Theoretical Analysis of Spectral Graph Neural Networks

Fangbing Liu

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

July 2025

I certify that the thesis is my own original work, except where otherwise indicated.

Fangbing Liu
9 July 2025

Acknowledgements

My PhD journey has been an unusually long and challenging one, marked by unexpected obstacles including the COVID-19 pandemic, program leave, travel restrictions that kept me outside Australia, and program extensions. Throughout this difficult period, many people provided the support that made completion possible.

First and foremost, I wish to express my deepest gratitude to my primary supervisor, Qing Wang, for her unwavering support and guidance throughout my candidature. Her passion for research and curiosity were a constant source of inspiration to me. She stood by me through many challenges, offering encouragement during stalled research progress and providing assistance with the administrative processes required to extend my program. I am deeply saddened that she is no longer here to witness the completion of this work, but I know she would have been incredibly proud of this achievement. Her mentorship, kindness, and belief in me will always be remembered. I would also like to thank the members of my supervisory panel for their feedback and support throughout the research.

I wish to express my gratitude to my family and friends, who showed remarkable understanding and compassion throughout this journey. Rather than adding pressure, they encouraged me to prioritize my well-being and offered me the option to quit if the challenges became too overwhelming. Their unconditional support and patience gave me the strength to continue.

I would like to extend my sincere thanks to our HDR team, in particular Kathryn Giri and Timothy Bocquet, for their tireless support throughout my candidature. Their dedication and expertise in navigating scholarship applications, program leave, and the complex procedures required to continue my studies were invaluable. Without their commitment and assistance, I could not have reached this stage of my journey.

This thesis would not have been possible without all of you.

Publications

Primary Publications

The contributions of this thesis have been published in top-tier conferences. Below is a chronological list of these publications:

- Fangbing Liu and Qing Wang. Asymmetric learning for spectral graph neural networks. In Annual AAAI Conference on Artificial Intelligence, 2025.

  The design of the preconditioning method, the theoretical analysis and experimental verifications of this paper are presented in Chapter 4.

- Fangbing Liu and Qing Wang. SpecMat: Adaptive Matrix Generation to Enhance Node Distinguishability of Spectral Graph Neural Networks. Under review.

  The design of the SpecMat module, the theoretical analysis and experimental verifications of this paper are presented in Chapter 5.

- Fangbing Liu and Qing Wang. Unveiling the Roles of Graph Homophily and Polynomial Order in Spectral GNN Generalization. Under review.

  The transductive generalization error bound, the analysis of factors affecting the bound and experimental verifications of this paper are presented in Chapter 6.

Secondary Publications

Three more paper are published. However, contributions from these publications are not presented in this thesis. These papers are:

- Fangbing Liu and Qing Wang. Dynamic chunkwise cnn for distantly supervised relation extraction. In 2020 IEEE International Conference on Big Data (Big Data), pages 738–747. IEEE, 2020.

- Fangbing Liu and Qing Wang. Episode adaptive embedding networks for few-shot learning. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 3–15. Springer, 2021.

- Jingyu Shao, Qing Wang, and Fangbing Liu. Learning to sample: an active learning framework. In 2019 IEEE international conference on data mining (ICDM), pages 538–547. IEEE, 2019.

Abstract

Spectral Graph Neural Networks (GNNs) have demonstrated strong performance in graph node classification tasks, yet factors affecting their performance remains insufficiently understood. We provide theoretical analysis of factors affecting the performance of spectral GNNs in node classification task: the optimization process, the node distinguishability, and the generalization of spectral GNNs.

Optimizing spectral GNNs remains a critical challenge in the field. The underlying processes are not well understood. We investigate the inherent differences between graph convolution parameters and feature transformation parameters in spectral GNNs and their impact on the optimization landscape. Our analysis reveals that these differences contribute to a poorly conditioned problem, resulting in suboptimal performance. To address this issue, we introduce the concept of the block condition number of the Hessian matrix, which characterizes the difficulty of poorly conditioned problems in spectral GNN optimization. We then propose an asymmetric learning approach, dynamically preconditioning gradients during training to alleviate poorly conditioned problems. Theoretically, we demonstrate that asymmetric learning can reduce block condition numbers, facilitating easier optimization. Extensive experiments on eighteen benchmark datasets show that asymmetric learning consistently improves the performance of spectral GNNs for both heterophilic and homophilic graphs. This improvement is especially notable for heterophilic graphs, where the optimization process is generally more complex than for homophilic graphs.

The node distinguishability of spectral GNNs are under explored. We analyze how the interplay between graph matrices and node features influences node distinguishability. We derive a theoretical lower bound for the number of distinguishable nodes, proving that it is determined by two key factors: the number of distinct eigenvalues in the graph matrix and the nonzero frequency components of node features in its eigenbasis. Building on these theoretical insights, we propose SpecMat, an adaptive graph matrix generation module designed to generate graph matrices adapting to different learning tasks. SpecMat can be integrated with any spectral GNN architecture to enhance their node distinguishability without increasing the order of computational complexity. We prove that spectral GNNs augmented with SpecMat preserve permutation equivariance, ensuring that reordering the graph nodes results in a corresponding reordering of the node embeddings. Extensive experiments across eighteen benchmark datasets, validate the effectiveness of SpecMat in enhancing node distinguishability of spectral GNNs.

The generalization properties of spectral GNNs also remain insufficiently understood. To advance the understanding of generalization of spectral GNNs in node classification task, we introduce a stability-based framework for transductive generalization in spectral GNNs. In particular, we derive a uniform transductive stability bound and provide an explicit analysis in a two-class setting, revealing how homophily and heterophily drive generalization. Our findings show that spectral GNNs generalize well on graphs with strong homophily or heterophily but struggle on graphs with weaker structural properties. We also identify conditions under which increasing the polynomial order in spectral GNN architectures may degrade generalization. Empirical evaluations on synthetic and real-world datasets corroborate these theoretical insights, underscoring the important role of structural properties and architectural design in spectral GNN performance.

Uniform transductive stability fail to provide non-vacuous bounds due to the accumulation of perturbations during iterative training. To derive a non-vacuous generalization error bound, we introduces a novel PAC-Bayes framework specifically designed for spectral GNNs in node classification tasks. Firstly, we extend PAC-Bayes theory from its conventional inductive setting to the transductive setting, capturing dependency between training and testing nodes on graph. Secondly, we propose a parameter-specific prior-posterior assignment method to take the distinction between graph convolution parameter Θ and feature transformation parameter W into account. Then, we use a tailored PAC-Bayes optimization framework to minimize empirical loss and generalization error simultaneously. Extensive experiments on real-world graph datasets show that transductive PAC-Bayes bounds are much tighter than those derived from uniform transductive stability.

Contents

List of Figures

List of Tables

Notation and Terminology

Notations	Description
G	Graph
$\mathcal{V}$	Node set
$\mathcal{E}$	Edge set
Θ	Graph convolution parameter
W	Feature transformation parameter
M	Graph matrix
X	Node features
Y	Node labels
$\hat{Y}$	Predicted node labels
$\hat{L}$	Normalized Laplacian matrix
$\tilde{L}$	Shifted normalized Laplacian matrix
$\tilde{A}$	Normalized adjacency matrix
$\tilde{A}'$	Normalized adjacency matrix with self-loops
A	Adjacency matrix
D	Degree matrix
n	Node number on graph
$\mathbf{H}$	Hessian matrix
H_{edge}	Edge homophilic ratio
H_{node}	Node homophilic ratio
H_G	Graph homophilic ratio
$\mathcal{N}_k$	k-hop node set
x_i^l	Representation of i-th node in the l-th layer
α_{ij}^l	Aggregate coefficient from node j to i in l-layer
$\|\cdot\|_F$	Frobenius norm
$\|\cdot\|_2$	ℓ_2 norm
$\|\cdot\|_\infty$	Infinity norm
$J_k^{a,b}$	Jacobi polynomial basis
g_t	Gradient in the t-th iteration
m_t	Momentum in the t-th iteration
τ^t	Parameter in the t-th iteration
$VC(f)$	VC dimension of function f
$\mathcal{R}_S(\mathcal{F})$	Inductive Rademarcher complexity

$\mathcal{R}_{\mathcal{S}+\mathcal{U},p}(\mathcal{F})$	Transductive Rademarcher complexity
$US(f)$	Uniform stability of f
$US_{\mathcal{S}+\mathcal{U}}(f)$	Uniform transductive stability of f
$\mathcal{L}_{D_u}$	Expected loss
$\mathcal{L}_{S_m}$	Empirical loss
ℓ_i	Loss for node i
KL	KL-divergence
P	Prior distribution
Q	Posterior distribution
π	Permutation on graph
spec	Spectrum of matrix
d_M	Distinct eigenvalue number
$\|\tilde{X}^{(M)}\|_0$	Frequency component number of X in eigenbasis of M
λ_i	i-th eigenvalue
$\mathbf{\Omega}$	SpecMat
$Lip(\cdot)$	Lipschitz constant
$Smt(\cdot)$	Smoothness
γ	Uniform stability of spectral GNNs
β	Gradient norm
Tr	Trace of matrix
$\lambda_{\Theta,k}$	Variance of prior of θ_k
λ_W	Variance of prior of W

Chapter 1

Introduction

Graph Neural Networks (GNNs) have emerged as powerful tools for learning on graph-structured data, demonstrating state-of-the-art performance in tasks such as node classification and graph classification [Corso et al., 2024, Wu et al., 2019b, Xu et al., 2019]. Among the various types of GNNs, spectral GNNs transform graph-structured data into the spectral domain, where graph convolution functions are applied to process information for downstream tasks. Despite the empirical success of spectral GNNs, particularly in node classification tasks, their theoretical foundations remain underexplored. Key aspects such as optimization dynamics, node distinguishability, and generalization ability of spectral GNNs lack a rigorous theoretical understanding. This thesis aims to bridge this gap by systematically investigating these fundamental properties of spectral GNNs.

1.1 Background

Graph-structured data is ubiquitous across various domains, such as social networks, recommendation systems, and drug discovery [Sharma et al., 2024, Zhou et al., 2020]. From modeling human connections in social networks to mapping protein-protein interactions in biology, graphs provide a powerful way to capture complex relationships underlying various computational problems. Their inherent structure, where nodes represent entities and edges encode relationships, makes them a natural choice for modeling interconnected data. In drug discovery, GNNs can predict molecular properties by learning over chemical graph structures. In social network analysis, they enable community detection and influence propagation modeling. In finance, GNNs can capture complex relations between users and transactions and enable abnormal detection. Thus, graph representation learning has become a fundamental approach for analyzing real-world datasets, enabling more effective insights and predictions across multiple fields [Hamilton, 2020].

The growing need to effectively learn from graph-structured data has driven rapid advancements in machine learning techniques specifically designed for graphs. Tra-

ditional machine learning methods, which are built for grid-like structures such as images or sequential data like text, struggle to capture the unique structural properties of graphs. These include permutation equivariance, multi-hop dependencies, and hierarchical features [Abu-El-Haija et al., 2019, Satorras et al., 2021]. These limitations, combined with the increasing availability of graph data and the rising complexity of graph-based applications, have fueled research efforts toward developing more effective learning frameworks for graphs [Ortega et al., 2018, Wu et al., 2022]. As a result, Graph Neural Networks (GNNs) have emerged as a powerful approach, enabling the effective modeling of both structural and node features during the learning process.

GNNs extend machine learning to non-Euclidean data, allowing models to effectively learn from complex graph structures where relationships do not conform to regular grids. Early GNN models were based on recurrent neural networks (RNNs) operating on graph structures, they suffered from inefficiencies and gradient vanishing issues [Li et al., 2016]. These challenges led to the development of more effective message-passing GNNs, which iteratively aggregate information from local neighborhoods to learn node representations. By leveraging both node features and graph connectivity, these models capture structural dependencies more effectively [Kipf and Welling, 2017, Scarselli et al., 2009]. To obtain a graph-level representation, a pooling function is applied to the node embeddings, characterizing the overall structure of the graph.

