer of GCN. The message is generated by *weighted sum* of the hidden embeddings of neighbors and center node.

3.2 Collaborative Filtering

Collaborative filtering in recommendation systems can be approached in a variety of ways, from the very traditional Matrix Factorization based on Singular Value Decomposition [34]. However, with the recent development of GNNs, collaborative filtering models using graph neural networks have been

actively developed. Among them, NGCF [5] and LightGCN [6] are the major ones.

Alternating Least Squares

$$\min_{X,Y} \sum_{r_{ui}:observed} (r_{ui} - x_u^\top y_i)^2 + \lambda(\sum_u ||x_u||^2 + \sum_i ||y_i||^2) =: L(X,Y) \quad (3.5)$$

The Alternating Least Square (ALS) method [35] is based on Matrix Factorization, which basically means that given a user-item interaction matrix $R \in \mathbb{R}^{n \times m}$, we want to represent it as the matrix product of the user embedding matrix $X \in \mathbb{R}^{k \times n}$ and the item embedding matrix $Y \in \mathbb{R}^{k \times m}$. That is, $R \approx X^\top Y$. Here, n, m and k are the number of users, the number of items, and the embedding dimension, respectively. To achieve this, the ALS method minimizes $L(X,Y)$ in the equation (3.5), where the optimization is performed by alternating between X and Y. The training process stops when $L(X,Y)$ no longer decreases meaningfully during optimization.

Neural Graph Collaborative Filtering

Neural Graph Collaborative Filtering(NGCF) model intends to reflect a new characteristic called *High-order Connectivity* in the model [5]. In this way, to represent collaborative signals in high-order connectivity, the idea of GNN is utilized in the embedding propagation layer.

Unlike the existing GCN, it encodes user-item interaction into the message passaing. Because of this interaction, messages depend on the affinity between the two embeddings, allowing more messages to be delivered from similar items.

The propagation rule of NGCF in matrix form was proposed as follows

$$E^{(k)} = \text{LeakyReLU}((\hat{A} + I)E^{(k-1)}W_1^{(k)} + \hat{A}E^{(k-1)} \odot E^{(k-1)}W_2^{(k)})) \quad (3.6)$$

where $E^{(k)} \in \mathbb{R}^{(N+M) \times d_l}$ are the embeddings for users and items obtained after kth layer, $\odot$ denotes element-wise product, and $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is normalized adjacency matrix of bipartite graph. $E^{(0)}$ is the initial embeddings for users and items. They generate final embeddings by concatenating all

$E^{(k)}$, and conduct the inner product to estimate user's preference towards the target item.

LightGCN

In a follow-up study, LightGCN [6], it was confirmed that the two characteristics of GCN in NGCF models, *feature transformation* and *non-linear activations*, do not contribute significantly to the performance of collaborative filtering. Even worse, adding them made learning more difficult and less recommended performance. As a result, a concise and accurate LightGCN model was proposed by simplifying NGCF. In particular, LightGCN learned user and item embedding by linearly propagating in the user-item interaction graph and used the sum of weights of the learned embeddings in all layers as the final embedding without using any nonlinear activation functions.

The propagation rule of LightGCN in matrix form was proposed as follows

$$E^{(k)} = (D^{-\frac{1}{2}}AD^{-\frac{1}{2}})E^{(k-1)}. \tag{3.7}$$

3.3 Salient Object Detection

Figure 3.1: Reference image and Salient Object Detection

Salient object detection is important for computer vision and image processing tasks because it enables the identification and extraction of the most visually prominent objects or regions in an image. The salient objects are those that capture the attention of human observers and are visually distinct from their surroundings.

There are two main types of Salient Object Detection (SOD) methods that use deep learning. One uses multi-layer perceptrons (MLPs) to find a coarse salient object region, and then uses a CNN to segment the salient object region in detail. This is called the classic convolutional network (CCN) based method. The second method is called fully convolutional networks (FCN), and since SOD is inherently prone to segmentation, many researchers use FCN-based methods [36].

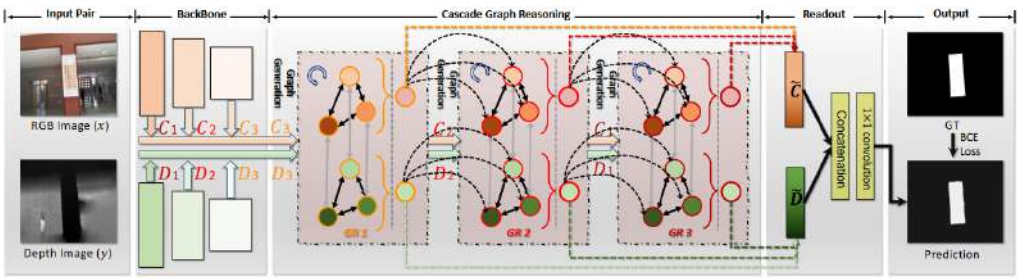


Figure 3.2: The overall architecture of our Cas-GNN

In the field of SOD tasks, researchers have also been using GNNs to model graph structures. One of them is an SOD model called Cascade GNN, which tries to solve the SOD task on RGB-D datasets. In this model, the basic structure is similar to the U-net [37] based segmentation model, and in addition the Cascade graph structure is used to fuse the RGB image and the depth image together for better learning [38].

Specifically, the structure of Cas-Gnn is as follows. First, they used VGG-16 [39] in Cas-Gnn as the backbone network. Internally, when the input of $3 \times 256 \times 256$ shape comes in, the 2d convolution is accumulated and the final feature output (or feature embedding) has the shape of $512 \times 32 \times 32$. However, in order to use *cascasding features*, Cas-Gnn also uses intermediate feature embeddings before the final output in a U-net-like fashion. It utilizes two of the intermediate features, one of which is $256 \times 64 \times 64$ shape and one of which is $128 \times 128 \times 128$ shape. Finally, Cas-Gnn employs the shapes of $(512 \times 32 \times 32)$, $(256 \times 64 \times 64)$, and $(128 \times 128 \times 128)$ of the backbone network.

And since these cascading features occur in both RGB and depth images, it can be considered that there are a total of 6 features, and Cas-Gnn utilizes

a GNN with 6 node features to systematically extract the cross-modality and high-order relationship between the 6 features. The graph message passing in the middle of the Figure 3.2 indicates that each GNN layer of these 6 nodes is passed through. The purpose of Cas-Gnn is to find an embedding representation for the input image comprehensively by leveraging GNNs, rather than just using the final feature embedding of the backbone network.

3.4 Rain Removal Tasks

Figure 3.3: Synthetic Rainy image and Ground-truth image

Deraining in autonomous driving is important because it helps improve the perception and performance of autonomous vehicles in rainy or adverse weather conditions. Rainfall can significantly impact the visibility of the environment, making it challenging for sensors and cameras to accurately detect and interpret objects, obstacles, and road markings. Therefore, deraining techniques aim to mitigate the negative effects of rain, enhance the quality of sensor data, and ensure the safety and reliability of autonomous vehicles in rainy weather conditions. It is a vital component in developing robust and all-weather autonomous driving systems.

For such deraining tasks, CNN-based models [40] and diffusion models [41] have been actively studied recently. Diffusion models have relatively superior deraining results, but the model is heavy, inference speed is slow, and training requires a lot of time and GPU resources. However, CNN-based derain models are relatively fast and have lightweight models, but their derain performance is not as good as diffusion models.

Deraining via Image Inpainting

Figure 3.4: Inpainting results of *LaMa*, (Suvorov et al., [42])

The deraining task can be approached directly as shown above, but it can also be approached in the manner of *Image Inpainting*. Image inpainting refers to the process of filling in missing or corrupted parts of an image with plausible content. There are also various image inpainting methods that use diffusion models ([43], [41]) or Vision Transformer-based models ([44], [45]). However, these methods have the limitation that the performance is good, but the model is heavy and the inference speed is relatively slow.

On the other hand, among the image inpainting models, those that use the convolution are relatively lightweight and therefore fast when inference time. Suvorov et al. [42] propose such an inpainting model, named *LaMa*. The distinctive feature of the model is that it employs a convolution module called Fast Fourier convolution (FFC) [46].

This FFC module uses an FFT internally, which is then processed with a 1 by 1 convolution and nonlinear activation, and then an inverse FFT. So while 2d-convolution usually relies heavily on local features, the FFC module has the advantage of extracting global features as well by taking the Fourier transform. In particular, when there are repetitive image patterns, this advantage seems to be more noticeable, so it is applied to image inpainting in LaMa by leveraging this advantage. The results in Figure 3.4 for image inpainting with LaMa illustrate these attributes. The pseudocode for the basic module (Fourier Unit, FU) of the FFC is given below.

Algorithm 1 Pseudocode of Fourier Unit (FU). (Chi et al., [46])

function FU(x)
    $y_r, y_i \leftarrow \text{FFT}(x)$ ▷ y_r/y_i: $[N, C, H, W/2]$
    $y \leftarrow \text{Concatenate}([y_r, y_i], \text{dim} = 1)$ ▷ $[N, C * 2, H, W/2]$
    $y \leftarrow \text{ReLU}(\text{BN}(\text{Conv}(y)))$ ▷ $[N, C * 2, H, W/2]$
    $y_r, y_i \leftarrow \text{Split}(y, \text{dim} = 1)$ ▷ y_r/y_i: $[N, C, H, W/2]$
    $z \leftarrow \text{iFFT}(y_r, y_i)$ ▷ $[N, C, H, W]$
    return z
end function

Chapter 4

Spectral Analysis of Graph Neural Networks

Question: Given a graph $\mathcal{G} = (V, E)$, how do we define a convolution on $\mathcal{G}$?

Before answering the question, let us consider the convolution operation in Euclidean space $\mathbb{R}^d$ and discrete ring graph R_n.

4.1 Schwartz space $\mathscr{S}(\mathbb{R}^d)$ and Ring graph R_n

At first, consider the space of Schwartz functions. The space of Schwartz functions is defined as

$$\mathscr{S}(\mathbb{R}^d) = \left\{ f \in C^\infty(\mathbb{R}^d, \mathbb{R}) \,:\, \sup_{x \in \mathbb{R}^d} |x^\alpha \partial_\beta f(x)| < \infty \quad \forall \alpha, \beta \right\}, \tag{4.1}$$

where α, β are multi-indices. If $f, g \in \mathscr{S}(\mathbb{R}^d)$, then the convolution $f * g$ is defined as follows:

$$f * g(x) := \int_{-\infty}^{\infty} f(x - y) g(y) dy \tag{4.2}$$

Also, let the Laplacian operator $\Delta := \sum_{i=1}^{d} \dfrac{\partial^2}{\partial x_i^2}$. We readily recognize that the following equation is satisfied.

$$\Delta(f * g) = (\Delta f) * g \tag{4.3}$$

Therefore, the operator $\Delta : \mathscr{S}(\mathbb{R}^d) \to \mathscr{S}(\mathbb{R}^d)$ and the operator $*g : \mathscr{S}(\mathbb{R}^d) \to \mathscr{S}(\mathbb{R}^d)$ commute.

In more depth, let us analyze the Laplacian and the convolution operations with Fourier transforms. At first, let f be given as above, and if we take a Fourier transform to Δf, then

$$\Delta f(x) = \Delta \int_{\mathbb{R}^d} \hat{f}(\xi) e^{2\pi i x \cdot \xi} d\xi \tag{4.4}$$

$$= \int_{\mathbb{R}^d} \hat{f}(\xi) \Delta e^{2\pi x \cdot \xi} d\xi \tag{4.5}$$

$$= \int_{\mathbb{R}^d} (-4\pi^2 |\xi|^2) \hat{f}(\xi) e^{2\pi i x \cdot \xi} d\xi. \tag{4.6}$$

And again, if we take the inverse Fourier transform, we get

$$\widehat{\Delta f}(\xi) = (-4\pi |\xi|^2) \hat{f}(\xi), \tag{4.7}$$

since Δf has a unique representation $\int_{\mathbb{R}^d} \widehat{\Delta f}(\xi) e^{2\pi x \cdot \xi} d\xi$.

Now, we take the Fourier transform of the convolution operation $f \mapsto f * g$. Then, by the Convolution theorem of the Fourier transform,

$$\widehat{f * g}(\xi) = \hat{f}(\xi) \cdot \hat{g}(\xi) \tag{4.8}$$

In the case of these two Fourier transforms, we can see that the Fourier transform *diagonalizes* the Laplacian and convolution operator in the frequency domain.

Secondly, let's look at the ring graph. Figure 4.1 is an example of ring graphs R_n. The adjacency matrix A and the graph Laplacian L of the ring graph R_n are as follows.

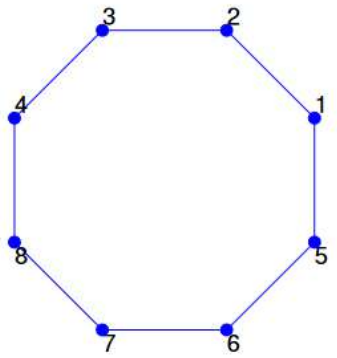


Figure 4.1: Ring graph R_8

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 1 \\ 1 & 0 & 1 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 2 & 0 & 0 & \dots & 0 & 0 \\ 0 & 2 & 0 & \dots & 0 & 0 \\ 0 & 0 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & 0 \\ 0 & 0 & 0 & \dots & 0 & 2 \end{bmatrix}$$

$$L = D - A = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 & -1 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & -1 \\ -1 & 0 & 0 & \dots & -1 & 2 \end{bmatrix}$$

The convolution operation on the ring graph $R_n = (V, E)$ acts on the functions $f, g : V \to \mathbb{R}$. Also we can identify f and g as elements of R^n, so let $f = (x_0, x_1, \cdots, x_{n-1})$ and $g = (c_0, c_1, \cdots c_{n-1})$. Therefore, the convolution $f * g \in \mathbb{R}^n$ is defined as follows.

$$f * g(k) = \sum_{j=0}^{n} x_j c_{k-j} \quad (k = 0, 1, \cdots n-1) \tag{4.9}$$

In the above equation, if $m < 0$ for c_m, then we define $c_m := c_{n+m}$. Once we've defined the convolution as above, we can represent the operation as a matrix-vector product, which is shown below.

$$f * g = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_3 & c_2 \\ c_2 & c_1 & c_0 & \dots & c_4 & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n-2} & c_{n-3} & c_{n-4} & \dots & c_0 & c_{n-1} \\ c_{n-1} & c_{n-2} & c_{n-3} & \dots & c_1 & c_0 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \tag{4.10}$$

To analyze it a bit further, let the above circulant matrix

$$C := \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_3 & c_2 \\ c_2 & c_1 & c_0 & \dots & c_4 & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n-2} & c_{n-3} & c_{n-4} & \dots & c_0 & c_{n-1} \\ c_{n-1} & c_{n-2} & c_{n-3} & \dots & c_1 & c_0 \end{bmatrix} \tag{4.11}$$

.

Then, by a simple calculation, we get

$$L \cdot C = C \cdot L \tag{4.12}$$

. This result is an analogue of the equation (4.3).

Also, we take a Discrete Fourier transform to these objects. For instance, if we denote the DFT matrix W as

$$W = \frac{1}{\sqrt{n}} \begin{bmatrix} \omega_0^0 & \omega_0^1 & \omega_0^2 & \dots & \omega_0^{n-1} \\ \omega_1^0 & \omega_1^1 & \omega_1^2 & \dots & \omega_1^{n-1} \\ \omega_2^0 & \omega_2^1 & \omega_2^2 & \dots & \omega_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_{n-1}^0 & \omega_{n-1}^1 & \omega_{n-1}^2 & \dots & \omega_{n-1}^{n-1} \end{bmatrix} \tag{4.13}$$

, where $\omega_k = e^{-2\pi ik/n}$, then we get

$$\hat{f} = W \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = Wf, \ \hat{g} = W \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = Wg \tag{4.14}$$

.

Also,

$$\widehat{f * g} = W \cdot C \cdot f \tag{4.15}$$
$$= diag(\hat{g}) \cdot W \cdot f \tag{4.16}$$
$$= \hat{g} \odot \hat{f} \tag{4.17}$$

.

This result is an analogue of the equation (4.8). Next, if we apply DFT to the graph Laplacian, then

$$\widehat{Lf} = W \cdot Lf \tag{4.18}$$
$$= \begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & 2 - 2\cos\left(\frac{2\pi k}{n}\right) & \vdots \\ 0 & 0 & \dots & 2 - 2\cos\left(\frac{2\pi(n-1)}{n}\right) \end{bmatrix} \cdot Wf \tag{4.19}$$
$$= \Lambda \cdot \hat{f} \tag{4.20}$$

, where Λ is a diagonal matrix of whose elements are eigenvalues of the graph Laplacian L. This result is an analogue of the equation (4.7).

In this case, as in Schwartz space, we can observe that the Fourier transform *diagonalizes* the graph Laplacian and the convolution operation in the frequency domain.

Furthermore, the basic theorem of linear algebra in finite dimensional vector spaces tell us more:

Theorem 4.1.1. *If two linear operator L and M on finite dimensional vector space V are diagonalizable and $LM = ML$, then they are simultaneously diagonalizable. That is, there exists an invertible matrix U such that U diag-*

onalize both L and M, i.e., ULU^{-1} and UMU^{-1} are both diagonal matrices.

In particular, for ring graphs R_n, we can see that the DFT matrix W diagonalizes both the graph Laplacian L and the convolution matrix C. Using the same notation as above, this is equivalent to

$$WCW^{-1} = diag(\hat{g})$$
$$WLW^{-1} = \Lambda$$

4.2 Convolution on General Graphs

We can define convolutions such as Euclidean space $\mathbb{R}^n$, ring graphs R_n, or image data which has grid structures, as discussed in the previous section 4.1.

More generally, in a locally compact abellian group $(G, +)$ with a Haar measure μ that has the property of translation invariance, the convolution $f * g$ for $f, g \in L^1(G)$ can be defined (almost everywhere) as follows.

$$f * g(x) = \int_G f(x - y)g(y)d\mu(y) \quad (x, y \in G). \tag{4.21}$$

From Young's inequality $||f * g||_{L^1(G)} \leq ||f||_{L^1(G)}||g||_{L^1(G)}$, we know that $f * g$ is defined almost everywhere, and belongs to $L^1(G)$. Moreover, we still have the following property, for all $\xi \in \hat{G}$,

$$\widehat{f * g}(\xi) = \hat{f}(\xi)\hat{g}(\xi) \tag{4.22}$$

, where $\hat{G}$ is the Pontryagin dual of G.

In this general view, Euclidean space $\mathbb{R}^n$ is a group with addition operations $(\mathbb{R}^n, +)$, and a ring graph R_n can be seen as a finite abelian group $(\mathbb{Z}/n\mathbb{Z}, +)$. Also, image data stored on a computer can be viewed as $((\mathbb{Z})^2, +)$ or $((\mathbb{Z}/n\mathbb{Z})^2, +)$. Thus, for those spaces, the convolution operation could be defined in terms of the above expression (4.21). However, in general, the group operation is not given for a general graph $\mathcal{G} = (V, E)$.

Now let's examine the relationship between Laplacian and convolution operations. The Laplacian Δ of $(\mathbb{R}^n, +)$ as a differential operator commutes with the convolution operator by the equation (4.3). In addition, the graph Laplacian L of $(\mathbb{Z}/n\mathbb{Z}, +)$ also commutes with the convolution, the circulant matrix C by the equation (4.12). Note that we can assign a finite abelian group $(\mathbb{Z}/n\mathbb{Z}, +)$ the same connectivity information as a ring graph R_n, i.e., we can view a finite abelian group $\mathbb{Z}/n\mathbb{Z}$ as a ring graph R_n.

Now let's define a convolution on a general graph $\mathcal{G} = (V, E)$. We still have the graph Laplacian $L = D - A$ and the function $f : V \to \mathbb{R}$ or $f \in \mathbb{R}^n$ where $n = |V|$. The convolution is a linear operator, so if we define a convolution on $\mathcal{G}$, it can be represented as a n by n matrix. We also want this convolution matrix C to still have the same properties as equations (4.3) and (4.3). And this commutativity and Theorem 4.1.1 provide a way to define convolution operations on general graphs.

If we further assume the diagonalizability of L and C, then the above commutativity and Theorem 4.1.1 let us know that there must exist an invertible matrix $U \in \mathbb{R}^{n \times n}$ that diagonalizes both L and C. In other words, $C = U^{-1} D_c U$ and $L = U^{-1} \Lambda U$, where D_c is a diagonal matrix whose elements are eigenvalues of C and Λ is a diagonal matrix whose elements are eigenvalues of L.

From these results, we can see that the convolution matrix C is eventually determined by the invertible matrix U and its n eigenvalues. However, since we don't know these simultaneously diagonalizing invertible matrices U, we still have a problem. If the eigenvalues of L are all different, then the set of matrices that diagonalize L is specific, so if we fix one such U and determine the eigenvalues of C, then the convolution matrix C will be fixed. However, even with this method, if the number of nodes in the graph is large , $n = |V| \gg 1$, the complexity of finding the eigendecomposition of L is $O(n^3)$, which is computationally intractable.

Thus, the spectral-based graph neural networks discussed in section 3.1 define the graph convolution under the following assumptions.

Assumption 1. The convolution matrix C and the graph Laplacian L are diagonalizable.

Assumption 2. The convolution matrix C and the graph Laplacian L commute.

Assumption 3. The eigenvalues of the graph Laplacian L are all distinct.

The above three assumptions are the fundamental assumption of Spectral-based GNNs. The Assumption 1 and 2 are necessary for the Theorem 4.1.1. If these assumptions are satisfied, consider the following two eigendecomposition versions of $\tilde{C}_\theta$ to approximate and learn the convolution matrix C.

$$\tilde{C}_\theta = U^{-1} \cdot \mathrm{diag}(g_\theta(\lambda_1, \cdots, \lambda_n)) \cdot U \tag{4.23}$$

$$\tilde{C}_\theta = U^{-1} \begin{bmatrix} g_\theta(\lambda_1) & 0 & 0 & \dots & 0 & 0 \\ 0 & g_\theta(\lambda_2) & 0 & \dots & 0 & 0 \\ 0 & 0 & g_\theta(\lambda_3) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & g_\theta(\lambda_{n-1}) & 0 \\ 0 & 0 & 0 & \dots & 0 & g_\theta(\lambda_n) \end{bmatrix} U \tag{4.24}$$

where g_θ might be constructed via a neural network with parameters θ and $\lambda_1, \cdots \lambda_n$ are all distinct eigenvalues of the graph Laplacian L. Note that g_θ in the equation (4.23) is a function from $\mathbb{R}^n$ to $\mathbb{R}^n$ and g_θ in the equation (4.24) is a function from $\mathbb{R}$ to $\mathbb{R}$.

First, since the input domain and output domain of g_θ are fixed to $\mathbb{R}^n$ in the case of the equation (4.23), the same g_θ cannot be used for graphs with different numbers of nodes. This means that we need a different model of g_θ depending on the number of nodes in the graph, or we need to devise a novel model to solve this problem so that g_θ does not need to have a fixed domain and codomain.