GNNs can generally be categorized into two main types: *spatial GNNs* and *spectral GNNs*. Spatial GNNs, such as GraphSAGE [Hamilton et al., 2017] and Graph Attention Networks (GAT) [Velickovic et al., 2018], define convolution operations directly in the vertex domain, relying on neighborhood aggregation. On the contrary, spectral GNNs transform graph-structured data into the spectral domain by leveraging spectral graph theory and the eigendecomposition of the graph matrix [Bruna et al., 2013]. Graph convolution is performed in the spectral domain using graph filters, and the transformed data is then projected back to the vertex domain. This spectral perspective offers a mathematically grounded understanding of GNNs, linking them to well-established theories in graph signal processing [Shuman et al., 2013]. Unlike spatial GNNs, which aggregate information from neighboring nodes based on learned functions, spectral methods explicitly design graph filters based on eigenvalues and eigenvectors. This approach establishes a theoretical connection between graph convolutions and graph structure, capturing both local and global graph properties through the lens of graph frequencies [Defferrard et al., 2016].

Building on this spectral perspective, graph signal processing (GSP) provides a theoretical framework for analyzing GNNs and their properties. First, GSP enables a precise understanding of how GNNs process and transform node features by examining frequency components in graph-structured data. From this viewpoint, graph convolution functions in spectral GNNs act as graph filters, designed to selectively capture

both local and global structural information. Furthermore, GSP offers valuable insights into the expressive power of spectral GNNs, allowing researchers to analyze their behavior under different graph convolution operations [He et al., 2021, Wang and Zhang, 2022]. This theoretical foundation has proven particularly useful for analyzing critical properties of GNNs, such as stability, generalization, and expressiveness, helping to enhance their design for a wide range of graph-based applications.

Generally, state-of-the-art spectral GNNs can be expressed in the following form:

$$\Psi(M, X) = g_\Theta(M) f_W(X), \tag{1.1}$$

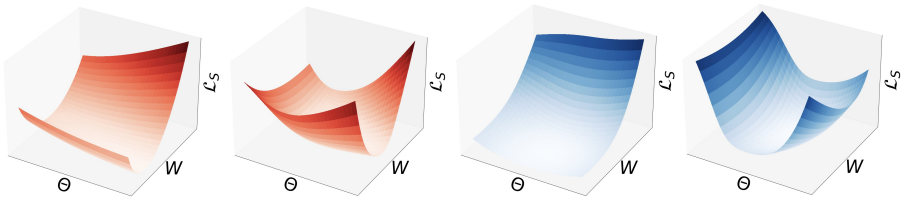
where $M \in \mathbb{R}^{n\times n}$ represents the graph matrix (such as the Laplacian or adjacency matrix), $X \in \mathbb{R}^{n\times h}$ denotes the node feature matrix, $g_\Theta(M) = \sum_{k=0}^{K} \theta_k T_k(M)$ is the graph convolution function parameterized by $\Theta = \{\theta_k\}_{k=0}^{K}$, and $T_k(\cdot)$ denotes the k-th polynomial basis. The term $f_W(X)$ represents the feature transformation function parameterized by W. Spectral GNNs learn meaningful node features by optimizing W, projecting them into the spectral domain. By adjusting Θ, spectral GNNs filter out unnecessary information and enhance useful information for downstream tasks.

Despite the strong empirical performance of spectral GNNs, the underlying theoretical factors driving their success remain under-explored. Key aspects such as the optimization dynamics, node distinguishability, and generalization ability require further investigation to fully understand the effectiveness of these models.

1.1.1 Optimization

In spectral GNNs, notably, the parameters Θ and W play distinct roles in modeling and also significantly differ in dimensions. Specifically, the dimension of Θ is fewer than 20 while W has thousands of dimensions. During the training of spectral GNNs, different learning rates for parameters Θ and W are often adopted to achieve good performance. Interestingly, we observe that the contours of the loss function undergo stretching or compression along specific directions during training. In stretched directions, small alterations in parameters lead to significant changes in the loss function. Figure 1.1 shows the loss landscapes of a spectral GNN model, ChebNet [Defferrard et al., 2016]. There always exists a stretched direction for Θ or W during training, indicating that parameters Θ and W exhibit different sensitivities to learning rates and gradient updates.

The distorted landscapes of spectral GNNs can be further explained from the view of Hessian matrix. According to [Granziol et al., 2020, LeCun et al., 1992], neural networks exhibit non-convex characteristics and the optimal learning rate of a parameter is inversely proportional to the largest eigenvalue of the Hessian matrix. The Hessian matrix $\mathbf{H}$ of a spectral GNN contains second partial derivatives of Θ and W, which



(a) Texas-50 (b) Texas-200 (c) Cora-50 (d) Cora-200

Figure 1.1: The landscapes of ChebNet on Texas and Cora at iterations 50 and 200. The x, y, z-axes represent the directions of Θ, W, and the empirical loss $\mathcal{L}_S$, respectively. The landscapes change during training, and the stretched direction of the parameters Θ and W also shifts.

can be divided into two diagonal block matrices $\mathbf{H}_{\Theta,\Theta}$ and $\mathbf{H}_{W,W}$, and two off-diagonal block matrices. In scenarios where Θ and W exhibit minimal interference, the optimal learning rates for Θ and W correspond inversely to the largest eigenvalues of their respective diagonal block Hessian matrices. The largest eigenvalue of a Hessian matrix indicates the maximum rate of change of the gradient with respect to a small change of its corresponding parameter. In regions where the largest eigenvalue of a Hessian matrix is large, the gradient norm tends to be large as well, indicting a large loss change with respect to a small parameter change [Bonnans et al., 2006]. The larger the gap between two diagonal block matrices of Hessian with respect to Θ and W respectively, the more distorted of the local landscape. Figure 1.2 shows the ratio of the largest eigenvalues of two diagonal block Hessian matrices (largest eigenvalue ratio) when a spectral GNN is applied on different graphs. There is a notable disparity in the largest eigenvalue ratio across different graphs. When a graph is more homophilic, the largest eigenvalue ratio is closer to one, indicting a less distorted landscape. For example, the landscapes of Texas are more distorted than those of Cora because Texas is a heterophilic graph while Cora is a homophilic graph.

The distorted landscapes of spectral GNNs can be further explained from the view of Hessian matrix. According to [Granziol et al., 2020, LeCun et al., 1992], neural networks exhibits non-convex characteristics and the optimal learning rate of a parameter is inversely proportional to the largest eigenvalue of the Hessian matrix. The Hessian matrix $\mathbf{H}$ of a spectral GNN contains second partial derivatives of Θ and W, which can be divided into two diagonal block matrices $\mathbf{H}_{\Theta,\Theta}$ and $\mathbf{H}_{W,W}$, and two off-diagonal block matrices $\mathbf{H}_{\Theta,W}$ and $\mathbf{H}_{W,\Theta}$. In scenarios where Θ and W exhibit minimal interference, the optimal learning rates for Θ and W correspond inversely to the largest eigenvalues of their respective diagonal block Hessian matrices. The largest eigenvalue of a Hessian matrix indicates the maximum rate of change of the gradient with respect to a small change of its corresponding parameter. In regions where the largest eigenvalue of a Hessian matrix is large, the gradient norm tends to be large as well, indicting a large loss change with respect to a small parameter change [Bonnans et al., 2006]. The larger the gap between two diagonal block matrices of Hessian with respect to Θ and

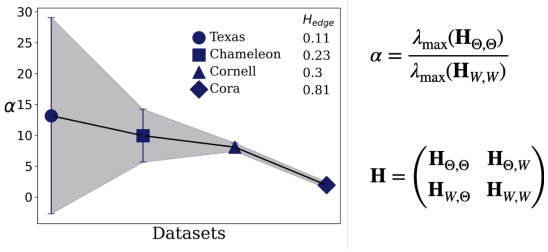


Figure 1.2: The largest eigenvalue ratio α of ChebNet when applied to graphs of varying edge homophily ratios H_{edge} [Zhu et al., 2020b] for node classification. $\mathbf{H}$ represents the Hessian matrix and variances are depicted as grey shades.

W respectively, the more distorted of the local landscape. Figure 1.2 shows the ratio of the largest eigenvalues of two diagonal block Hessian matrices (largest eigenvalue ratio) when a spectral GNN is applied on different graphs. There is a notable disparity in the largest eigenvalue ratio across different graphs. When a graph is more homophilic, the largest eigenvalue ratio is closer to one, indicting a less distorted landscape. For example, the landscapes of Texas are more distorted than those of Cora because Texas is a heterophilic graph while Cora is a homophilic graph.

To understand why the distorted landscape exists in the optimization of spectral GNNs and its unique characteristics compared to other neural networks, we conduct a theoretical analysis of its optimization process in this thesis.

1.1.2 Node Distinguishability

While the optimization of spectral GNNs plays a crucial role in training stability and classification performance, another fundamental aspect affecting performance is node distinguishability. Spectral GNNs rely on choices of graph matrices (e.g., adjacency matrix, Laplacian matrix) and the distribution of frequency components of node features in the eigenspace of graph matrices.

Different spectral GNNs employ different graph matrices, such as the normalized adjacency matrix and the normalized Laplacian matrix. Moreover, within a specific spectral GNN, the distribution of node features on the graph significantly impacts performance [He et al., 2022b, Platonov et al., 2023a]. However, to the best of our knowledge, no prior work has systematically examined how the interaction between the choice of graph matrix and the spectral distribution of node features affects node distinguishability in spectral GNNs.

Based on Eq. (1.1), which illustrates how spectral GNNs process node features via graph convolution, two key questions naturally arise:

(1) How does the interaction between the graph matrix M and the projected node features in the spectral domain influence node distinguishability of spectral GNNs?

(2) How can we construct a graph matrix for spectral GNNs which is well-suited to a given graph?

By exploring these questions in this thesis, we aim to contribute to a deeper understanding of node distinguishability in spectral GNNs and inform the development of more effective graph-based learning models.

1.1.3 Generalization

While understanding node distinguishability provides valuable insights into how spectral GNNs process graph information, another critical challenge remains: *how well do these models generalize to unseen data?* Generalization is a cornerstone of machine learning, crucial for both understanding theoretical limits and optimizing practical performance.

In the literature, researchers have proposed various measures for generalization, such as the VC dimension [Cherkassky et al., 1999], PAC-Bayes bounds [McAllester, 1998], Rademacher complexity [Bartlett and Mendelson, 2002], and algorithm stability [Bousquet and Elisseeff, 2002], to gauge how model complexity and training data influence generalization. However, applying these measures to Graph Neural Networks (GNNs) is challenging due to varying graph structures and inherent node dependencies. Understanding GNN generalization remains an open and critical research question.

In light of these complexities, previous studies on GNN generalization have largely centered on two tasks: graph classification and node classification. In graph classification, each graph is typically treated as an i.i.d. sample, allowing for well-established theoretical tools such as VC dimension [Franks et al., 2024, Morris et al., 2023], PAC-Bayes bounds [Behboodi et al., 2022, Ju et al., 2023, Liao et al., 2021], Rademacher complexity Garg et al. [2020], Maskey et al. [2022], and margin-based analyses [Li et al., 2025] to quantify model capacity and establish generalization guarantees under various learning assumptions. Unlike graph classification, node classification operates on a single, partially labeled graph where labeled and unlabeled nodes are interdependent. These dependencies influence predictions, making the transductive setting essential for leveraging the graph structure to improve classification accuracy. To address this, transductive generalization methods such as transductive Rademacher complexity [Esser et al., 2021, Oono and Suzuki, 2020, Tang and Liu, 2023b] and uniform transductive stability [Cong et al., 2021] have been developed. While previous studies have explored factors like graph size, training set size, graph matrix norms, and node features, the impact of graph homophily (the tendency for connected nodes to share labels) remains largely unexplored. Further, the influence of architectural choices beyond GNN depth is still underexamined.

While Spectral GNNs with higher polynomial orders K can theoretically approximate any desired frequency response with negligible error [Powell, 1981], empirical studies have shown that simply increasing K does not guarantee better results. This observation raises a critical question: *How does polynomial order affect the generalization of spectral GNNs?* Beyond polynomial order, another key aspect is *graph homophily*. Because node labels are closely tied to graph connectivity, *how does homophily (or lack thereof) influence the generalization capabilities of spectral GNNs?* These two questions – regarding polynomial order and graph homophily – are central to understanding how spectral GNNs adapt to a variety of graph structures.

1.1.4 Non-vacuous Generalization Error Bound

Although transductive Rademacher complexity and uniform transductive stability are widely used to evaluate the generalization ability of GNNs, these approaches often yield vacuous bounds [Neyshabur et al., 2017]. Rademacher complexity depends on the size of hypothesis space. A large hypothesis space leads to a loose bound. For spectral GNNs, Rademacher complexity scales with the number of parameters where the dimension of feature transformation parameter W are usually more than 10,000. This leads to extremely loose bounds. While the generalization error bound derived with uniform stability does not depend on the hypothesis space, instead it depends on the stability of the algorithm and thus depend on the iteration number. A small perturbation at the beginning can be amplified to a large value after iterations and the bound can be large. Usually hundreds of iterations are needed to train spectral GNNs. Therefore, generalization error bounds derived with uniform transductive stability increases with the iteration number and are vacuous bounds.

On the contrary, PAC-Bayes methods target the specific region of the parameter space that is actually explored during training, rather than attempting to bound the entire hypothesis space. The central idea is to establish a prior distribution over the parameters before training and then examine how this prior is updated to a posterior distribution through the training process. By carefully selecting the prior and formulating an optimization objective that minimizes both the empirical error and the divergence between the prior and posterior, PAC-Bayes methods can produce bounds that more accurately reflect the true generalization behavior of the network [Dziugaite and Roy, 2017a].

Notably, most research on PAC-Bayes bounds for GNNs has focused on graph classification tasks within an inductive setting. Extending these results to transductive settings is not trivial because the training process can be influenced by test node features and the underlying graph structure. Further, due to the special architecture of spectral GNNs, graph convolution parameter Θ and feature transformation parameter W contribute differently to the divergence between prior and posterior distribution. Thus,

they cannot be treated in the same way and different priors are needed.

1.2 Research Questions and Objectives

Spectral GNNs have achieved remarkable empirical success, yet key theoretical and practical challenges still limit their robustness and overall performance. This thesis addresses these challenges by investigating the following research questions:

- **RQ1:** *How does the optimization of spectral GNNs impact training stability and performance?*
  - ◇ How does the structure of the Hessian matrix influence the optimization landscape of spectral GNNs?
  - ◇ What role do the differences between Θ and W play in the optimization process?
  - ◇ How can we develop training strategies that enhance stability and lead to superior solutions?
- **RQ2:** *How does the interaction between the graph matrix and node features influence node distinguishability in spectral GNNs?*
  - ◇ How do different choices of graph matrices (e.g., adjacency vs. Laplacian) affect node distinguishability?
  - ◇ Can we design a novel architecture that incorporates a matrix generation module to adapt to various graphs, thereby enhancing node distinguishability and classification performance?
- **RQ3:** *What key factors influence the generalization ability of spectral GNNs?*
  - ◇ How does graph homophily influence the generalization ability of spectral GNNs?
  - ◇ What is the relationship between the polynomial order K and generalization performance?
  - ◇ What practical guidelines can be established for designing spectral GNNs?
- **RQ4:** *How can we derive non-vacuous PAC-Bayes generalization bounds for spectral GNNs?*
  - ◇ How can PAC-Bayes theory be extended to spectral GNNs in a transductive learning setting?

- ◇ How can we construct parameter-specific priors that account for the distinct roles of the graph convolution parameters Θ and feature transformation parameters W?
- ◇ How can we determine optimal priors and posteriors for Θ and W to achieve tighter generalization bounds?

The primary goal of this thesis is to advance theoretical understanding of spectral GNNs by analyzing their optimization dynamics, node distinguishability, and generalization properties. Specifically, this research aims to answer fundamental questions about why and how spectral GNNs achieve high performance, and to derive rigorous theoretical guarantees that can guide future model design. To achieve this, our key research objectives are as follows:

- **Preconditioning for Optimization**
  - ◇ To investigate how the graph convolution parameter Θ and feature transformation parameter W affect the optimization of spectral GNNs.
  - ◇ To develop a metric to estimate the difficulty of the optimization problem in spectral GNNs.
  - ◇ To design a preconditioning method that alleviates optimization challenges and leads to superior solutions.
- **Enhancing Node Distinguishability**
  - ◇ To theoretically analyze how the interplay between the graph matrix and node features affects node distinguishability.
  - ◇ To derive a bound on the number of nodes that spectral GNNs can distinguish.
  - ◇ To design a novel spectral GNN architecture that enhances node distinguishability for improved performance.
- **Analyzing Factors Influencing Generalization**
  - ◇ To investigate how various architectural choices, such as the polynomial order K, impact the generalization performance of spectral GNNs.
  - ◇ To analyze the influence of graph homophily on generalization.
  - ◇ To develop theoretical guidelines to inform the design of spectral GNNs with robust generalization capabilities.
- **Deriving Non-vacuous Generalization Error Bound**

- ◇ To extend the PAC-Bayes bound from the inductive learning setting to the transductive setting for node classification.
- ◇ To develop a novel prior-posterior assignment method that accounts for the distinct roles of the graph convolution parameters Θ and feature transformation parameter W.
- ◇ To derive non-vacuous, tighter generalization error bounds for spectral GNNs.

1.3 Contributions

This thesis makes several key contributions to our theoretical understanding and practical performance of spectral GNNs. First, we analyze the architectural components of spectral GNNs by distinguishing between the graph convolution parameters Θ and feature transformation parameter W, and we investigate how these differences impact optimization. Next, we study the interaction between the graph matrix and node features to understand its effect on node distinguishability. We then examine how architectural choices (such as polynomial order) and graph data properties (like homophily) influence the generalization performance of spectral GNNs. Finally, we bridge theory and practice by developing a non-vacuous generalization bound using PAC-Bayes theory, providing a rigorous framework to characterize the generalization behavior of spectral GNNs. We elaborate these contributions in the following.

- We investigate the inherent differences between graph convolution parameters and feature transformation parameters in spectral GNNs and their impact on the optimization landscape. Our analysis reveals that these differences contribute to a poorly conditioned problem, resulting in suboptimal performance. To address this issue, we introduce the concept of the block condition number of the Hessian matrix, which characterizes the difficulty of poorly conditioned problems in spectral GNN optimization. We then propose an asymmetric learning approach, dynamically preconditioning gradients during training to alleviate poorly conditioned problems. Theoretically, we demonstrate that asymmetric learning can reduce block condition numbers, facilitating easier optimization.
- We analyze how the interplay between graph matrices and node features influences node distinguishability. We derive a theoretical lower bound for the number of distinguishable nodes, proving that it is determined by two key factors: the number of distinct eigenvalues in the graph matrix and the number of nonzero frequency components of node features in its eigenbasis. Building on these theoretical insights, we propose SpecMat, an adaptive graph matrix generation module designed to generate graph matrices adapting to different learning

tasks. SpecMat can be integrated with any spectral GNN architecture to enhance their node distinguishability. We prove that spectral GNNs augmented with SpecMat preserve permutation equivariance, ensuring that reordering the graph nodes results in a corresponding reordering of the node embeddings.

- We introduce a stability-based framework for transductive generalization in spectral GNNs. We analyze the uniform transductive stability of spectral GNNs, exploring how small perturbations in training data impact model stability through Lipschitz continuity, smoothness, and gradient norm bounds. A theoretical generalization error bound is established, demonstrating that improved stability leads to stronger generalization. A precise gradient norm bound is derived for the two-class case. Generalization insights are provided that spectral GNNs generalize well on graphs that are either strongly homophilic or heterophilic but struggle on moderate cases. Increasing polynomial order will degrade generalization under certain conditions, highlighting key trade-offs in model design.

- We extend classic PAC-Bayes theory to spectral GNNs for node classification in the transductive setting, which is a multi-class problem with cross-entropy loss. To account for the distinct roles of the graph convolution parameters Θ and feature transformation parameters W, we assign them different priors. We also apply the union bound theorem to avoid relying on a single optimal prior. Finally, by employing the PAC-Bayes optimization framework to simultaneously minimize the empirical loss and divergence terms, we obtain a non-vacuous generalization bound.

1.4 Thesis Outline

The reminder of the thesis is structured as follows:

- Chapter 2 (Graph Neural Networks): We provide a comprehensive review of existing research on Graph Neural Networks, focusing on two primary categories: spatial GNNs and spectral GNNs.

- Chapter 3 (Literature Review): We discuss adaptive optimizers and preconditioning methods that address the challenges posed by poorly conditioned optimization problems. Additionally, we review work on the expressive power of spectral GNNs and introduce key methods for generalization analysis, including classical techniques (e.g., VC dimension, Rademacher complexity, uniform stability, and PAC-Bayes theory) as well as recent approaches based on augmentation and representation distributions. We also summarize studies on the generalization analysis of GNNs for both graph and node classification.

- Chapter 4 – Chapter 7 (Main Contributions): Each chapter follows a structured format, including an overview of the research problem, necessary preliminaries, proposed solutions with key findings, experimental results, and a summary of contributions.

- Chapter 4 (Optimization Challenges and Preconditioning): We analyze the optimization challenges in spectral GNNs and demonstrate that the problem is poorly conditioned, as indicated by the large block condition number of the Hessian matrix. To address this issue, we introduce an asymmetric preconditioner that can be integrated with existing optimizers to improve both convergence and solution quality.

- Chapter 5 (Node Distinguishability in Spectral GNNs): We establish a lower bound on node distinguishability in spectral GNNs, which depends on both the graph matrix and node features. Building on this analysis, we propose *SpecMat*, a module that dynamically generates the graph matrix to adapt to different graph structures, thereby enhancing node distinguishability.

- Chapter 6 (Generalization Analysis and Stability): We investigate the generalization properties of spectral GNNs using uniform transductive stability and the contextual stochastic block model (CSBM). Our analysis explores the relationships between stability, generalization error bounds, graph homophily, and the polynomial order of spectral GNNs.

- Chapter 7 (PAC-Bayes Generalization Bound): We extend the classic PAC-Bayes bound to the multi-class classification setting with cross-entropy loss in a transductive learning scenario. By employing the PAC-Bayes optimization framework to jointly minimize the empirical loss and divergence terms, we obtain a non-vacuous generalization bound for spectral GNNs.

- Chapter 8 (Conclusion and Future Directions): We enhance the understanding of the unique properties of spectral GNNs through theoretical analysis. Further, we highlight existing open challenges in spectral GNNs and propose interesting directions for future research.

Chapter 2

Graph Neural Networks

In this chapter, we first discuss graph homophily, an important property influencing GNN performance, and metrics to evaluate the graph homophily. Next, we review different types of GNN architectures, including spatial GNNs and spectral GNNs, highlighting their advantages and limitations.

Graphs are structured data and nodes on graphs also have features and labels. According to the label distribution, graphs are divided into homophilic and heterophilic graphs. Graphs meeting the assumption that nodes of same labels are connected are homophilic graphs. Graphs that nodes of different labels are linked are heterophilic graphs. In real word, both kinds of graphs are widely exist. For example, graphs of academic publications whose nodes are publications, labels are their research fields and edges represent citation relationship, such as Cora, Pubmed and so on. On the other hand, in a dating network, nodes are users, labels are their genders and edges represent dating relationship, we find that most users tend to connect with others of opposite genders. Graphs extracted from dating network are heterophilic graphs. Another example is the protein network where amino acids are nodes, labels are amino acids types and edges represent the connectivity. Usually, different types of amino acids are linked. We need a metric to evaluate the hetrophilic ratio of a graph. There are two types of heterophilic ratio: node homophilic ratio and edge homphilic ratio [Lim et al., 2021, Pei et al., 2020, Zhu et al., 2020b]. For a graph $G = (\mathcal{V}, \mathcal{E}, X)$ where $\mathcal{V}, \mathcal{E}$ are nodes set and edge set separately, X are node features, which is associated with node labels Y. We list several metrics are commonly used to estimate whether a graph is a homophilic graph or a heterophilic graph.