On the other hand, in the case of the equation (4.24), since g_θ is applied to each eigenvalue of $\tilde{C}_\theta$, g_θ can be adaptively applied to graphs with different numbers of nodes, i.e., the convolution matrix can be estimated for each input

graph, even if the adjacency matrix and the corresponding graph Laplacian change as the input graph changes.

So let us inestigate the estimation in the equation (4.24) a bit more for its general applicability. Given a graph $\mathcal{G} = (V, E)$, suppose there exists an optimal convolution matrix $C * (\mathcal{G})$ of that graph, and let the eigenvalues of $C * (\mathcal{G})$ be $c_1^*, \cdots c_n^*$. Then, we observe that we just need to learn or find a function that interpolates the points $(\lambda_1, c_1^*), \cdots, (\lambda_n, c_n^*)$. Here, $\lambda_1, \cdots, \lambda_n$ and $c_1^*, \cdots, c_n^*$ are the values that depend on V and E of the given (input) graph $\mathcal{G} = (V, E)$.

Now, to simplify things further and make the computation easier, let's assume that g_θ is a polynomial, i.e,

$$g_\theta(\lambda) := \sum_{j=0}^{k} \theta_j \lambda^j \tag{4.25}$$

, where $k \in \mathbb{N}$ is a degree of the polynomial and $\theta = (\theta_1, \cdots, \theta_k)$ are parameters to be learned.

Under these polynomial conditions, the equation (4.24) can be simply written as a graph Laplacian as follows.

$$\tilde{C}_\theta = g_\theta(U^{-1}\Lambda U) = g_\theta(L) \tag{4.26}$$

In theory, given a fixed graph Laplacian L in a situation that satisfies the above three assumptions, we can interpolate exactly all the points we need to interpolate as long as the polynomial has degree at least $n - 1$.

At this point, we now see that in the case of the Spectral-based GNNs discussed in Section 3.1 those message passing networks are the graph convolution $\tilde{C}_\theta$ that follows the above equation (4.26). Of course, the models discussed in Section 3.1 did not specifically address **Assumption 3**, but we can easily conclude that it is an important theoretical assumption. This is because if the eigenvalues of the graph Laplacian of a graph $\mathcal{G}$ are some of the same (having multiplicity) or, in the extreme, all of the same, then a

convolution matrix C like the following cannot be represented.

$$C = U^{-1} \begin{bmatrix} c_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & c_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & c_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & c_{n-1} & 0 \\ 0 & 0 & 0 & \dots & 0 & c_n \end{bmatrix} U \tag{4.27}$$

, where $c_1 < c_2 < \cdots < c_n$, so that all eigenvalues are distinct.

Putting together the discussion so far, we have the following theorem, which concludes this section.

Theorem 4.2.1. *If one assumes the three assumptions in this section, i.e., Assumption 1, Assumption 2 and Assumption 3, then for every convolution matrix C of $\mathcal{G} = (V, E)$, there exists a polynomial p of degree greater than equal at least $|V| - 1$ such that the matrix C can be exactly represented by the form $C = p(L)$, where L is the graph Laplacian of $\mathcal{G}$.*

Further Discussions

As mentioned in the introduction to Section 4.2, a general graph $\mathcal{G}$ is not given a group operation and non-trivial Haar measure, so convolution operators on a locally compact abelian group G are not easily defined, nor is the Fourier transform. However, it seems that the invertible matrix U introduced in this section can take over the role, because for a convolution matrix given by $C = U^{-1}DU = U^{-1}\text{diag}(d_1, \cdots d_n)U$, the component vector $(d_1, \cdots d_n)^\top \in \mathbb{R}^n$ of the diagonal matrix D become an element in the frequency domain [1]. As a consequence, a convolution of $f \in \mathbb{R}^n$ with the matrix C is expressed as an element-wise product on $\hat{f}$, which is equivalent to $\widehat{Cf} = UCf = UCU^{-1}Uf = (d_1, \cdots, d_n)^\top \odot \hat{f}$. Therefore, in the future, when researching GNNs with spectral analysis, one might attain a better analysis by utilizing U as a substitute of the Fourier transform, and in addition, one might develop a novel GNN model that performs better with this approach.

[1] Here, *the frequency domain* means the image of a matrix multiplication operator by U, i.e. $\{Uf \ : \ f \in \mathbb{R}^{|V|}\}$

Chapter 5

Proposed Method

Before proposing the method of each experiment, we would like to point out the motivation behind each experiment.

5.1 Proposal Background

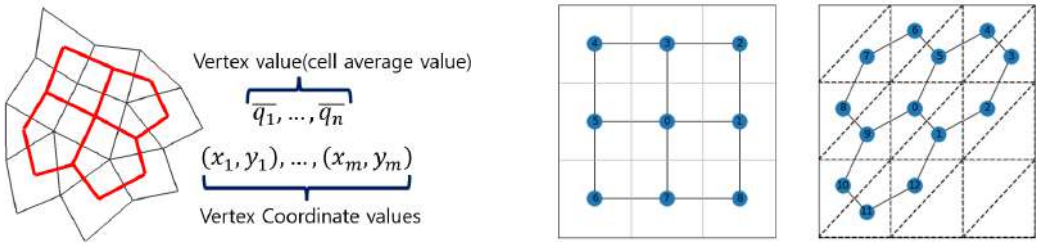
As a first attempt to demonstrate the expressive power of spectral-based GNNs, we will show that spectral-based GNNs are sufficient for solving PDEs in **Computational Fluid Dynmaics (CFD)** instead of traditional graph data such as citation networks or molecule dataset.

For **collaborative filtering**, we compared a Matrix Factorization-based method [35] and spectral-based GNNs, such as NGCF and LightGCN. In addition, to improve the recommendation performance of spectral-based GNNs, we show that the category information of items can be used in combination with these data to improve the recommendation performance of collaborative filtering.

In the case of **Salient Object Detection**, we want to learn the structure (DAG) of the existing model, Cas-Gnn, through *structure learning*. The DAG obtained can be viewed as an adjacency matrix of a graph, where each cascading model feature acts as a node, and the adjacency matrix obtained through DAG learning aims to improve performance by aggregating features using spectral-based GNN's messaging passing.

For the **Rain Removal Task**, we will show that it is empirically effective to design a model efficiently by reflecting the features of rainstreaks in rainy images and the results obtained through Fourier analysis, rather than training the model end-to-end.

5.2 Spectral GNNs to Computational Fluid Dynamics



(a) Required domain for MLP limiter (b) Graph structure (Regular Grid)

Figure 5.1: An illustration of the data for the experiment

Method Formulation

The experiments are designed in two main ways. One is to regress one of the slope limiters, Multi-dimensional Limiting Process (MLP) limiter [47], with FCN and GNNs. The other is to actually run the simulation with the limiter obtained in the first experiment for two basic PDEs, Burgers' equation and Linear advection equation.

To compute the MLP limiter at each cell C, we need information from all cells adjacent to all vertices in the cell C, as shown in Figure 5.1 (a). Specifically, we need the cell average value of the cells and the vertex coordinates of each cell.

The experiments were conducted using the Finite Volume Method (FVM) for basic PDEs, dividing the analysis domain into irregular mixed meshes. The Burgers' equation and Linear advection are tested for the simulation experiment. The models used in the experiments are Fully Connected Net-

work(FCN), GCN, and PointNet [48]. A visualization of the computational graph of PointNet used for the experiment is shown as follows.

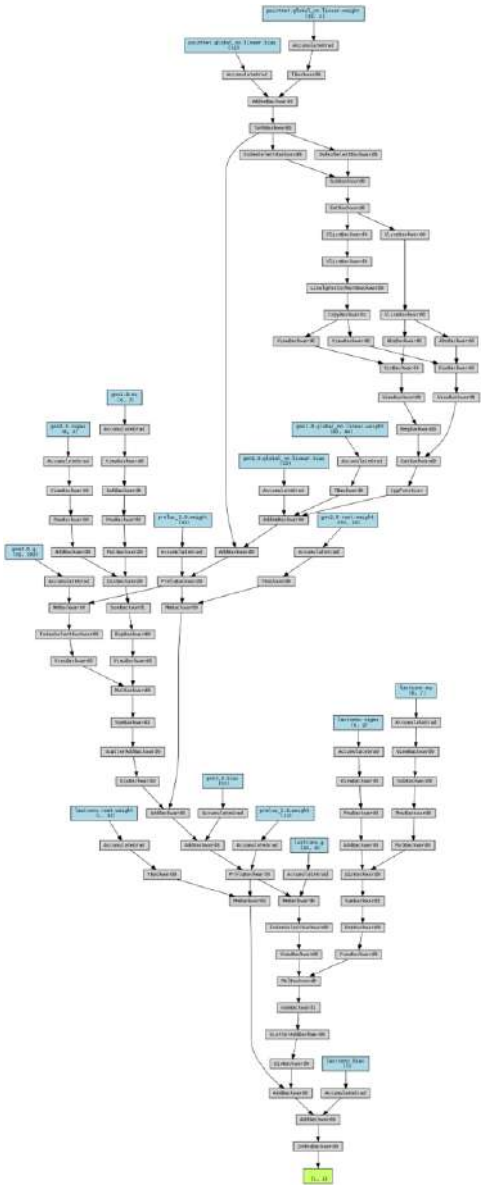


Figure 5.2: Computational graph of PointNet used for the experiment.

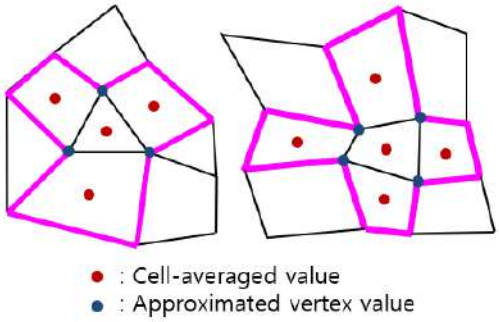


Figure 5.3: Construction of FCN Input

Unlike GNN, FCN can only be applied to a fixed grid shape, since the shape of the input changes when the structure of the graph changes. In this case, we set the input data as cell average value and approximated vertex value as shown in Figure 5.3.

In the case of GNN, the graph structure is input data as shown in Figure 5.1 (b), and for node feature, the cell average value and gradient value of each node (cell) are given as features. In other words, the dimension of node feature is $1 + 2 = 3$.

5.3 Collaborative Filtering

Method Formulation

GCN-based NGCF and LightGCN use the adjcency matrix of users and items as $A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix}$ with the user-item interaction matrix $R \in \mathbb{R}^{M \times N}$. The lower right of this block matrix means that there is no interaction between the items. Since general GCN is applicable not only to the bipartite graph but also to general graphs, it is possible to add interaction between items using item category data. For example, from the $C \in \mathbb{R}^{N \times N_c}$ matrix containing the category information of the item, it is possible to make the adjacency matrix $\hat{C} \in \mathbb{R}^{N \times N}$ between the items as follows

$$\hat{C} = (c_{i,j}), \quad \text{where} \quad c_{ij} = \begin{cases} 1 & \text{, if i and j share at least one category} \\ 0 & \text{, otherwise} \end{cases}$$

Alternatively, it is also possible to allocate corresponding weights in proportion to the number of categories shared by the two items.

With $\hat{C}$ which represents interactions between items, GCN using adjacency matrix as $A = \begin{pmatrix} 0 & R \\ R^T & \hat{C} \end{pmatrix}$ was applied to NGCF and LightGCN.

5.4 Salient Object Detection

Salient object detection (SOD) is the computer vision task that enhances our understanding of visual attention, aids in image and video analysis, and enables various computer vision applications across different domains. Therefore, in this work, we adaptively learn the optimal structure between shallow and deep features in the SOD model and demonstrate the performance of the model using spectral-based GNNs.

Method Formulation

Basically, we use Cas-Gnn's approach as a backbone for Salient Object Detection. If you examine the structure of Cas-Gnn's model, there are two main lines: one is the intermediate values for the input of RGB images, and the other is the intermediate values for the input of depth images.

As mentioned in Section 5.1, the adjacency matrix used by the GNN layer of Cas-Gnn was fixed. To elaborate on this, let's call the nodes for the RGB image obtained by the backbone network c_1, c_2, c_3, and the nodes for the depth image d_1, d_2, d_3. Then the connectivity of the graph $\mathcal{G} = (V, E)$ in Cas-Gnn is $(c_1, d_1), (c_2, d_2), (c_3, d_3) \in E$, $(c_1, c_2), (c_2, c_3), (c_1, c_3) \in E$, and $(d_1, d_2), (d_2, d_3), (d_1, d_3) \in E$.

However, it is the assumption is based on what the authors of Cas-Gnn believe is a reasonable graph structure for the internal cascading features of the model. In other words, the optimal graph structure may not necessarily be the same as above. Therefore, we want to learn this graph structure using the methods of DAG learning covered in Section 2.3.

Let F_{rgb}, F_d be the backbone networks for the RGB image and the depth image, and let A be the adjacency matrix between the features in the model that we want to find, and introduce the following $h(A)$ from Section 2.3.

$$h(A) = tr[(I + \alpha A \odot A)^d] - d \quad (A \in \mathbb{R}^{d\times d},\ \alpha > 0) \tag{5.1}$$

, where $\odot$ means element-wise multiplication.

Then, using the linear SEM in the equation (2.6) and the structure of [32], we can construct the following ELBO expression, which has the structure of a variational autoencoder [49].

$$\begin{aligned}
MLP(X, W^1, W^2) &:= ReLU(XW^1)W^2 \\
[M_Z|\log S_Z] &= (I - A^\top)MLP(X, W^1, W^2) \\
[M_X|\log S_X] &= MLP((I - A^\top)^{-1}Z, W^3, W^4) \\
D_{KL}(q(Z|X)||P(Z)) &= \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{n}(S_Z)_{ij}^2 + (M_Z)_{ij}^2 - 2\log(S_Z)_{ij} - 1 \\
E_{q(Z|X)}\Big[\log p(X|Z)\Big] &\approx \sum_{i=1}^{d}\sum_{j=1}^{n} -\frac{(X_{ij} - (M_X)_{ij}^2}{2(S_X)_{ij}^2} - \log(S_X)_{ij} \\
f(X, A) &= -E_{q(Z|X)}\Big[\log p(X|Z)\Big] + D_{KL}(q(Z|X)||P(Z))
\end{aligned}$$

, where W^1, W^2, W^3, W^4 are trainable parameter and X is a node feature.

To summarize, the node feature matrix X created by concatenating all $c_1, c_2, c_3, d_1, d_2, d_3$ obtained from the backbone network F_{rgb}, F_d enters $f(X, A)$ in the above expression. Therefore, the final loss L_{DAG} for DAG structure learning is as follows.

$$L_{DAG} = f(X, A) + \lambda h(A) \tag{5.2}$$

, where λ is a hyperparameter.

Hence, training proceeds as follows. First, fix the initialized $A \in \mathbb{R}^6$ and update the parameter with the AdamW [50] optimizer for the loss function of the existing Cas-Gnn. Then, get the node feature X from the updated Cas-Gnn, put it into L_{DAG} in the above expression, and update the parameter with another optimizer for A. In other words, just as the discriminator and generator modules in the GAN [51] are trained by two optimizers alternately, the proposed model is trained by two separate optimizers alternately between

the Cas-Gnn model and the DAG learning model.

5.5 Rain Removal Task

(a) Ground-truth image (no rain)

(b) Rainy image

Figure 5.4: Sample images of RainDS dataset

(a) Low-band filtered image (no rain)

(b) Low-band filtered image (rain)

Figure 5.5: Comparison of Low-band filtered images

The rain removal task can play an important role in autonomous driving, as discussed in Section 3.4. These tasks need to be handled in real time to achieve safety, responsiveness, efficient decision-making, and adaptability in autonomous driving.

Motivation

The model is motivated by the following observation. Figure 5.4 shows an sample image of rain from the RainDS dataset. The shape of this image is $3 \times W \times H$. Now, if we take this image and perform a Discret Fourier Transform using the FFT, we get $3 \times W \times H$, which is made up of complex numbers. Now, after replacing the FFT values of the $0.02W \times 0.02H$ region

at the four corners with zeroes, then we take the inverse FFT and convert it elemenetwisely to absolute values, resulting in Figure 5.5. As you can see in Figure 5.5, the pattern or signal corresponding to the high frequency is still there. The remaining signals show a high proportion of signals that appear to be rain.

In addition to Figure 5.4, for the entire RainDS data with rainstreak ($\mathcal{D}_{\text{rain}}$) and the entire RainDS data without rainstreak ($\mathcal{D}_{\text{norain}}$), only the four corners are truncated to 0 after FFT as before, i.e., the low frequencies are filtered out. Then, we take the sum of the absolute values of the filtered values, and compare the difference between $\mathcal{D}_{\text{rain}}$ and $\mathcal{D}_{\text{norain}}$.

Table 5.1: Sum of Low-band filtered values of RainDS dataset

	Rain ($\mathcal{D}_{\text{rain}}$)	No rain ($\mathcal{D}_{\text{norain}}$)
Sum	179,009	171,881

As you can see from the table 5.1, the value is higher for $\mathcal{D}_{\text{rain}}$ with rain. In other words, we can see that the rainy image contains a lot of high frequency information. Based on the characteristics of these rainy images, we propose the following method formulation to remove the rain.

Method Formulation

Considering the nature of the deraining task, we design the deraining process in two steps. The first step is to segment the rainstreak in the rainy image. The second step is to apply convolution-based image inpainting by masking the rainstreak segmentation area found in the first step. And we want lightweight models with fast inference at both stages.

In the first step, we want to learn to segment rainstreak pixels in a rainy image. For our train dataset, we used the RainDS dataset. This dataset is characterized by the fact that it is not a synthetic rainy image data, but an image taken in a real rainy environment, and the same image was taken in a non-rainy environment with the same settings. In other words, the data is not composed of synthesized rain images, but pairs of actual rainy and non-rainy images. In this dataset, we can subtract the rainy images from the non-rainy

images to get the desired segmentation result for real rain. We want to train our model with this subtracted image as the correct label.

Specifically, the model in the first step is influenced by the aforementioned motivation to take an FFT as an input and provide a signal with the low frequency portion blown out. We can assume that since our goal is to segment rainy pixels, the information in the low frequency part is not needed to train the model.

In the second step, we use the LaMa model mentioned in Section 3.4 to inpaint the image. In the case of the LaMa model, we did not train it using the rainstreak dataset, but used a pre-trained model for image inpainting. The data used to train LaMa is the Places [52] dataset. The dataset contains a collection of 10 million scene photographs that have been categorized based on their semantic scene labels. It encompasses a wide range of diverse environments encountered in the world, providing a comprehensive and extensive representation of different scene types. LaMa is then trained to perform image inpainting by applying some sorts of random specific forms of masking.

As a result, since rainy images often contain a large number of images similar to this Places dataset with rain falling, if we successfully segment the rainstreak pixels in the initial step, we can expect acceptable image inpainting results, which means reliable derained outcomes.

Chapter 6

Experiments

6.1 Spectral GNNs to Computational Fluid Dynamics

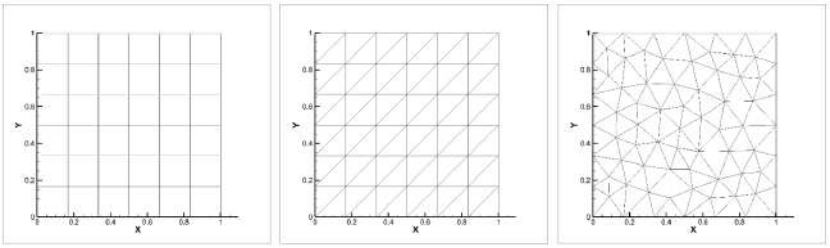


Figure 6.1: Various meshes generated for construction of datasets

6.1.1 Datasets

Table 6.1: Statistics of the datasets generated for the experiment

Dimension	Element type	Number of data
2-D	Triangular	100,000
	Quadrilateral	102,000

For the experiment, we created various meshes as shown in Figure 1 and recorded values such as cell average value, cell coordinate value, and cell

gradient using discrete functions that we can control such as step function. We also calculated the MLP limiter value of each cell as it was generated to create a supervised dataset for regression.

6.1.2 Experimental Results

Regression Result

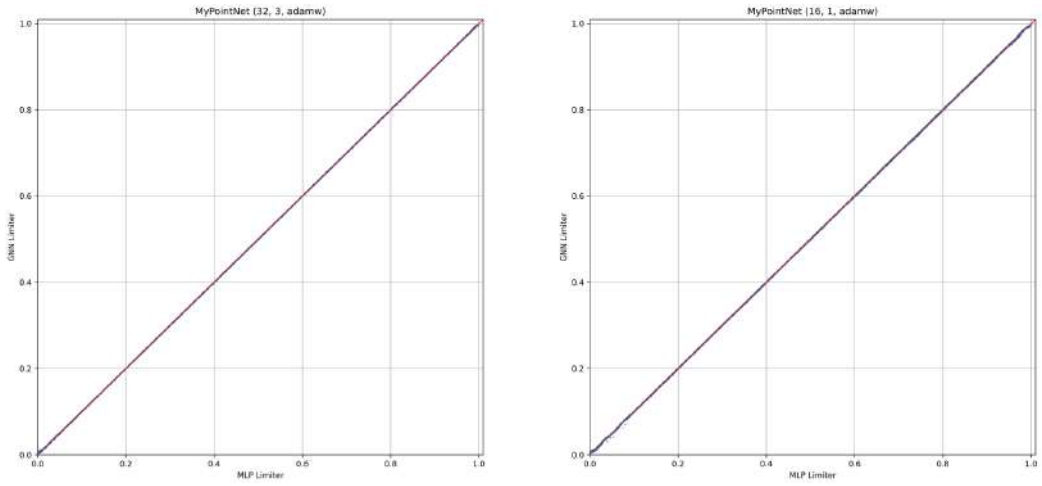


Figure 6.2: Regression result of PointNet

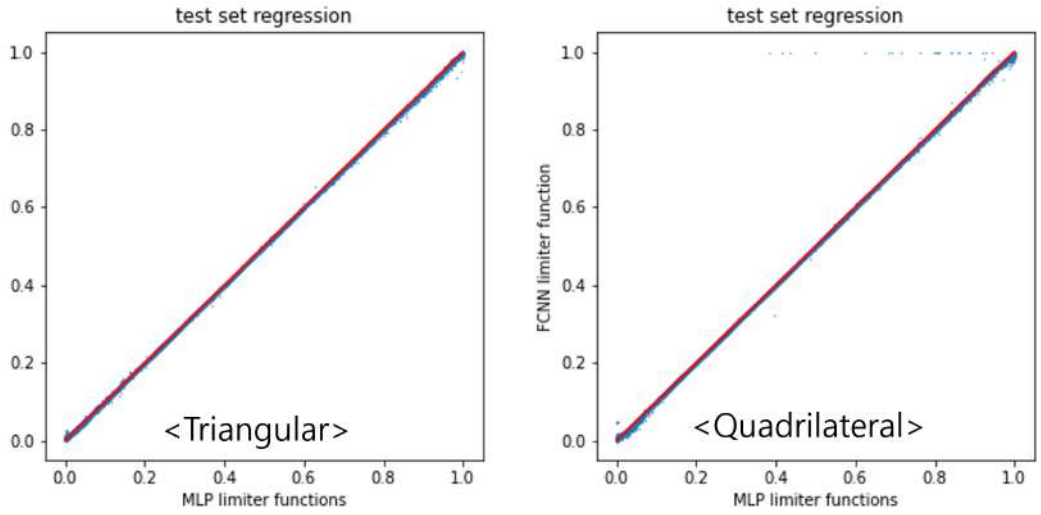


Figure 6.3: Regression result of FCN

Table 6.2: Regression results of proposed models

Model	Parameter (hidden,#layer)	MSE
GCN	(64,3)	6.06e-4
	(64,5)	8.04e-4
	(256,3)	3.36e-4
	(256,5)	5.26e-4
PointNet	(32,1)	3.805e-7
	(32,2)	3.320e-7
	(32,3)	**2.635e-7**
	(16,1)	8.527e-7
FCN	(32,5)	5.002e-7

Table 6.2 shows the results of the MLP limiter regression experiment. The results are 3.37e-4 for GCN, **2.635e-7** for PointNet, and 5.002e-7 for FCN. Comparing the FCN and GNN models, the GNN model shows better results in terms of MSE, and it is noteworthy that the GNN-based model can apply the learned model directly under any mesh grid, while the FCN model may not be applicable if the grid shape (Triangular or Quadrilateral) is different, which is because the input dimension of trained FCN model must be fixed.