Node Homophilic Ratio [Pei et al., 2020]: It is the ratio of neighbors sharing same label with the ego node and all neighbors, then average across all nodes on graph:

$$H_{node} = \frac{1}{|V|} \sum_{v_i \in \mathcal{V}} \frac{|\{e_{ij} | v_i, v_j \in \mathcal{V}, e_{ij} \in \mathcal{E}, y_i = y_j\}|}{d_i} \tag{2.1}$$

Edge Homophilic Ratio [Zhu et al., 2020b]: It is the proportion that edges connected

nodes of same labels and the total edge number:

$$H_{edge} = \frac{|\{e_{ij}|v_i, v_j \in \mathcal{V}, e_{ij} \in \mathcal{E}, y_i = y_j\}|}{|\mathcal{E}|} \tag{2.2}$$

Both node homophilic ratio and edge homopholic ratio depend on the node label class number and the nodes number in each class. If node label class are highly imbalance, then above two metrics will be screwed. Another graph homophobic metric is the *Graph Homopholic Ratio [Lim et al., 2021]*:

$$\begin{aligned} H_G &= \frac{1}{T-1}\sum_{t=1}^{T}[h_t - \frac{|C_t|}{|\mathcal{V}|}]_+ \\ h_t &= \frac{\sum_{i\in C_t}|\{e_{ij}|v_i, v_j \in \mathcal{V}, e_{ij} \in \mathcal{E}, y_i = y_j\}|}{\sum_{i\in C_t} d_i} \end{aligned} \tag{2.3}$$

where $[x]_+ = \max(x, 0)$, $C_t = \{v_i|y_i = t\}$ and T is the total classes in the graph.

The range of metric $H_{node}, H_{edge}, H_G \in [0, 1]$. When a graph is a homophilic graph, above three metrics will be small (approximating 0) and three metrics will be large on heterophilic graphs (approximating 1).

2.1 Spatial Graph Neural Networks

A foundational work for GNNs is [Gori et al., 2005], which treated GNNs as a recursive neural network that process graph inputs, establishing the basic concept of information propagation between nodes. Building upon this, the first general architecture for GNNs was introduced by Scarselli et al. [2009], where the idea of recursive neighborhood aggregation was proposed. This work established the foundation for future GNN architectures by modeling graph nodes as representations updated through iterative message passing. A significant breakthrough came with the introduction of Graph Convolutional Networks (GCN) [Kipf and Welling, 2017], which simplified previous spectral approaches into an efficient spatial aggregation operation. Spectral graph convolution functions were approximated using Chebyshev polynomial basis. In each layer, node representations were updated by by aggregating features from their neighbors. To handle large-scale graphs, GraphSAGE introduced an inductive learning approach that samples neighbors instead of aggregating information from all neighbors [Hamilton et al., 2017]. It also introduced learnable aggregation functions to maintain strong performance while being scalable to large graphs. The Graph Attention Network (GAT) introduced attention mechanisms to weight neighbors' contributions [Velickovic et al., 2018]. Only the most relevant neighbors are focused. Meanwhile, Graph Isomorphism Network (GIN) derived a theoretical upper bound of ex-

pressiveness of GNNs by connecting GNNs with the Weisfeiler-Lehman graph isomorphism test [Xu et al., 2019]. The Jumping Knowledge Networks (JK-Nets) addressed the over-smoothing problem by allowing flexible combinations of representations from different layers [Xu et al., 2018]. Further developments included works like DiffPool by Ying et al. [2018], which introduced hierarchical pooling operations for graph-level representations, and Graph U-Nets by Gao and Ji [2019], which proposed graph pooling and unpooling operations analogous to those in conventional neural networks. The Simple Graph Convolution (SGC) by Wu et al. [2019a] demonstrated that removing nonlinearities and collapsing weight matrices can simplify GCNs while maintaining competitive performance. Additionally, the APPNP model combined GNNs with personalized PageRank to enable more flexible propagation schemes [Klicpera et al., 2018]. The Position-aware Graph Neural Networks (P-GNNs) by [You et al., 2019] incorporated structural information to capture positional and structural roles of nodes. Graphormer used effective structural encoding methods to make graph structure available to the model and thus Transformers architecture can be utilized for graph representation learning [Ying et al., 2021].

The success of spatial GNNs in representation learning of graphs is attributed to their design, which aligns with the underlying assumption that neighboring nodes are likely to belong to the same class. However, when applied to heterophilic graphs, where connected nodes often belong to different classes, spatial GNNs struggle to maintain high classification accuracy [Lim et al., 2021, Zhu et al., 2020b]. The reason is that these models are primarily designed for homophilic graphs and fail to capture complex patterns on heterophilic graphs. To address these limitations, recent research on spatial GNNs for heterophilic graphs has gained significant attention. Existing approaches can be categorized into three main groups based on their design strategies: (1) leveraging multi-hop neighborhoods, allowing information aggregation beyond immediate neighbors; (2) discovering distant neighbors, enabling better structural awareness; and (3) adaptively aggregating neighbor information, ensuring more flexible and context-aware message passing. These advancements aim to enhance the performance of spatial GNNs in heterophilic graphs, bridging the gap between homophilic and heterophilic graph learning.

2.1.1 Higher Order Neighbors

k-hop neighbors are defined as $\mathcal{N}_k(v_i) = \{v_j | s(v_i, v_j) = k\}$ where $s(v_i, v_j)$ is the shortest path length between node v_i and v_j. As one-hop neighbors may not have similar features with ego node, thus GNN models only consider one hop neighbors perform bad on heterophilic graphs. Incorporating high order neighbor information allow the ego node to utilize latent useful information from long distance. In MixHop, each layer takes P-hops neighbors information into account and P different linear

transformation for different hops of neighbours. Then, all P-hops neighbor information are concatenated as the output of this layer [Abu-El-Haija et al., 2019]. H2GCN uses similar method to aggregate multi-hop information. It also proves that there is high probability that neighbors in the the second hop have the same label with the ego node [Zhu et al., 2020b]. EvenNet proves that ignoring odd hops neighbors will improve the robustness of the model and the learnt node representations have smaller variance than using all hops of neighbors [Lei et al., 2022]. UGCN points out that a graph can be divided into a complete homophilic graph, a complete heterophilic graph and a random graph. Thus, utilizing one hop, two hop and k-nearest hops of neighbors help achieve good performance on all kinds of graphs [Jin et al., 2021]. S2-GNN utilizes a cascade of filters with increasing Hadamard powers to generate a diverse set of functions and maintain the sparsity level in graph representations [Giraldo et al., 2024]. GAIPSRec uses a universal heterogeneous graph sampling framework for path sampling and takes into account all nodes on the aggregation path [Xiong et al., 2024].

2.1.2 Distant Neighbor Discovery

Different from above multi-hops GNNs which directly uses given topology, the definition of neighbors and the graph topology are changed in distant neighbor discovery methods. Neighbor set $\mathcal{N}_\tau(v_i) = \{v_j | s(v_i, v_j) \leq \tau\}$, where $s(v_i, v_j)$ is a metric to evaluate the similarity between node v_i and v_j, τ is a threshold.

Geom-GCN maps nodes of graph to a new latent continuous space and latent neighbors are defined according to distance in the latent space [Pei et al., 2020]. CPGNN defines compatible matrix between ego node and its neighbors. The learnable compatible matrix are incorporated in the aggregation process [Zhu et al., 2020a]. In GloGNN and GloGNN++, global homophilic neighbors are found through solving an optimization problem and a learnable coefficient matrix allow signed values in aggregation [Li et al., 2022]. Traditional GNNs tend to change the similarities between nodes during learning, SimP-GCN preserves node similarity while exploiting graph structure during updating. Self-supervised learning is used to capture the feature similarity and different relations between nodes [Jin et al., 2020]. LRGNN proposes a novel approach with the help of neural architecture search (NAS) to design data-specific inter-layer connections for stacking-based GNNs [Wei et al., 2023].

2.1.3 Adaptive Feature Aggregation

The feature aggregation process can be written as:

$$x_i^l = agg(\alpha_i^l x_i^{l-1}, \{\alpha_{ij}^l h_j^{l-1} | v_j \in \mathcal{N}^l(v_i)\}) \tag{2.4}$$

where x_i^l is the node representation of v_i in the l-th layer, α_i^l is the coefficient of the ego node v_i, α_{ij} is the edge weight of edge e_{ij} between the ego node v_i and its neighbor v_j, $\mathcal{N}^l(v_i)$ is the neighbor set considered in the l-th layer feature aggregation. The adaptive feature aggregation mechanism dynamically calculates the coefficient α_{ij}^l according to node representations h_i, h_j and the structural information.

GAT is a classic GNN model that adaptively assigns weight to an edge e_{ij} only according to the similarity of node representations x_i^l, x_j^l [Velickovic et al., 2018]. Compared with GCN, its variant GCNII has a learnable coefficient to control the ratio of information of ego node and information from its neighbors, which improves performance on heterophiliic graphs [Chen et al., 2020]. MAGNA extend the attention mechanism to multi-hop neighbors [Wang et al., 2020]. FAGCN uses a self-gating mechanism to adaptively integrate low frequency and high frequency signals in the process of message passing [Bo et al., 2021]. The coefficient α_{ij}^l is divided into two parts, $\alpha_{ij,L}^l$ and $\alpha_{ij,H}^l$, weighting the importance of low frequency and high frequency signals separately. ACM uses adaptive multiple channel mixing instead of uni-channel [Luan et al., 2022]. In other words, the coefficient α_{ij}^l is extended to a set $\alpha_{ij}^l = \{\alpha_{ij,1}^l, \alpha_{ij,2}^l, \ldots, \alpha_{ij,d}^l\}$ for d-dimensional node features. GBK-GNN incorporates two kernels to capture the homopholic information and the heterophilic information separately and a gate function is applied to choose which kernel to use for a pair of nodes [Du et al., 2021]. FSGNN decouples the feature aggregation and the layer wise representation learning, features are assigned weights summed to one during aggregation [Maurya et al., 2021]. A local mixing pattern is considered and the graph is transformed into a computation graph that node proximity and structures information are treated as two different types of edges. The aggregation process is conducted on the multi relational graph [Suresh et al., 2021]. A node-wise adaptive diffusion mechanism for information aggregation is developed in ApeGNN [Zhang et al., 2023b]. The AP-DGNN assigns unique aggregation combination weights to each node and category, culminating in a final model representation through a process of weighted aggregation [Chen et al., 2024].

2.2 Spectral Graph Neural Networks

Spectral GNNs have their roots in spectral graph theory. It provides a mathematical framework for analyzing graph structures in spectral domain. For a graph matrix, we can conduct eigendecomposition to get its eigenvalues and eigenvectors. These eigenvectors form a complete basis for signals on the graph. This connection with Fourier analysis allows us to interpret graph convolutions in the spectral domain. Eigenvalues represent frequencies and eigenvectors correspond to different spatial patterns on the graph.

An early work of spectral GNNs is [Bruna et al., 2013]. Graph convolutions are implemented as graph filters in the spectral domain, showing the potential of spectral methods in deep learning for non-Euclidean data. However, this work suffers from high computing cost due to the eigendecomposition of graph matrix. To address this, Defferrard et al. [2016] proposed ChebNet, which utilized Chebyshev polynomials to approximate graph convolution function. It achieved localized filtering while avoiding explicit eigendecomposition and improved the scalability. The success of ChebNet inspired various extensions and refinements, such as CayleyNet, JacobiConv and BernNet [He et al., 2021, Levie et al., 2019, Wang and Zhang, 2022] that different polynomial basis are used to enhance the stability of training and flexibility of design of graph convolution functions.