Numerical Simulation Result

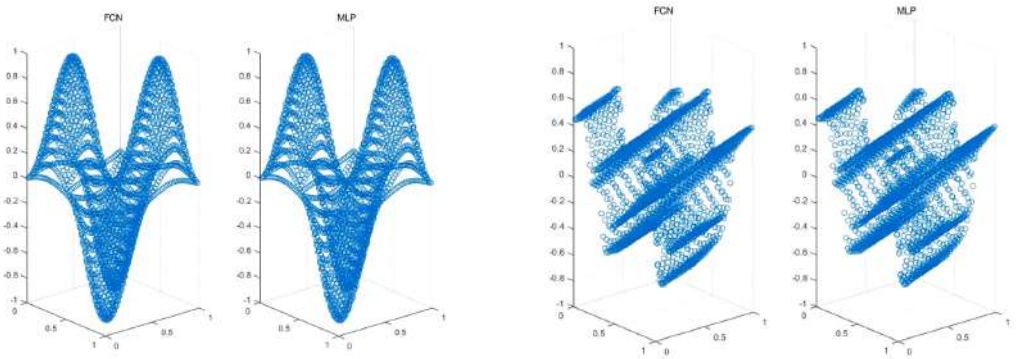
For the following Burgers' equation and Linear advection PDE, we conducted an experiment to compare the simulation results with the ground-truth MLP limiter and the simulation results with the model trained in the first step. The grid in the region $(x, y) \in [0, 1] \times [0, 1]$ is an irregular mixed mesh with a size of 100×100.

- Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial y} = 0 \tag{6.1}$$

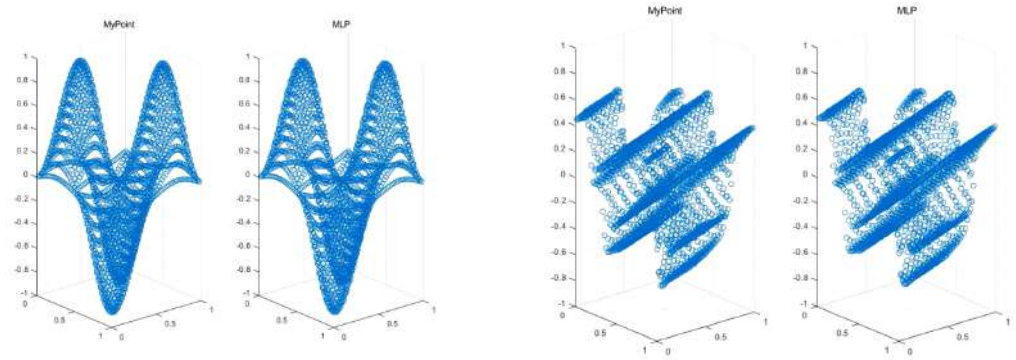
$$u_0(x, y) = \sin(2\pi x)\sin(2\pi y) \tag{6.2}$$

$$u_0(x, y) = \begin{cases} 1, & \text{if } 0.25 \le x \le 0.75,\ 0.25 \le y \le 0.75 \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$



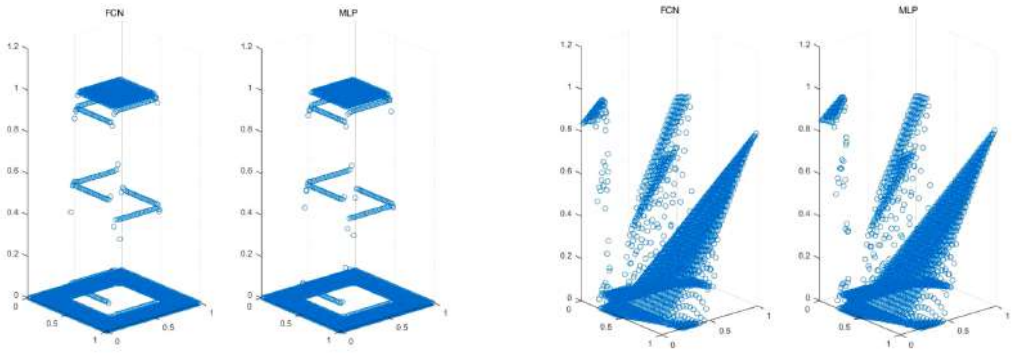
(a) $t = 0$ (b) $t = 1$

Figure 6.4: Numerical simulation result of FCN (Burgers' equation)



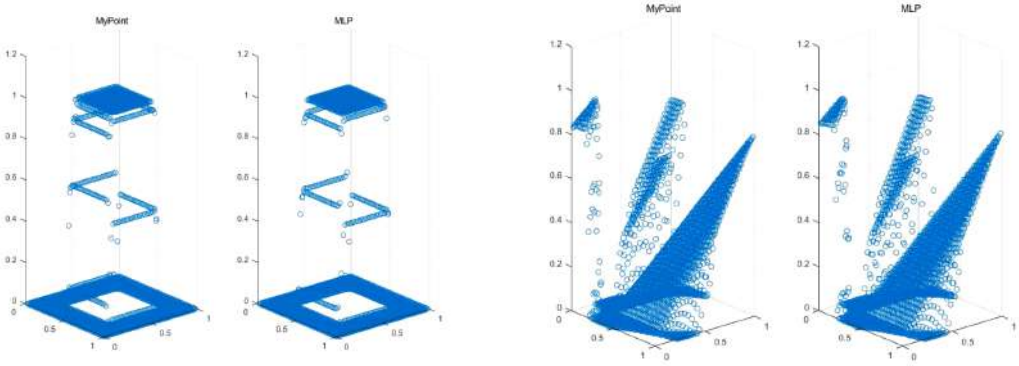
(a) $t = 0$ (b) $t = 1$

Figure 6.5: Numerical simulation result of Pointnet (Burgers' equation)



(a) $t = 0$ (b) $t = 1$

Figure 6.6: Numerical simulation result of FCN (Burgers' equation)



(a) $t = 0$ (b) $t = 1$

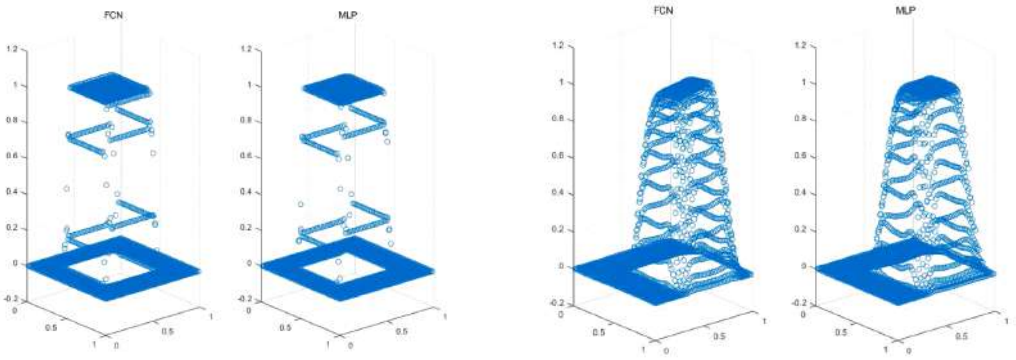
Figure 6.7: Numerical simulation result of Pointnet (Burgers' equation)

- Linear Advection

$$\frac{\partial u}{\partial t} + c_x \frac{\partial u}{\partial x} + c_y \frac{\partial u}{\partial y} = 0 \tag{6.4}$$

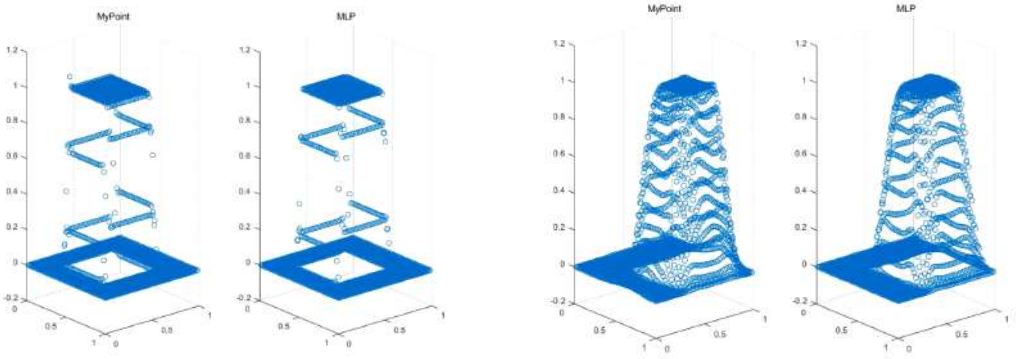
$$u_0(x, y) = \begin{cases} 1, & \text{if } 0.25 \leq x \leq 0.75,\ 0.25 \leq y \leq 0.75 \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

$$(c_x, c_y) = (1.0, 0.5) \quad \text{(Advection Speed)} \tag{6.6}$$



(a) $t = 0$ (b) $t = 1$

Figure 6.8: Numerical simulation result of FCN (Linear advection)



(a) $t = 0$ (b) $t = 1$

Figure 6.9: Numerical simulation result of Pointnet (Linear advection)

6.2 Collaborative Filtering

6.2.1 Datasets

Table 6.3: Statistics of the datasets

Datasets	#Users	#Items	#Interactions	Density	#Categories
MovieLens-100K	943	1,682	100,000	0.06305	19
CiteUlike-A	3,277	16,807	178,062	0.00323	46,391

In the experiment, two datasets were used in the experiment, including MovieLens-100k and CiteUlike-A. ALL of the data were processed with 20-core settings. N-core setting refers to a setting in which at least N items are rated for each user and at least one user rates each item.

MovieLens. The MovieLens dataset contains a set of movie ratings from the MovieLens website, a movie recommendation service [53]. This dataset was collected and maintained by GroupLens, a research group at the University of Minnesota. There are 5 versions included: "25m", "latest-small", "100k", "1m", and "20m". In all datasets, the movies data and ratings data are joined on "movieId". The 25m dataset, latest-small dataset, and 20m dataset contain only movie data and rating data. The 1m dataset and 100k dataset contain demographic data in addition to movie and rating data. In this thesis, MovieLens-100k is used.