Spectral GNNs utilize the entire graph spectrum to capture strucuture information of graphs. This makes them suitable for heterophilic graphs, where nodes with different labels are more likely to be connected. Unlike spatial GNNs relying on neighborhood aggregation, spectral GNNs learn complex graph convolution functions, enabling them to enlarge or shrink frequency components of graph signals. GPRGNN leveraged a generalized PageRank framework to adaptively adjust different frequency components, making it robust to both homophilic and heterophilic structures [Chien et al., 2021]. ARMA utilized adaptive rational functions to approximate graph convolution functions to fit for graph topology [Bianchi et al., 2021].

Different eigenvalues correspond to different frequency components of graph signals, i.e., different connectivity patterns on graph. High frequency components capture osculation between connected nodes, representing multi-partite structure while low frequency components capture smoothness between connected nodes, representing cluster structure. Therefore, retaining high frequency components are important for heterophilic graphs while enhancing low frequency components are important for homophilic graphs. This principle is explored by [Bo et al., 2021] that adaptive frequency response mechanisms were introduced to dynamically adjust the emphasis on high and low frequency components. Similarly, BernNet proposed to use several band pass graph filters to extract useful frequency components from graph signal [He et al., 2021]. It allows to learn very complex graph convolution functions to adapt to different graph homophily levels.

These works demonstrate the superior of spectral GNNs in capturing complex graph structures from graphs. Spectral GNNs can adapt to both heterophily and homophily graphs while spatial GNNs heavily rely on the defined local connectivities.

2.2.1 Polynomial Spectral Graph Neural Networks

Spectral GNNs with polynomial graph filters are computationally efficient and localized in the vertex domain [Defferrard et al., 2016, Hammond et al., 2009]. Various

polynomial bases are utilized to approximate these graph filters. For instance, ChebNet and ChebNetII use Chebyshev polynomial bases [Defferrard et al., 2016, He et al., 2022a], while JacobiConv employs the Jacobi polynomial basis [Wang and Zhang, 2022]. These models leverage orthogonal polynomial bases, which provide efficient approximation, as the approximation error decreases exponentially with the order of the polynomial. Additionally, non-orthogonal polynomial spectral GNNs have gained attention. For example, BernNet [He et al., 2021] uses the Bernstein polynomial basis, ensuring all frequency responses of graph signals are non-negative. GPRGNN [Chien et al., 2021], on the other hand, employs a monomial polynomial basis, providing an intuitive explanation for node classification on graphs with diverse label distributions. GLN adopts a generalized Laguerre polynomial basis for function approximation [Li and Wang, 2024], while FavardGNN learns an optimal polynomial basis directly from the graph structure and signals [Guo and Wei, 2023]. UniFilter introduces a universal polynomial basis capable of handling graphs with varying degrees of heterophily [Huang et al., 2024].

For a graph with the adjacency matrix A, the degree matrix D, and the identity matrix I, we use $\hat{L} = I - D^{-1/2}AD^{-1/2}$, $\tilde{L} = -D^{-1/2}AD^{-1/2}$, $\tilde{A} = D^{-1/2}AD^{-1/2}$, and $\tilde{A}' = (D+I)^{-1/2}(A+I)(D+I)^{-1/2}$ to denote the normalized Laplacian matrix, the shifted normalized Laplacian matrix, the normalized adjacency, matrix and the normalized adjacency matrix with self-loops, respectively. In the following, we provide the detailed description and formulation of spectral GNNs with polynomial graph filters achieving the state-of-the-art performance in node classification task.

ChebNet [Defferrard et al., 2016]: This model uses the Chebyshev basis to approximate a spectral filter:

$$\hat{Y} = \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) f_W(X)$$

where X is the raw feature matrix, $\Theta = [\theta_0, \theta_1, \ldots, \theta_K]$ is the graph convolution parameter, W is the feature transformation parameter and $f_W(X)$ is usually a 2-layer MLP. $T_k(\tilde{L})$ is the k-th Chebyshev basis expanded on the shifted normalized graph Laplacian matrix $\tilde{L}$ and is recursively calculated:

$$\begin{aligned} T_0(\tilde{L}) &= I \\ T_1(\tilde{L}) &= \tilde{L} \\ T_k(\tilde{L}) &= 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L}) \end{aligned}$$

ChebNetII [He et al., 2022a]: The model is formulated as

$$\hat{Y} = \frac{2}{K+2} \sum_{k=0}^{K} \sum_{j=0}^{K} \theta_j T_k(x_j) T_k(\tilde{L}) f_W(X),$$

where X is the input feature matrix, W is the feature transformation parameter, $f_W(X)$ is usually a 2-layer MLP, $T_k(\cdot)$ is the k-th Chebyshev basis expanded on $\cdot$, $x_j = \cos\left((j+1/2)\,\pi/(K+1)\right)$ is the j-th Chebyshev node, which is the root of the Chebyshev polynomials of the first kind with degree $K+1$, and θ_j is a learnable parameter. Graph convolution parameter in ChebNet is reparameterized with Chebyshev nodes and learnable parameters θ_j.

JacobiConv [Wang and Zhang, 2022]: This model uses the Jacobi basis to approximate a filter as:

$$\hat{Y} = \sum_{k=0}^{K} \theta_k J_k^{a,b}(\tilde{A}) f_W(X),$$

where X is the input feature matrix, $\Theta = [\theta_0, \theta_1, \ldots, \theta_K]$ is the graph convolution parameter, W is the feature transformation parameter and $f_W(X)$ is usually a 2-layer MLP. $J_k^{a,b}(\tilde{A})$ is the Jacobi basis on normalized graph adjacency matrix $\tilde{A}$ and is recursively calculated as

$$\begin{aligned}
J_k^{a,b}(\tilde{A}) &= I \\
J_k^{a,b}(\tilde{A}) &= \frac{1-b}{2} I + \frac{a+b+2}{2} \tilde{A} \\
J_k^{a,b}(\tilde{A}) &= \gamma_k \tilde{A} P_{k-1}^{a,b}(\tilde{A}) + \gamma'_k P_{k-1}^{a,b}(\tilde{A}) + \gamma''_k P_{k-2}^{a,b}(\tilde{A})
\end{aligned}$$

where $\gamma_k = \frac{(2k+a+b)(2k+a+b-1)}{2k(k+a+b)}, \gamma'_k = \frac{(2k+a+b-1)(a^2-b^2)}{2k(k+a+b)(2k+a+b-2)}, \gamma''_k = \frac{(k+1-1)(k+b-1)(2k+a+b)}{k(k+a+b)(2k+a+b-2)}$. a and b are hyper-parameters. Usually, grid search is used to find the optimal a and b values.

APPNP [Klicpera et al., 2018]: it is a spectral filter based on personalized PageRank procedure and the model could be formulated and

$$\hat{Y} = \sum_{k=0}^{K} \alpha(1-\alpha)^k \tilde{A}'^k f_W(X) \tag{2.5}$$

where α is the teleport probability used in PageRank algorithm and f is a node feature transformation function parameterized by θ.

GPRGNN [Chien et al., 2021]: This model uses the monomial basis to approximate a

filter:

$$\hat{Y} = \sum_{k=0}^{K} \theta_k \tilde{A}'^k f_W(X)$$

where X is the input feature matrix, $\Theta = [\theta_0, \theta_1, \ldots, \theta_K]$ is the graph convolution parameter, W is the feature transformation parameter and $f_W(X)$ is usually a 2-layer MLP. $\tilde{A}'$ is the normalized adjacency matrix with self-loops.

BernNet [He et al., 2021]: This model uses the Bernstein basis for approximation:

$$\hat{Y} = \sum_{k=0}^{K} \theta_k \frac{1}{2^K} \binom{K}{k} (2I - \hat{L})^{K-k} \hat{L}^k f_W(X)$$

where X is the input feature matrix, $\Theta = [\theta_0, \theta_1, \ldots, \theta_K]$ is the graph convolution parameter, W is the feature transformation parameter and $f_W(X)$ is usually a 2-layer MLP. $\hat{L}$ is the normalized Laplacian matrix.

2.2.2 Rational Spectral Graph Neural Networks

Rational spectral GNNs, while offering greater flexibility, introduce increased complexity. For instance, CayleyNet applies parametric rational functions to isolate relevant frequency bands [Levie et al., 2019]. ARMA integrates moving average components to enhance noise robustness and capture long-term graph dynamics [Bianchi et al., 2021]. LanczosNet enables multi-scale analysis of graph signals, providing further insights into spectral behavior [Liao et al., 2019]. DFNet uses feedback-looped filters for better vertex localization and robust performance [Wijesinghe and Wang, 2019].

We summarize GNN models in Table. 2.1 according to whether they are spatial GNNs or spectral GNNs and whether they use multi-hop neighbors, discovery distant neighbors and adaptive aggregate features.

GNNs	Spatial	Spectral	Multi-hop	Distant Neighbors	Adaptive Aggregation
GCN [Kipf and Welling, 2017]	√	-	-	-	-
GCNII [Chen et al., 2020]	√	-	-	√	
GAT [Velickovic et al., 2018]	√	-	-	-	√
GIN [Xu et al., 2019]	√	-	-	-	√
H2GCN [Zhu et al., 2020b]	√	-	√	-	-
LINKX [Lim et al., 2021]	√	-	√	-	√
MixHop [Abu-El-Haija et al., 2019]	√	-	√	-	-
EvenNet [Lei et al., 2022]	√	-	√	-	-
S2-GNN [Giraldo et al., 2024]	√	-	√	-	-
GAIPSRec [Xiong et al., 2024]	√	-	√	-	√
UGCN [Jin et al., 2021]	√	-	√	√	√
Geom-GCN [Pei et al., 2020]	√	-	-	√	-
CPGNN [Zhu et al., 2020a]	√	-	-	√	-
GloNet [Li et al., 2022]	√	-	-	√	√
SimP-GCN [Jin et al., 2020]	√	-	-	√	√
LRGNN [Wei et al., 2023]	√	-	-	√	-
MAGNA [Wang et al., 2020]	√	-	√	-	√
FAGCN [Bo et al., 2021]	√	-	-	-	√
ACM [Luan et al., 2022]	√	-	-	-	√
GBK-GNN [Du et al., 2021]	√	-	-	-	√
FSGNN [Maurya et al., 2021]	√	-	-	-	√
WRGAT [Suresh et al., 2021]	√	-	-	-	√
ApeGNN [Zhang et al., 2023b]	√	-	-	-	√
AP-DGNN [Chen et al., 2024]	√	-	-	-	√
APPNP [Klicpera et al., 2018]	√	√	√	-	-
GPRGNN [Chien et al., 2021]	√	√	√	-	√
ChebNet [Defferrard et al., 2016]	-	√	√	-	√
ChebNetII [He et al., 2022b]	-	√	√	-	√
BernNet [He et al., 2021]	-	√	√	-	√
JacobiNet [Wang and Zhang, 2022]	-	√	√	-	√
ARMA [Bianchi et al., 2021]	-	√	√	-	√
CayleyNet [Levie et al., 2019]	-	√	√	-	√
DFNet [Wijesinghe and Wang, 2019]	-	√	√	-	√
GLN [Li and Wang, 2024]	-	√	√	-	√
FavardGNN [Guo and Wei, 2023]	-	√	√	-	√
UniFilter [Huang et al., 2024]	-	√	√	-	√

Table 2.1: Summary of GNNs.

Chapter 3

Related Works

We provides a extensive review of related works in areas of optimization, expressiveness and generalization of GNNs in this chapter. We first introduce advanced optimizers designed to improve the training efficiency and stability of neural networks. Further, we examine the node distinguishability of GNNs, discussing metrics to evaluate the expressiveness of GNNs. Finally, we introduce main methods for generalization analysis and existing works on generalization of GNNs.

3.1 Optimization

Stochastic gradient descent (SGD) is widely used to solve the non-covex optimization problem. However, it performs poorly when the underlining loss landscape is complex or the training data is sparse [Luo et al., 2019], i.e, SGD cannot deal with poorly conditioned optimization problem. The reason is that SGD uniformly scales all directions of the gradient of loss function. To adapt to complex landscape of non-convex optimization problem, advanced gradient-adaptive optimizers are proposed to adaptively scale directions of gradients according to parameter updating history in a non-uniform way, such as the Adam, AdaGrad and RMSProp [Duchi et al., 2011, Kingma and Ba, 2014, Tieleman and Hinton, 2012].

A widely adopted strategy for addressing poorly conditioned optimization problems is *preconditioning* [Saad, 2003]. Above gradient-adaptive optimizers can be viewed as preconditioning methods on gradient with a diagonal matrix as a preconditioner [Cohen et al., 2022, Das et al., 2024, Gupta et al., 2018]. We summarize popular gradient-adaptive optimizers with diagonal preconditioner used in optimization of neural networks.

3.1.1 Diagonal Preconditioners

Momentum plays an important role in the design of these gradient-adaptive optimizers. It accelerates convergence in landscapes with small or oscillating gradients. Accumu-

lated gradients increase updating velocity. It also help escape local minima by maintaining momentum when the gradient change the direction [Nesterov, 2014]. When the data has sparse features, the infrequent feature may contain more informative information than frequent features. Thus, AdaGrad increases the learning rate of infrequent features and decreases the learned rate of frequent features by dividing square root of accumulated square gradient. The problem is that the accumulated gradient always increase and the parameter updating will vanish over time and thus leads to slow convergence. It enables efficient convergence in sparse data settings [Duchi et al., 2011]. RMSprop solves the vanishing updating of AdaGrad by maintaining a moving average of square gradient. It performs well in non-stationary problems where the underlying data distribution changes over time and is less sensitive to the learning rate hyperparameter [Tieleman and Hinton, 2012]. Adam combines RMSprop and momentum method. It uses exponential moving averages of first and second moments of gradients to encourage updates in confident direction that sign seldom change in that direction and focus on the sparse feature that seldom appear in training data. It is robust to noisy data. However, it may converge to sub optimal solution due to the short term memory of gradients. The decay parameter makes the large gradient only affect few steps. If most gradients are small, then Adam will converge to sub optimal [Balles and Hennig, 2017, Kingma and Ba, 2014]. AdaDelta eliminates the need for manual specification of the learning rate hyperparameter [Zeiler, 2012].

SGD and all above gradient-adaptive optimizers are first-order optimizers and their update rule can be summarized in a general format:

$$\begin{aligned} g_t &= \nabla \mathcal{L}_{\mathcal{S}}(\tau^t) \\ m_t &= \phi_t(g_1, g_2, \ldots, g_t), v_t = \psi_t(g_1, g_2, \ldots, g_t) \\ \tau^{t+1} &= \tau^t - \eta_t \frac{m_t}{\sqrt{v_t}} \end{aligned} \tag{3.1}$$

We show ϕ_t, ψ_t of SGD and several popular adaptive optimizers in Table 3.1.

Optimizers	u_t	ϕ_t	ψ_t
SGD	-	g_t	I
SGDM [Nesterov, 2014]	-	$\sum_{i=1}^{t} \gamma^{t-i} g_i$	I
AdaGrad [Duchi et al., 2011]	-	g_t	$diag(\sum_{i=1}^{t} g_i^2)/t$
RMSProp [Tieleman and Hinton, 2012]	-	g_t	$(1-\beta_2)diag(\sum_{i=1}^{t} \beta_2^{t-i} g_i^2)$
Adam [Kingma and Ba, 2014]	-	$\frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i}{1-\beta_1^t}$	$\frac{(1-\beta_2)diag(\sum_{i=1}^{t} \beta_2^{t-i} g_i^2)}{1-\beta_2^t}$
NAdam [Dozat, 2016]	$\beta_1(1 - 0.5 * 0.96^{t+1/250})$	$\frac{u_{t+1}(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i}{(1-\prod_{i=1}^{t+1} u_i)} + \frac{(1-u_t)g_t}{(1-\prod_{i=1}^{t} u_i)}$	$\frac{(1-\beta_2)diag(\sum_{i=1}^{t} \beta_2^{t-i} g_i^2)}{1-\beta_2^t}$

Table 3.1: Update rules of popular optimizers. I denotes identity matrix, γ, β_1, β_2 denote different weight decay parameters.

3.1.2 Non-diagonal Preconditioners

The preconditioner can be more general. The classic Newton's method and quasi-Netwton methods use the Hessian matrix or its approximation as the preconditioner [Lewis and Overton, 2012]. Equilibration preconditioner is proposed in [Dauphin et al., 2015] that can escape saddle points and empirically has better converge speed than RMSProp. For preconditioning in neural networks, there are two challenges: (1) neural networks are usually over-parameterized, preconditioning calls for storing and manipulating very large matrix; (2) there are interference between parameters in neural networks. In order to solve above two challenges, the preconditioner is usually designed as a diagonal matrix plus a matrix can be decomposed into two low-rank matrices [Cutajar et al., 2016, Gao et al., 2023]. Scaled gradient descent is proposed in [Jia et al., 2024] to solve the large scale matrix factorization problem that they theoretically prove that gradient descent method will converge faster with preconditioning. Another way to reduce the size of preconditoner is replacing the full matrix with a left side matrix and a right hand matrix [Gupta et al., 2018, Yen et al., 2024]. Structure-aware preconditioning algorithm Shampoo is proposed in [Gupta et al., 2018]. The full-matrix preconditioner flatten the parameter space while Shampoo use a left and a right low dimensional matrix respectively, which significantly reduce the size of preconditioner. Similar to Shampoo, different block low rank preconditioners are used for different layers of a neural network and all preconditioners share a biased matrix in [Yen et al., 2024].

Although non-diagonal preconditioners can achieve better performance, they are designed for specific architectures of neural networks and have higher computing cost, diagonal preconditioners dominant the optimization area of neural networks as it is good at adapting to different neural networks and has low computing cost.

3.2 Expressiveness

3.2.1 Expressive Power of Spectral GNNs

The expressive power of GNNs in graph classification setting has been extensively studied through the Weisfeiler-Lehman (WL) test [Li and Leskovec, 2022, Zhang et al., 2023a], which tests comprise a series of algorithms designed to determine whether two graphs are isomorphic [Weisfeiler and Leman, 1968]. However, the expressive power of GNNs in node classification setting remains less understood. There is limited works examining the node distinguish ability of spectral GNNs specifically for node classification.

[Wang and Zhang, 2022] measures the expressive power of linear spectral GNNs using the uniform approximation theorem and points that when there is no repeat

eigenvalues and the node feature contains all frequency components, it can produce any one-dimensional prediction. However, the two conditions are usual not hold in real-world graphs as symmetric structure is very common and node features are usually sparse in real-world graphs. We show this in Fig. 5.2 and Fig. 5.3 that multiplicity of eigenvalues and missing component is very common in real-world datasets. [Lu et al., 2024] propose a method to increase the expressive power of spectral GNNs based on the analysis of [Wang and Zhang, 2022]. It increase the distinct eigenvalues to meet the first condition. However, this method does not guarantee that the spectral GNN with matrix after eigenvalue correction is permutation equavariance. Node embeddings will depend on the node ordering, which is unreasonable.

Enhancing the expressive power of GNNs has been shown to improve their performance on node classification tasks. In parallel, another line of research focuses on improving GNN performance through graph rewiring techniques, which modify the graph topology. While closely related, this represents an orthogonal direction of investigation.

3.2.2 Graph Rewiring

Early methods include DropEdge and EDGEWIRE, which randomly or uses degree-preserving strategy to remove edges to alleviate over-smoothing [Chan and Akoglu, 2016, Rong et al., 2020]. Curvature-based approaches [Topping et al., 2022] adjust connectivity using discrete Ricci curvature to combat over-squashing, while locality-aware strategies preserve sparsity and structural efficiency [Barbero et al., 2024]. More recent methods include DiffWire, a differentiable and parameter-free approach guided by the Lovász bound [Adrián Arnaiz-Rodrıguez et al., 2022]; FoSR, which improves spectral expansion [Karhadkar et al., 2023]; and GPER, which selects edges based on effective resistance to enhance information flow [Shen et al., 2024].

Approaches that enhance the expressive power of spectral GNNs typically modify the graph matrices used in graph convolution without altering the graph topology. In contrast, graph rewiring methods adjust the topology in the spatial domain as a preprocessing step to address structural issues that affect model performance.

3.3 Generalization

Generalization is another aspect that will affect the performance of spectral GNNs. The generalization error of a model refers to the difference between the model's performance on the training data and its performance on unseen test data and it is used to evaluate the generalization capacity of a model. Generalization error bound provides a theoretical guarantee on how well a neural network will perform on unseen data. Ana-

lyzing factors affecting generalization error bounds and minimizing the generalization error bounds provide guidelines for model design and help build robust models.

We summarize main methods for generalization analysis, works studying the generalization of GNNs in graph classification and node classification in this section.

3.3.1 Main Methods

Classic methods for generalization analysis include: the Vapnik-Chervonenkis dimension, the Rademacher complexity, the uniform stability and the probably approximately correct (PAC)-Bayesian bound [Bartlett et al., 2017, El-Yaniv and Pechyony, 2007, Tsuzuku et al., 2019].

VC Theory. Vapnik-Chervonenkis (VC) theory provides the earliest framework for generalization analysis [Vapnik, 1968]. It provides a theoretical guarantee of how well a learning algorithm can generalize from training data to unseen data based on the complexity of the hypothesis space. It measures the complexity of a hypothesis class by assessing how many points it can shatter. A binary function f of parameter θ shatters a set of points $\mathcal{S} = \{x_1, x_2, \ldots, x_m\}$ if for any labeling of points $(y_1, y_2, \ldots, y_m)$ in the set, there exist a parameter θ that the function f makes no error when classifying the set of points, i.e., $f_\theta(x_i) > 0$ if $y_i = 1$ and $f_\theta(x_i) < 0$ if $y_i = 0$. VC dimension is the cardinality $|\mathcal{S}|$ of point set $\mathcal{S}$ can shatter:

$$VC(f) = \max_{f \text{ shatters } \mathcal{S}} |\mathcal{S}|$$

Early works extend VC dimension analysis to neural networks by computing the VC dimension for neural networks. It was shown that the VC dimension grows linearly with the number of parameters [Bartlett and Maass, 2003], implying that deep networks require large datasets to generalize well. However, it is pointed out that modern neural networks can perfectly fit randomly labeled training data, despite having VC dimensions that would predict poor generalization [Zhang et al., 2017]. Although VC-theory remains a fundamental theoretical tool in statistical learning, it does not suit for generalization analysis of deep neural networks.

Rademacher Complexity. Rademacher complexity measures the capacity of a hypothesis class by evaluating its ability to fit random noise. It provides a data-dependent generalization bound. The relationship between Rademacher complexity and generalization error was first discussed in [Bartlett and Mendelson, 2002]. It provides an expectation-based complexity measure by evaluating how well a function class correlates with random Rademacher variables and demonstrate how these complexities provide tighter bounds than VC dimension. Let $\mathcal{S} = \{x_1, x_2, \ldots, x_m\}$ be a set of samples and a function class $\mathcal{F}$ over X^m, the empirical Rademacher complexity of $\mathcal{F}$ on

$\mathcal{S}$ is

$$\mathcal{R}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{m}\mathbb{E}_{\sigma}[\sup_{f\in\mathcal{F}}\sum_{i=1}^{m}\sigma_i f(x_i)]$$

where $\sigma_1, \sigma_2, \ldots, \sigma_m$ are independent variables from Rademacher distribution that $Pr(\sigma_i = 1) = Pr(\sigma_i = -1) = \frac{1}{2}$.