CiteUlike. The Citeulike-A dataset, was used in the IJCAI paper (Wang *et al.* [54]). It was collected from CiteULike and Google Scholar, so it has two versions collected from CiteULike and Google Scholar independently, i.e., CiteUlike-A and CiteUlike-T. CiteULike allows users to create their own collections of articles. There are abstracts, titles, and tags for each article. In this thesis, CiteUlike-A is used.

In experiments, 10% of all intersections are randomly selected for training, and the remaining intersections are used for validation.

6.2.2 Evaluation Metric

To measure the performance of the recommendation system, *Normalized Discounted Cumulative Gain*(NDCG)@K[55, 56], Precision@K, and Recall@K are adopted. NDCG is one of the widely-used ranking metric computed by the all items and all users. NDCG metric address higher importance to the top ranks and grade successively lower ranks with marginal fractional utility. Discounted Cumulative Gain(DCG) for the particular rank position p is obtained by

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} \tag{6.7}$$

where rel_i is the *grade relevance* of the result at position i. The formula of DCG shows that highly relevant items appearing lower rank should be penalized as the graded relevance is reduced logarithmically. The alternative formulation of DCG can emphasize the relevance by replacing numerator to exponential term.

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i}}{\log_2(i+1).} \tag{6.8}$$

The two expressions of DCG are same when the relevance values are binary, $rel_i \in \{0, 1\}$. The final evaluation metric, NDCG for particular p is calculated by the ratio between DCG_p and IDCG_p,

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p} \tag{6.9}$$

where ideal discounted cumulative gain(IDCG) on best result list REL_p, IDCG_p is defined as

$$\mathrm{IDCG}_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i}}{\log_2(i+1).} \tag{6.10}$$

On the other hand, Precision@K and Recall@K mean simply the fraction of K recommended items, regardless of the order.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{6.11}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{6.12}$$

6.2.3 Bayesian Personalized Ranking

Implicit Data

In some case of recommendation system, the users' preferences can be expressed explicitly as positive or negative rating scores. However, in case of recommendation system for purchasing documentation, the negative data cannot be observed. The *implicit data* might be considered as *real negative feedback* or *missing value*, which has not been observed yet. Intuitively, the missing value should not be regarded as a negative feedback.

Personalized Total Ranking

Let U be the set of all users and I the set of all items. In the implicit feedback, the user-item pair is defined by $(u, i) = s \in S \subseteq U \times I$. Each user's preference is described as personalized total ranking, $>_u \subset I^2$, where $>_u$ meets the properties of a total order, meaning *totatlity*, *antisymmetry*, and *transitivity*, repectively:

$$\forall i, j \in I : i \neq j \quad \Longrightarrow \quad (i >_u j) \vee (j >_u i) \tag{6.13}$$

$$\forall i, j \in I : (i >_u j) \wedge (j >_u i) \quad \Longrightarrow \quad i = j \tag{6.14}$$

$$\forall i, j, k \in I : (i >_u j) \wedge (j >_u k) \quad \Longrightarrow \quad i >_u k \tag{6.15}$$

For convenience, we also define :

$$I_u^+ := \{i \in I : (u, i) \in S\} \tag{6.16}$$

$$U_i^+ := \{u \in U : (u, i) \in S\} \tag{6.17}$$

The training dataset $D_S : U \times I \times I$ is defined by :

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \backslash I_u^+\} \tag{6.18}$$

where user u of $(u, i, j) \in D_S$ is assumed to prefer i over j. This approach allows us to train data consists of both positive and negative pairs and missing

values. And the D_S addresses the actual objective of ranking.

Bayesian Personalized Ranking

Bayesian Personalized Ranking(BPR) optimization [57] is derived by a Bayesian analysis of the problem using the likelihood function for $p(i >_u j|\Theta)$ and the prior probability for the model parameter $p(\Theta)$. The Bayesian formulation is equivalent to maximizing the posterior probability.

$$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta) \tag{6.19}$$

where Θ is the learnable parameters.
As the above user-specific likelihood function $p(>_u |\Theta)$ follows Bernoulli distribution, it can be rewritten as follow :

$$p(>_u |\Theta) = p(i >_u j)^{\delta((u,i,j)\in D_S)}(1 - p(i >_u j))^{\delta((u,i,j)\notin D_S)}, \tag{6.20}$$

where δ is the indicator function.
Following the properties(totality, antisymmetry, and transitivity), the individual probability that a user really prefers item i to item j is defined as :

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta)) \tag{6.21}$$

where σ is the loggistic sigmoid and $\hat{x}_{uij}$ is obtained by *matrix factorization*.
For all users, the overall likelihood function is

$$\prod_{u\in U} p(>_u |\Theta) = \prod_{(u,i,j)\in D_S} p(i >_u j|\Theta) \tag{6.22}$$

$$= \prod_{(u,i,j)\in D_S} p(i >_u j)^{\delta((u,i,j)\in D_S)}(1 - p(i >_u j))^{\delta((u,i,j)\notin D_S)}. \tag{6.23}$$

Let $p(\Theta) \sim N(0, \sum_\Theta)$, the maximum posterior estimator to derive opti-

mization criterion for personalized ranking $\texttt{BPR} - \texttt{OPT}$ is

$$\begin{aligned}
\texttt{BPR} - \texttt{OPT} : &= \ln p(\Theta| >_u) \\
&= \ln p(>_u |\Theta) p(\Theta) \\
&= \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{x}_{uij} p(\Theta)) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2.
\end{aligned} \tag{6.24}$$

where λ_Θ are model specific regularization parameter.
We employ the this optimization as loss function, *Bayesian Personalized Ranking*(BPR) loss, which is a pairwise loss that encourages the prediction of an observed entry to be higher than its unobserved counterparts :

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\mathsf{E}^{(0)}\|^2. \tag{6.25}$$

6.2.4 Experimental Results

Table 6.4: Comparison between collobarative filtering methods

	MovieLens-100K			CiteUlike-A		
	ndcg@20↑	recall@20↑	precision@20↑	ndcg@20↑	recall@20↑	precision@20↑
MF (ALS)	0.0761	0.0357	0.0946	0.0350	0.0135	0.0336
NGCF	0.3538	0.0891	0.3303	0.0298	0.0090	0.0286
NGCF + category	**0.3769**	**0.1126**	**0.3585**	**0.0372**	**0.0106**	**0.0356**
LightGCN	**0.4984**	**0.1371**	**0.4739**	**0.0789**	**0.0241**	**0.0692**
LightGCN + category	0.3720	0.0892	0.3453	0.0393	0.0121	0.0364

Table 6.4 summarizes the result of using category information on the model's performance for two datasets and three base models. The Alternating Least Square (ALS) method based on Matrix Factorization (MF) performed worse than both spectral-based GNN methods on MovieLens data, and performed better than naive NGCF without category information on CiteUlike data, but the final performance was not as good as LightGCN without category information.

In the case of NGCF, the model with category data generally performed better for the two datasets. Interestingly, in the case of LightGCN, the performance of the model using additional category data has decreased.

For these results, we expect the difference to be caused by the presence or absence of learnable parameters inside the model. NGCF uses learnable parameters $W_1^{(k)}$ and $W_2^{(k)}$ in (3.6). On the other hand, LightGCN does not use learnable parameters other than the first embedding layer. Therefore, in the case of NGCF, message passing between the additional edges could be effectively utilized through internal parameters, and in the case of LightGCN, it can be interpreted that this was not the case.

In conclusion, MF-based methods try to factorize the user-item interaction matrix to obtain the embedding of user and item, so it is difficult to use additional information of these items. On the other hand, in GNN models such as NGCF, it is possible to significantly improve the performance of existing spectral-based GNNs by using information between items, i.e., category information, to reflect the graph information of user-items rather than simply reflecting user-item interaction.

6.3 Salient Object Detection

6.3.1 Datasets

Table 6.5: RGB-D datasets for Salient Object Detection

Dataset	Number of Images	Train/Test
NJU2K	1485	Train
NLPR	700	Train
RGBD135	135	Test
DUT-RGBD	1200	Test
LFSD	100	Test

We use the dataset NJU2K [58], NLPR [59], RGBD135[60] DUT-RGBD [61], LFSD [62]. These dataset are widely utilized to benchmark the performance of RGB-D Salient Object Detection.

6.3.2 Evaluation metrics

In the area of Salient Object Detection, there are five mainly used metrics: S-measure (S_α), F-measure (F_β), E-measure (E) and MAE.

E-measure is designed to capture image-level statistics and their local pixel matching information.

$$E = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} \phi_{FM}(i,j) \tag{6.26}$$

where W and H denote the width and height of the image, and ϕ_{FM} is the enhanced alignment matrix [63].

The S-measure [64] is introduced as a method to evaluate the similarity in structure between regional perception and object perception.

$$S_\alpha = \alpha S_o + (1 - \alpha) S_r \tag{6.27}$$

where S_r is the region-aware structural similarity measure and S_o is the objectp-aware structural similarity measure. In this experiment, we set $\alpha = 0.5$.

F-measure is the weighted harmonic mean of precision and recall.

$$F_\beta = (1 + \beta^2) \frac{Precision \times Recall}{\beta^2 Precision + Recall} \tag{6.28}$$

, where $\beta^2 = 0.3$ is set for our experiment.

Mean Absolute Error (MAE) is utilized to assess the average relative error at the pixel level between the ground truth and the normalized prediction.

$$MAE = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} |S(i,j) - G(i,j)| \tag{6.29}$$

, where W and H denote the width and height of the image, and G is the ground-truth and S is the normalized prediction.

6.3.3 Experimental Results

In this experiment, unlike Cas-Gnn, instead of using 3 feature embeddings from shallow to deep features of VGG19, we increased the number to 5 for our model. Then, the total number of nodes becomes 10 for both RGB and depth features. The purpose of this change is that we think that GNN's message passing can be more effective with more 10 nodes rather than 6.

Next, the training loss of our proposed model is the sum of the L_{DAG} required for DAG learning as discussed in Section 5.4 and the Salient Object Detection Loss (L_{SOD}) used by Cas-Gnn. These losses are also trained with two AdamW [50] optimizers.

However, models trained with two optimizers, such as GANs, often suffer unstable learning. In fact, if the adjacency matrix A, which we want to learn, is initialized near zero, it does not seem to learn well. To solve this problem, we initialized A as an i.i.d. random Gaussians with a mean of 1 and a standard deviation of 1. Then, we found that the training proceeded normally. As you can see from the loss curves in Figures 6.10 and 6.11, both losses are steadily reduced.

The learned DAG matrix as training progresses is shown in Figure 6.12. As we can see, it starts out with a high amount of 1s and changes shape as training progresses. Note that in this figure, the rows are the sources and the columns are the targets. Also, indices from 0 to 4 correspond to RGB nodes and indices from 5 to 9 correspond to depth nodes. Thus, as shown in (f) of Figure 6.12, we observe that the proposed model *learns a structure* in which overall RGB features affect depth features, and overall depth features affect depth features that are themselves.