The extension of Rademacher complexity to transductive learning settings was a milestone in the field [El-Yaniv and Pechyony, 2007]. It adaptes the framework to settings where test features were available during training. This development was particularly important for semi-supervised learning applications, where unlabeled test data can contribute to the learning process. Let $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}, \mathcal{U} = \{x_{m+1}, x_{m+2}, \ldots, x_{m+u}\}$ be a set of samples of known labels and a set of samples of unknown labels separately. Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{m+u})^T$ be a vector of i.i.d. random variables that

$$\sigma_i = \begin{cases} 1 & \text{with probability } p; \\ -1 & \text{with probability } p; \\ 0 & \text{with probability } 1-2p. \end{cases}$$

The Transductive Rademacher complexity with parameter p of a function class $\mathcal{F}$ over $\mathcal{S}+\mathcal{U}$ is:

$$\mathcal{R}_{\mathcal{S}+\mathcal{U},p}(\mathcal{F}) = (\frac{1}{m}+\frac{1}{u})\mathbb{E}_{\sigma}[\sup_{f\in\mathcal{F}}\sum_{i=1}^{m+u}\sigma_i f(x_i)]$$

Another breakthrough came from [Tolstikhin et al., 2014], who developed new bounds based on local Rademacher complexities in the transductive setting. Their work showed how incorporating locality information could lead to tighter generalization bounds, especially when the hypothesis class varies across different regions.

A key limitation of Rademacher complexity is that it scales with the number of parameters, leading to vacuous bounds when neural networks are deep or highly overparameterized.

Uniform Stability. Uniform stability quantifies how sensitive a learning algorithm is to small perturbations in the training data. The relation between uniform stability and generalization error was first built in [Bousquet and Elisseeff, 2002]. For a training set $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, replacing the i-th sample in $\mathcal{S}$ by (x_i', y_i') to form a new training set $\mathcal{S}^i = \{(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_i', y_i'), \ldots, (x_m, y_m)\}$, for a hypothesis f and a loss function $\ell(f(x_i), y_i)$, the uniform stability of f is

$$US(f) = \sup_{\mathcal{S}} |\mathbb{E}_{(x,y)\in\mathcal{S}}[\ell(f(x), y)] - \mathbb{E}_{(x,y)\in\mathcal{S}^{\rangle}}[\ell(f(x), y)]|$$

The extension of uniform stability to transductive settings is in [El-Yaniv and Pechyony, 2006]. It demonstrated how stability concepts could be adapted to scenarios

where features of samples in testing dataset are available during training. It is important to help understand the dynamics of semi-supervised learning. For a training set $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, replacing the i-th sample in $\mathcal{S}$ by (x_j, y_j) to form a new training set
$\mathcal{S}^{ij} = \{(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_j, y_j), \ldots, (x_m, y_m)\}$. Then, the uniform stability of ℓ is

$$US_{\mathcal{S}+\mathcal{U}}(f) = \sup_{a \in [m+u]} |\ell_a(\tau^T) - \ell_a(\tau_{ij}^T)| \tag{3.2}$$

where τ^T, τ_{ij}^T represents the parameter in the T-th step when training on $\mathcal{S}$ and $\mathcal{S}^{ij}$ separately.

The stability analysis to modern neural networks was advanced by [Hardt et al., 2016], which demonstrated that SGD inherently provides stability. This work helped explain the good generalization of deep neural networks trained with SGD without regularization. The relationship between stability and adversarial robustness was explored by [Xiao et al., 2022, Xing et al., 2021], which demonstrated that several commonly used practices in adversarial training actually enhanced the stability of models.

PAC-Bayes Theory. PAC-Bayes theory combines PAC learning with Bayesian principles. The foundational work [McAllester, 1999] related the expected error of a randomized classifier to its empirical error and the KL-divergence between the posterior and prior distributions. The general format of McAllester's PAC-Bayes bound is as follows:

$$\mathbb{E}_Q[\mathcal{L}_{D_u}(h)] \leq \mathbb{E}_Q[\mathcal{L}_{S_m}(h)] + \sqrt{\frac{\mathrm{KL}(Q||P) + \log \frac{m}{\delta}}{2(m-1)}}$$

where P and Q are the prior and posterior distribution over the hypothesis space respectively. $\mathrm{KL}(Q||P)$ is the Kullback-Leibler (KL) divergence between the posterior and prior distributions. D_u is the test dataset and S_m is the training dataset. $\mathbb{E}_Q[\mathcal{L}_{D_u}(h)]$ is the expected loss and $\mathbb{E}_Q[\mathcal{L}_{S_m}(h)]$ is the empirical loss. m is the number of training samples. Above bound hold with probability $1 - \delta$ for any $\delta \in (0, 1)$.

This bound is further expanded in [McAllester, 2003] and sophisticated model selection techniques within the PAC-Bayesian framework are developed. It is extend to Gaussian process models in [Seeger, 2002], demonstrating that PAC-Bayes bounds could be applied to kernel methods. A crucial milestone that PAC-Bayes bound in the transductive setting is [Derbeko et al., 2004b]. It adapted PAC-Bayes bounds to scenarios where unlabeled test data is available during training. This work laid the ground for applying PAC-Bayes theory to semi-supervised learning and domain adaptation. However, it only considers the binary classification with 0/1 margin loss. Another important milestone work is [Catoni, 2007], which introduced new PAC-Bayes bounds that are tighter than McAllester's bounds. Instead of using a simple square root dependency on KL divergence, Catoni's bound reweights the loss function in an exponential form, leading to an adaptive trade-off between empirical loss and model complexity.

It has the format:

$$\mathbb{E}_Q[\mathcal{L}(h)] \leq \frac{1}{\lambda} \log \mathbb{E}_Q \left[e^{\lambda \mathcal{L}_{S_m}(h)} \right] + \frac{\mathrm{KL}(Q||P)}{\lambda m}$$

where $\lambda > 0$ is a free parameter that controls the trade-off between empirical loss and model complexity.

The key differences between Catoni's bound and McAllester's bound are: (1) Catoni's bound exponentially reweights the empirical loss, making the bound more sensitive to small variations in loss values and leads to tighter bounds when the empirical loss has a low variance. (2) Catoni's bound introduces λ to balance between generalization error and model complexity. With a proper λ, a tighter generalization error bound can be obtained. (3) Catoni's bound scales linearly with λ while McAllester's bound scales with a square root of KL-divergence. This can results a more controlled trade-off between empirical loss and complexity.

Although Catoni's bound is theoretically tighter, McAllester's bound remains dominant in PAC-Bayes generalization analysis of neural networks due to following reasons: (1) Simplicity and interpretability. McAllester's PAC-Bayes bound is simple and interpretable, making it easier to implement in practical deep learning scenarios. It provides a clear trade-off between empirical loss and KL-divergence of the posterior and prior distributions, which aligns well with optimization objectives. The general format of the bound is more straightforward to use for optimization. It can be easily used in variational inference and stochastic optimization methods. (2) Compatibility with PAC-Bayes optimization Framework. PAC-Bayes optimization framework requires minimizing a PAC-Bayes bound during training. McAllester's bound is convex in KL-divergence and easy to optimize. Catoni's PAC-Bayes bound introduces a more complex dependency on the loss function and thus does not have a closed-form optimization objective that fits well with gradient descent based training.

The optimization perspective of PAC-Bayes theory has gained considerable attention in recent years. Researchers have developed various approaches to minimize PAC-Bayesian bounds directly, leading to learning algorithms with theoretical guarantees. For instance, [Germain et al., 2016] showed how to optimize PAC-Bayesian bounds for linear classifiers. [Dziugaite and Roy, 2017b] Introduced a PAC-Bayes training algorithm for deep networks by optimizing a PAC-Bayes bound using stochastic gradient descent. [Parrado-Hernández et al., 2012] proposed data-dependent priors for PAC-Bayes learning, allowing for tighter, data-specific generalization bound.

Classic methods analyze the generalization capability from different angles. Basically, VC dimension on depends on hypothesis set. Rademacher complexity, uniform stability and uniform convergence are data dependent. When there are lots of parameters in neural network, the hypothesis space is large, Rademacher complexity and uniform convergence cannot provide non-vacuous bound. Uniform stability does not

Methods	Hypothesis	Data Distribution	Non-Vacuous	Reasons
VC	√	-	-	Large parameter space
Rademacher Complexity	√	√	-	Large parameter space
Uniform Stability	-	√	-	Depend on iteration number
PAC-Bayes	√	√	√	Careful selection of priors and posteriors

Table 3.2: Classic methods for generalization analysis.

depend on the hypothesis space, instead it depend on the stability of the model and iteration number. When we need lots of iterations to train a neural network to convergence, the uniform stability bound is large, also a vacuous bound. PAC-Bayes bound can provides non-vacuous bound, but it calls for carefully selected priors and posteriors of parameters, otherwise, it gives a vacuous bound. We summarize them in Table 3.2.

Classical generalization analysis methods have provided important insights into model complexity and generalization error bounds. However, these methods often fail to produce tight meaningful bounds for modern deep neural networks due to their large parameter number, iterative training processes, and data dependencies. To address these limitations, recent works have explored alternative perspectives on generalization analysis, such as information-theoretic approaches, data augmentation-based approaches, and intermediate representation based approaches. These methods provide data-dependent insights, account for the training dynamics and the structure of representations.

Information-Theoretic Methods. Information-theoretic methods analyze how much information a trained model retains about the training data. One common approach is to bound the generalization error using mutual information between data and model parameters. Mutual information-based bounds accounting for data-dependent model behavior is proposed in [Xu and Raginsky, 2017], which has tighter generalization guarantees than classical methods. When models are trained with iterative algorithms such as stochastic gradient descent, generalization bounds derived from classical methods such as uniform stability often become loose. The training dynamics are taken into account and conditional mutual information bounds are introduced in [Negrea et al., 2019, Pensia et al., 2018]. However, these methods face computational challenges, as mutual information estimation in deep networks is costly and approximation techniques are needed [Steinke and Zakynthinou, 2020]. These information-theoretic bounds have also been compared with PAC-Bayes, VC-dimension, and uniform stability-based bounds, highlighting their ability to capture training-dependent generalization more effectively [Grünwald et al., 2021, Hellström et al., 2023].

Data Augmentation Based Methods. Data augmentation is widely used to improve generalization in deep learning. Recent studies show that it can also serve as a tool for generalization analysis. The assumption behind this approach is that a model that generalizes well should perform consistently across different augmentations of the data. How the statistics of intermediate representations under augmentation can be used to

measure generalization capability is discussed in [Dao et al., 2018]. Several works have explored theoretical justifications for augmentation as a regularization mechanism. Chen et al. [2019] proposed that data augmentation can be viewed as a variance regularization term, reducing the complexity of hypothesis space. Another perspective is to analyze augmentation using group theory, where augmentation transforms data points within structured orbits to maintain distributional invariance [Chen et al., 2019]. These works suggest that augmentation-based generalization analysis provides a practical and empirical complement to classical generalization bounds.

Intermediate Representation Based Methods. Another emerging approach to generalization analysis focuses on the structure of learned representations in deep models. Instead of measuring model complexity through the number of parameters, these methods analyze how hidden representations evolve during training. Kornblith et al. [2019] introduced canonical correlation analysis (CCA) for comparing representations across different neural network layers, demonstrating that well-generalized models learn similar intermediate representations. A more refined approach involves margin-based generalization bounds, where the distribution of margins in learned representations correlate with the generalization capability. Jiang et al. [2018] proposed that the margin distribution in hidden representations can be used to design better loss functions for improved generalization. Recent works have also extended margin-based analysis to GNNs, deriving generalization bounds using optimal transport [Chuang et al., 2021]. Instead of focusing on hidden layer representations, Mouton et al. [2023] studied the relation between input data margins and generalization gap, emphasizing the importance of considering data manifolds when designing neural networks for better generalization. These representation-based approaches shift the focus from model complexity to distribution of representations, offering a more direct and empirical understanding of generalization in deep learning.