Quantitative & Qualitative Results

Table 6.6: Quantitative comparisons with Cas-Gnn

	Ours				Cas-Gnn			
	$MAE \downarrow$	$F_\beta \uparrow$	$S_\alpha \uparrow$	$E \uparrow$	$MAE \downarrow$	$F_\beta \uparrow$	$S_\alpha \uparrow$	$E \uparrow$
RGBD135	0.0288	0.9052	0.8833	0.9339	0.028	0.906	0.905	0.947
DUT-RGBD	0.0308	0.9384	0.9109	0.9598	0.025	0.904	0.919	0.952
LFSD	0.0815	0.8475	0.8192	0.8808	0.073	0.864	0.849	0.877

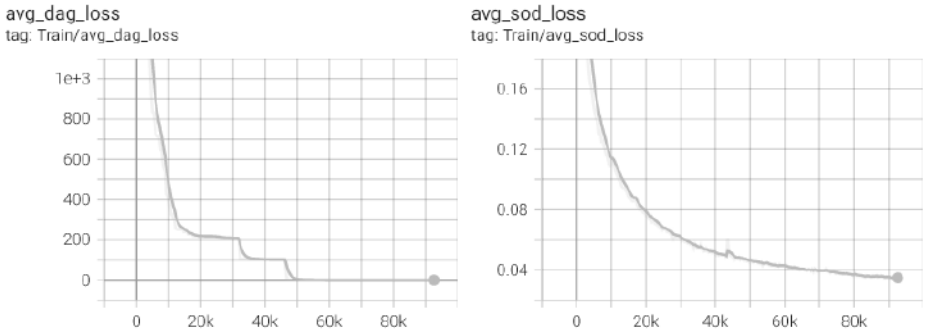


Figure 6.10: Train Loss

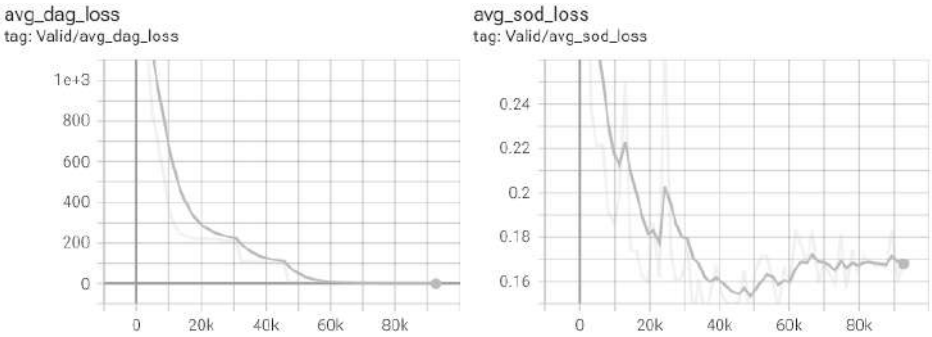
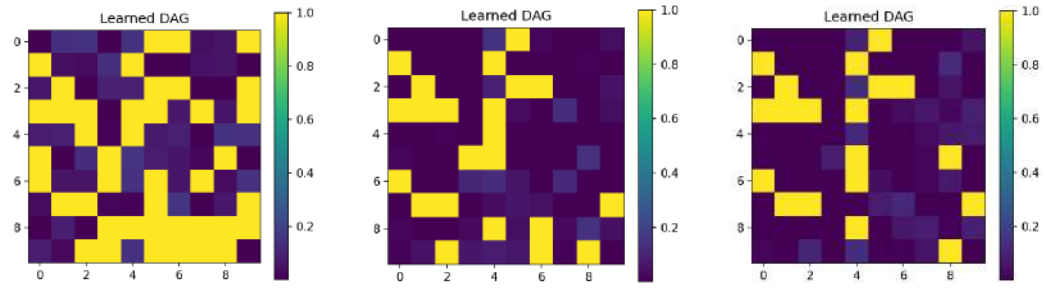
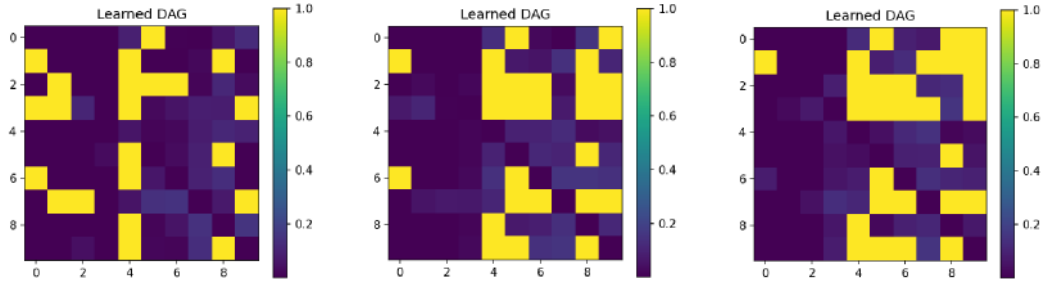


Figure 6.11: Validation Loss

Table 6.6 compares the results of our model and Cas-Gnn on three test datasets. Only the DUT-RGBD dominates and the other datasets do not obtain better results than Cas-Gnn except for the E-measure of LFSD. To analyze this, it can be considered that the optimal model or DAG matrix was not obtained since the training was terminated before the training progressed further to obtain better results. So, if we train enough so that the training does not stop early, then we expect the improvement. Despite these results, our model showed some progress on a specific dataset, so we observe the possibility that our model improves the performance of GNNs through structure learning.



(a) Epoch 19 (b) Epoch 104 (c) Epoch 164



(d) Epoch 189 (e) Epoch 249 (f) Epoch 284

Figure 6.12: The change as the DAG is trained.

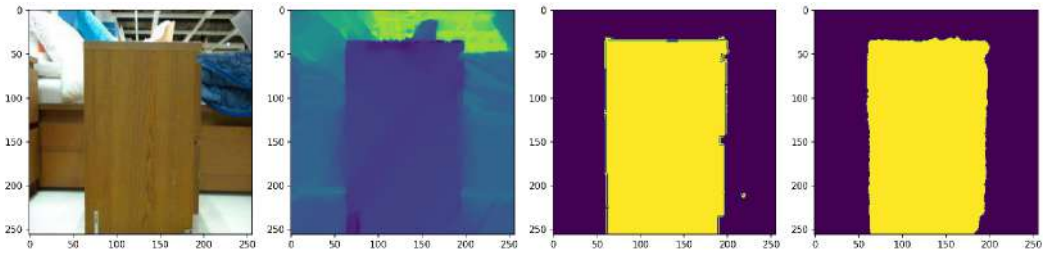


Figure 6.13: Salient Object Detection result (Train)

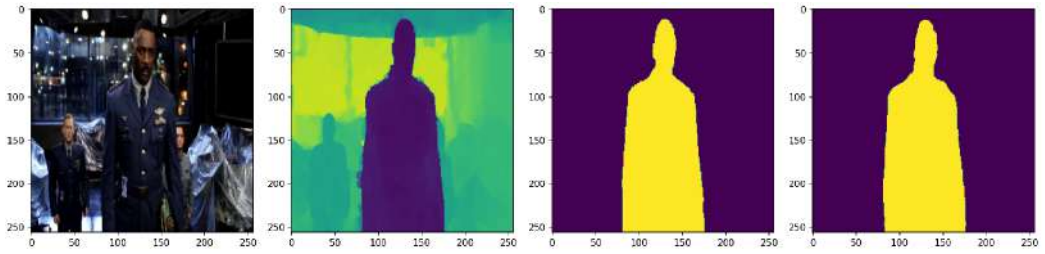


Figure 6.14: Salient Object Detection result (Train)

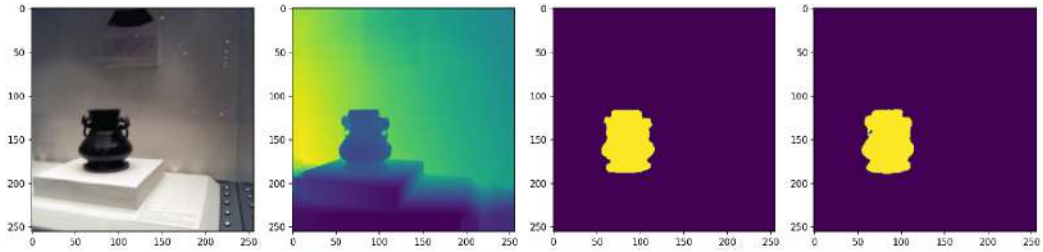


Figure 6.15: Salient Object Detection result (Train)

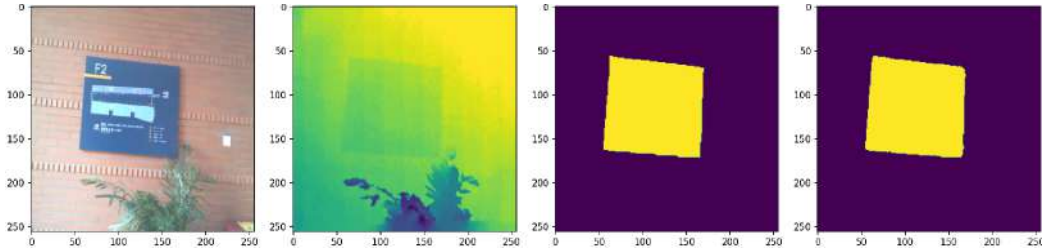


Figure 6.16: Salient Object Detection result (Train)

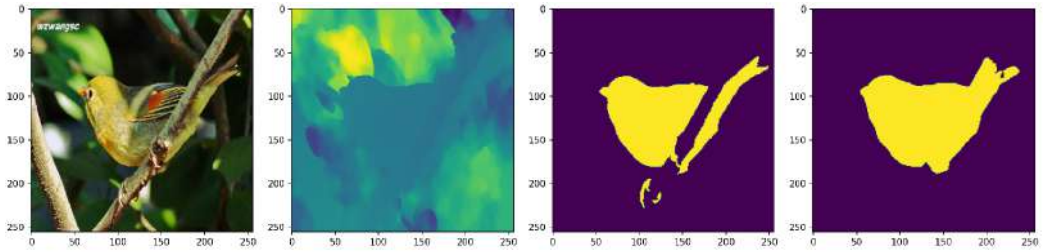


Figure 6.17: Salient Object Detection result (Validation)

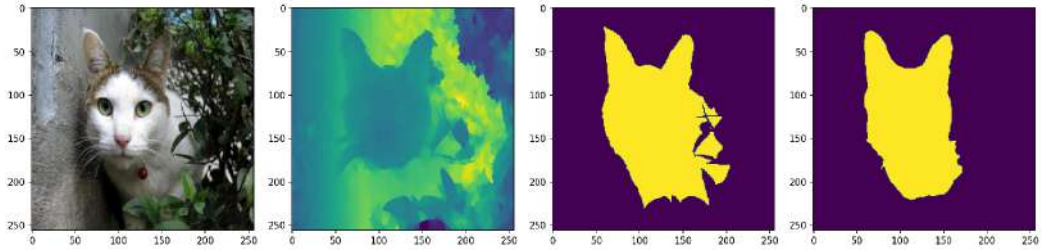


Figure 6.18: Salient Object Detection result (Validation)

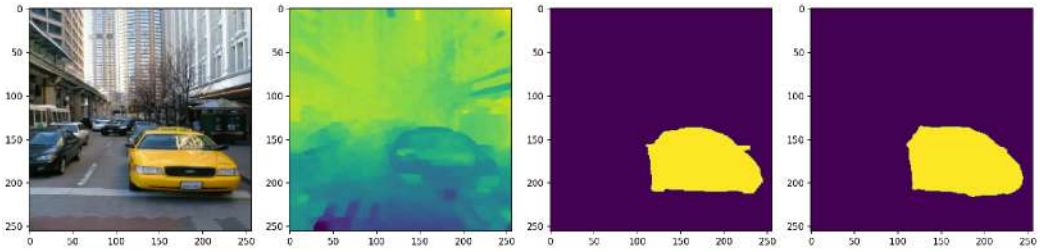


Figure 6.19: Salient Object Detection result (Validation)

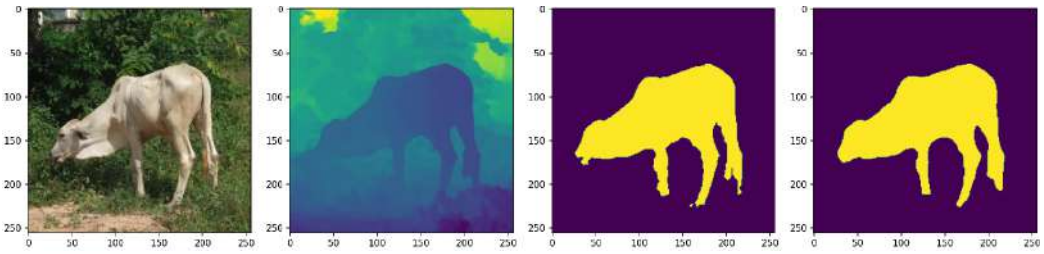


Figure 6.20: Salient Object Detection result (Validation)

6.4 Rain Removal Task

6.4.1 Datasets

Dataset	Number of Images	Description	Real/Synthetic
RainDS	1000	Real rainy images with pairs of no rain images.	Real
DQA	206	Real rainy images.	Real

RainDS/DQA

The RainDS dataset is real data, specifically a pair of ground-truth data that has not rained and data that has rained. This RainDS data was obtained by keeping the camera still, taking pictures, and simulating rain. DQA data is real data and consists of only rainy data.

6.4.2 Experimental Results

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.21: Derained result 1

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.22: Derained result 2

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.23: Derained result 3

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.24: Derained result 4

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.25: Derained result 5

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.26: Derained result 6

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.27: Derained result 7

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.28: Derained result 8

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.29: Derained result 9

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.30: Derained result 10

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.31: Derained result 11

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.32: Derained result 12

The above results in Figures [?] through [?] show that the derained results are qualitatively satisfactory. If we analyze the characteristics of these images, we can observe that they are not images with too heavy rain, but have moderate amount of rain, and the segmentation results are also not bad in the first stage of a proposed method.

However, the results are not good for the subsequent image results, and overall, it seems that the image inpainting itself is difficult due to too heavy rain. In addition, it is thought to be a problem because it does not segment well from the first stage due to the large amount of rain.

In the current experiment, we trained segmentation using only paired rain data (RainDS) and used the pretrained model of LaMa for image inpainting without training. However, in the future work, we expect better results if we train a model that can segment well across domains through frequency analysis between domains, and also train only rainstreak-similar masks other than the inpainting mask used in LaMa.

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.33: Failure case 1

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.34: Failure case 2

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.35: Failure case 3

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.36: Failure case 4

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.37: Failure case 5

(a) Ground-truth (b) Rainy image (c) Derained image

Figure 6.38: Failure case 6

(a) Rainy image

(b) Derained image

Figure 6.39: Failure case 7 (DQA dataset)

(a) Rainy image

(b) Derained image

Figure 6.40: Failure case 8 (DQA dataset)

(a) Rainy image

(b) Derained image

Figure 6.41: Failure case 9 (DQA dataset)

Chapter 7

Conclusion

This dissertation focuses on theoretical analysis and practical performance evaluation of spectral-based graph neural networks. The relationship between the spectra of a graph Laplacian and the convolution operation in graph neural networks is examined. The study investigates the expressive power of graph convolutional models and explores spectral graph convolution methods to achieve comparable expressiveness. The findings demonstrate that spectral-based graph neural networks can effectively handle graph-based tasks.

In addition, this dissertation discusses potential future improvements to enhance their practical performance and how to develop more diverse analysis methods. This discussion relates the fact that the spectral theory of the Laplacian can act as a partial alternative to the Fourier transform when dealing with situations where there is insufficient symmetry to utilize Fourier-analytic methods. Therefore, this might be particularly relevant in scenarios such as manifolds without any translation symmetries, for instance general graphs addressed in Section 4.2.

Furthermore, the study extends the application of these networks to various area such as computer vision tasks and recommendation systems, and compares their expressiveness to existing approaches. Several applications utilizing graphs are presented, including experiments on salient object detection using directed acyclic graphs and rain removal tasks. The empirical

results illustrate the benefits of incorporating graph theory and Fourier analysis knowledge into the model.

Bibliography

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

[2] 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.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[5] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, (New York, NY, USA), p. 165–174, Association for Computing Machinery, 2019.

[6] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford infolab, 1999.

[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[10] P. B. Thorat, R. Goudar, and S. Barve, "Survey on collaborative filtering, content-based filtering and hybrid recommendation system," *International Journal of Computer Applications*, vol. 110, no. 4, pp. 31–36, 2015.

[11] OpenAI, "Gpt-4 technical report," 2023.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision gnn: An image is worth graph of nodes," *arXiv preprint arXiv:2206.00272*, 2022.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, and T. Unterthiner, "Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[15] J. B. Estrach, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR*, vol. 2014, 2014.

[16] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning*, pp. 2702–2711, PMLR, 2016.

[17] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[18] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, PMLR, 06–11 Aug 2017.

[19] C. Merkwirth and T. Lengauer, "Automatic generation of complementary descriptors with molecular graph networks," *Journal of chemical information and modeling*, vol. 45, no. 5, pp. 1159–1168, 2005.

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[21] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, pp. 2014–2023, PMLR, 2016.

[22] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1416–1424, 2018.

[23] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[24] Z. Liu and J. Zhou, "Introduction to graph neural networks," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 2, pp. 1–127, 2020.

[25] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[26] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, R. Koper, F. Ricci, L. Rokach, B. Shapira, and P. Kantor, "Recommender systems handbook," *Recommender Systems in Technology Enhanced Learning*, pp. 387–415, 2011.

[27] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

[28] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[29] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," *Advances in neural information processing systems*, vol. 31, 2018.

[30] Y. Yu, T. Gao, N. Yin, and Q. Ji, "Dags with no curl: An efficient dag structure learning approach," in *International Conference on Machine Learning*, pp. 12156–12166, PMLR, 2021.

[31] D. M. Chickering, "Learning bayesian networks is np-complete," *Learning from data: Artificial intelligence and statistics V*, pp. 121–130, 1996.

[32] Y. Yu, J. Chen, T. Gao, and M. Yu, "Dag-gnn: Dag structure learning with graph neural networks," in *International Conference on Machine Learning*, pp. 7154–7163, PMLR, 2019.

[33] 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.

[34] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix factorization model in collaborative filtering algorithms: A survey," *Procedia Computer Science*, vol. 49, pp. 136–146, 2015.

[35] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE international conference on data mining*, pp. 263–272, Ieee, 2008.

[36] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Computational visual media*, vol. 5, pp. 117–150, 2019.

[37] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

[38] A. Luo, X. Li, F. Yang, Z. Jiao, H. Cheng, and S. Lyu, "Cascade graph neural networks for rgb-d salient object detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pp. 346–364, Springer, 2020.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3937–3946, 2019.

[41] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models," *arXiv preprint arXiv:2201.11793*, 2022.

[42] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2149–2159, 2022.

[43] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022.

[44] I. Makarov and G. Borisenko, "Depth inpainting via vision transformer," in *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 286–291, IEEE, 2021.

[45] Y. Deng, S. Hui, S. Zhou, D. Meng, and J. Wang, "T-former: An efficient transformer for image inpainting," in *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 6559–6568, 2022.

[46] L. Chi, B. Jiang, and Y. Mu, "Fast fourier convolution," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4479–4488, 2020.

[47] S.-H. Yoon, C. Kim, and K.-H. Kim, "Multi-dimensional limiting process for three-dimensional flow physics analyses," *Journal of Computational Physics*, vol. 227, no. 12, pp. 6001–6043, 2008.

[48] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[49] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[50] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[51] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[52] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[53] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, dec 2015.

[54] H. Wang, B. Chen, and W.-J. Li, "Collaborative topic regression with social regularization for tag recommendation," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, p. 2719–2725, AAAI Press, 2013.

[55] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[56] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1661–1670, 2015.

[57] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, (Arlington, Virginia, USA), p. 452–461, AUAI Press, 2009.

[58] R. Ju, L. Ge, W. Geng, T. Ren, and G. Wu, "Depth saliency based on anisotropic center-surround difference," in *2014 IEEE international conference on image processing (ICIP)*, pp. 1115–1119, IEEE, 2014.

[59] H. Peng, B. Li, W. Xiong, W. Hu, and R. Ji, "Rgbd salient object detection: A benchmark and algorithms," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III 13*, pp. 92–109, Springer, 2014.

[60] Y. Cheng, H. Fu, X. Wei, J. Xiao, and X. Cao, "Depth enhanced saliency detection method," in *Proceedings of international conference on internet multimedia computing and service*, pp. 23–27, 2014.

[61] Y. Piao, X. Li, M. Zhang, J. Yu, and H. Lu, "Saliency detection via depth-induced cellular automata on light field," *IEEE Transactions on Image Processing*, vol. 29, pp. 1879–1889, 2019.

[62] N. Li, J. Ye, Y. Ji, H. Ling, and J. Yu, "Saliency detection on light field," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2806–2813, 2014.

[63] D.-P. Fan, C. Gong, Y. Cao, B. Ren, M.-M. Cheng, and A. Borji, "Enhanced-alignment measure for binary foreground map evaluation," *arXiv preprint arXiv:1805.10421*, 2018.

[64] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji, "Structure-measure: A new way to evaluate foreground maps," in *Proceedings of the IEEE international conference on computer vision*, pp. 4548–4557, 2017.

국문초록

본 눈문에서는 스펙트럼 기반 그래프 인공신경망의 이론적 분석과 그 실용적 성능에 대해 다룬다. 그래프 신경망에서의 컨볼루션 연산과 라플라시안 그래프의 스펙트럼 간의 관계를 조사하고, 어떤 가정들 하에서 그래프 컨볼루션이 정의될 수 있으며, 그러한 가정 아래에서 어떤 방식으로 그래프 컨볼루션을 정확히 표현할 수 있는 지에 대해 분석하였다.

이러한 분석을 기반으로 다양한 그래프 컨볼루션을 실험하여 모델의 표현력과 성능을 논의하였다. 결과적으로, 스펙트럼 기반 그래프 신경망이 그래프 기반 작업에서 우수한 성능을 보여줌을 확인하며, 실제 성능을 향상시키기 위한 개선 가능한 부분에 대해 논의한다. 더불어, 이론과 적용 영역을 확장하여 그래프 기반 작업뿐만 아니라 전통적인 컴퓨터 비전 작업 등에도 적용할 수 있음을 보여주어 이러한 방식의 확장성을 보여주었다.

마지막으로 이 논문에서는 그래프를 활용한 몇 가지 응용 사례와 결과를 제시하였다. 구체적인 실험으로 방향성 비순환 그래프(DAG)를 이용한 두드러진 물체 검출 작업에 대한 실험을 수행하였다. 이외에도 푸리에 변환을 활용한 모듈을 활용하여 비를 제거하는 태스크 같은 실용적 분야에 적용한 모델과 실험 결과들을 살펴본다. 이러한 실험들을 통해, 그래프 이론과 푸리에 분석 지식과 같은 수학적 지식을 모델에 통합하고 이를 분석하는 것이 성능 향상에 유용함을 실증적으로 보여주었다.

주요어 : 스펙트럼 기반 그래프 인공 신경망, 그래프 인공 신경망, 협업 필터링, 두드러진 물체 검출, 빗물 제거 작업
학번 : 2016–25263