Briefly, classical methods (e.g., VC, PAC-Bayes, Uniform Stability) provide mathematical rigor but struggle with neural networks. Recent approaches including information-theoretic generalization bounds, augmentation-based generalization analysis, and intermediate representation analysis provide data-dependent insights and shifts the focus from model complexity to distribution of representations for generalization analysis. Mutual information-based methods offer tighter bounds, especially for iterative algorithms like stochastic gradient descent, but are computationally expensive. Data augmentation and representation analysis are more practical but lack strong theoretical guarantees. Optimal transport-based methods are computing costly and need strong assumption on transport cost function. When aiming for rigor theoretical guarantee, classic methods are still the main stream for generalization analysis.

In the following, we review studies on generalization of GNNs, categorizing them into graph classification and node classification settings, with a primary focus on the latter. A summary of existing methods to analyze generalization of GNNs are listed in

Table. 3.3.

3.3.2 Generalization in Graph Classification

Graph classification is typically framed as an inductive learning task, where models learn from a set of training graphs and generalize to unseen ones.

One prominent method for analyzing generalization is the Vapnik–Chervonenkis (VC) bound. Scarselli et al. [2018] analyze the VC dimension of GNNs on a restricted class of graphs, deriving upper bounds to assess model capacity and generalization ability. Morris et al. [2023] further connect the VC dimension of a GNN to the number of colors produced by the 1-WL algorithm, reflecting the number of graph structures the model can distinguish. Franks et al. [2024] employs classical margin theory to investigate the conditions under which an architecture's increased expressivity aligns with improved generalization performance.

Another approach is PAC-Bayes bounds, which offer probabilistic generalization guarantees. Liao et al. [2021] derive PAC-Bayes bounds for GNNs, linking generalization to maximum node degree and model depth. Ju et al. [2023] refine these bounds by incorporating graph diffusion properties, capturing how structural variations impact generalization. Behboodi et al. [2022] extend the PAC-Bayes framework to equivariant networks, demonstrating how group symmetry properties influence generalization.

Rademacher complexity provides an alternative perspective on GNN generalization by characterizing the expressiveness of learned functions. Garg et al. [2020] derive data-dependent generalization bounds by analyzing the complexity of a GNN's computational tree, focusing on binary classification. Maskey et al. [2022] further demonstrate that generalization bounds increase with model complexity but decrease with higher average node degrees, suggesting a trade-off between model expressivity and stability.

Recently, Li et al. [2025] introduced a k-variance margin-based bound, linking GNN generalization to graph structure variance. This approach provides insights into the trade-off between expressivity and generalization, offering a more fine-grained perspective on how structural variations affect learning performance.

3.3.3 Generalization in Node Classification

Generalization analysis for node classification is more complex than for graph classification due to the transductive nature of the problem [Tang and Liu, 2023b].

One common approach is Rademacher complexity, which has a strong theoretical foundation in transductive learning [El-Yaniv and Pechyony, 2007]. Oono and Suzuki [2020] derive a generalization error bound for multi-scale GNNs using trans-

Work	Task	GNN Type	Technique	Learning Setting
[Garg et al., 2020]	Graph classification	MPGNN	Rademacher complexity	Inductive
[Behboodi et al., 2022]	Graph classification	Equivalent GNN	PAC-Bayes	Inductive
[Liao et al., 2021]	Graph classification	GCN & MPGNN	PAC-Bayes	Inductive
[Maskey et al., 2022]	Graph classification	MPGNN	Uniform convergence	Inductive
[Ju et al., 2023]	Graph classification	MPGNN	PAC-Bayes	Inductive
[Morris et al., 2023]	Graph classification	MPGNN	VC dimension	Inductive
[D'Inverno et al., 2024]	Graph classification	MPGNN	VC dimension	Inductive
[Scarselli et al., 2018]	Node classification	GNN*	VC dimension	-
[Verma and Zhang, 2019]	Node regression	One-hidden-layer GNNs (1 filter)	Uniform stability	Inductive
[Zhang et al., 2020]	Node classification	One-hidden-layer GNNs (K filters)	Uniform convergence	Transductive
[Oono and Suzuki, 2020]	Node classification	Multi-scale GNNs	Rademacher complexity	Transductive
[Zhou and Wang, 2021]	Node classification	GCN	Uniform stability	Inductive
[Cong et al., 2021]	Node classification	GCN	Uniform stability	Transductive
[Esser et al., 2021]	Node classification	GCN	Rademacher complexity	Transductive
[Ma et al., 2021]	Node classification	MPGNN	PAC-Bayes	Noniid
[Tang and Liu, 2023b]	Node classification	MPGNN	Rademacher complexity	Transductive
[Sun et al., 2024]	Node classification	MPGNN	PAC-Bayes bound	Transductive

Table 3.3: Summary of existing works on generalization analysis of GNNs.

ductive Rademacher complexity analysis. They also establish a connection between the weak learning condition and Rademacher complexity to derive a test error bound. Esser et al. [2021] show that under certain distributional assumptions, transductive Rademacher complexity provides meaningful insights into generalization in stochastic block models. Tang and Liu [2023b] demonstrate that the transductive Rademacher complexity of a GNN is proportional to the infinity norm of its graph matrix, providing generalization bounds for several classic GNN architectures.

A recent work by Sun et al. [2024] analyzes GNN generalization in a semi-supervised transductive setting using PAC-Bayesian bounds, showing how L-hop interactions between training and test nodes affect generalization. Their findings suggest that greater interaction between training and test nodes improves generalization performance.

Another key approach is uniform stability. Verma and Zhang [2019] establish a connection between the generalization bound of single-layer GCNs and the largest absolute eigenvalue of the graph matrix in a semi-supervised setting. Later, Zhou and Wang [2021] extend this analysis to multi-layer GCNs. However, both studies focus on subgraphs under an inductive setting. In contrast, Cong et al. [2021] examine the transductive setting and show that increasing GNN depth enhances stability and reduces generalization error, making their work the most closely related to ours. Instead of using a constant value as the gradient norm bound in their analysis, we analyze key factors affect the gradient norm, such as the polynomial order of spectral GNNs and the graph homophily.

We summarizes our work and existing generalization bound methods for node classification in Table 3.4, highlighting key aspects: (1) Inductive/Transductive Settings. The VC bound by Scarselli et al. [2018] is data-independent and thus agnostic to inductive or transductive settings. While some studies [Verma and Zhang, 2019,

Method		Analysis		Key Factors in the Bound				
		Lipschitz	**Gradient**	**Graph Matrix**	**Train Size**	**Depth**	**Homophily**	**Poly Order**
VC bound	[Scarselli et al., 2018]	-	-	n	√	-	-	-
RC bound	[Oono and Suzuki, 2020]	√	-	$\|MX\|_F$	√	√	-	-
(*transductive*)	[Esser et al., 2021]	√	-	$\|M\|_\infty, \|MX\|_{2\to\infty}$	√	√	-	-
	[Tang and Liu, 2023b]	√	-	$\|M\|_\infty$	√	√	-	-
PAC-Bayes bound	[Sun et al., 2024]	-	-	$\|P_r\|$	√	√	-	-
US bound	[Verma and Zhang, 2019]	-	√	$\|M\|_2$	√	-	-	-
(*inductive*)	[Zhou and Wang, 2021]	-	√	$\|M\|_2, \|MX\|_2$	√	√	-	-
US bound	[Cong et al., 2021]	√	√	$d_{\max}$	√	√	-	-
(*transductive*)	**Our work**	√	√	M_{ij}	√	-	√	√

Table 3.4: Summary of generalization bound methods for node classification. Each method is categorized by whether it relies on Lipschitz or gradient-based analysis, the graph matrix norm constraints considered, and other key factors such as training set size, model depth, homophily, and polynomial order. A check mark (√) indicates that the factor is accounted for in the bound; a dash (-) indicates it is not. Here, VC refers to Vapnik-Chervonenkis complexity, RC stands for Rademacher complexity, and US denotes uniform stability. n is the number of nodes, $d_{\max}$ is the maximum node degree, and $\|\cdot\|_2$, $\|\cdot\|_F$, $\|\cdot\|_\infty$, $\|\cdot\|_{2\to\infty}$ denote the spectral, Frobenius, infinity, and maximum column ℓ_2-norms, respectively. P_r denotes the set of training-test node pairs with the shortest path of length r.

Zhou and Wang, 2021] derive bounds for GNNs in inductive settings, others [Cong et al., 2021, Esser et al., 2021, Oono and Suzuki, 2020, Tang and Liu, 2023b] and our work address the more complex transductive setting. (2) Analysis Frameworks. Rademacher complexity-based methods estimate a model's capacity to fit noise based on graph structure and node features but do not account for node labels or graph homophily [Esser et al., 2021, Oono and Suzuki, 2020, Tang and Liu, 2023b]. Uniform transductive stability allows us to analyze the relationship between generalization and graph homophily. Notably, while [Cong et al., 2021] also employs uniform transductive stability, their analysis is limited to the impact of GNN depth, without considering homophily or polynomial order. (3) Key Factors in Bounds. Training sample size is a critical factor in all the bounds. Model depth (number of GNN layers) is addressed in [Cong et al., 2021, Esser et al., 2021, Sun et al., 2024, Tang and Liu, 2023b, Zhou and Wang, 2021]. Our work examines spectral GNNs, where the architecture comprises only one layer of K-order polynomials, as well as the effects of graph homophily. Unlike prior studies that focus on various graph matrix norms, our analysis considers the expectation of individual graph matrix elements. We further analyze the impact of graph homophily and polynomial order on generalization.

Chapter 4

Asymmetric Learning for Optimization

4.1 Overview

To characterize the phenomenon of largest eigenvalue difference of diagonal block Hessian matrix, particularly the relationship between the parameters Θ and W in spectral GNNs and their impact on optimization, we introduce the concept of the *block condition number* of the Hessian matrix. This metric serves as an indicator for assessing the difficulty of an optimization problem. Specifically, utilizing the block condition number, we show that poorly conditioned spectral GNNs are characterized by a large block condition number of the Hessian matrix.

A widely adopted strategy for addressing poorly conditioned optimization problems is *preconditioning* [Saad, 2003]. Advanced adaptive learning rate optimizers tackle poorly conditioned problems by assigning different learning rates to different parameters, calculated based on a gradient history [Duchi et al., 2011, Kingma and Ba, 2014]. However, these optimizers overlook the inherent parameter magnitude differences exhibited by the parameters Θ and W during the optimization of spectral GNNs. To accelerate the learning process and potentially achieve superior solutions, we propose *asymmetric learning* for training spectral GNNs. This new method involves scaling the gradient of the empirical loss function asymmetrically during each iteration. The scaling factors within the asymmetric preconditioner are determined by the *gradient-parameter norm ratio*, which is closely associated with the largest eigenvalues of the block Hessian matrices $\mathbf{H}_{\Theta,\Theta}$ and $\mathbf{H}_{W,W}$. Notably, this asymmetric preconditioning of the gradient corresponds to an effective preconditioner for the Hessian matrix $\mathbf{H}$. Under reasonable assumptions, we demonstrate that asymmetric preconditioning can effectively reduce the block condition number of the Hessian matrix $\mathbf{H}$, thereby facilitating more efficient optimization of spectral GNNs.

We summarize contributions of this chapter as follows: (1) We reveal that the inherent difference between the graph convolution parameter Θ and the feature trans-

formation parameter W in spectral GNNs leads to a poorly conditioned optimization problem. (2) We introduce the concept of block condition number to quantify optimization difficulty. (3) We introduce an asymmetric learning approach to improve the training and performance of spectral GNNs. (4) Our extensive experiments on eighteen benchmark datasets demonstrate that asymmetric learning consistently enhances spectral GNN performance across various graphs.

4.2 Problem Analysis

Let $S = (X, \{y_i\}_{i=1}^{m})$ be the training set containing m labeled nodes randomly selected from all nodes of the graph. Similarly, we obtain the validation set $\mathcal{D}_{val} = (X, \{y_i\}_{i=m+1}^{m+q})$ containing q labeled nodes. A loss function $\ell(\hat{y}_i, y_i; \Theta, W)$ estimates the distance between the truth node label y_i and the prediction $\hat{y}_i$ from a spectral GNN parameterized by Θ, W in Eq. (1.1). The empirical loss is $\mathcal{L}_S(\Theta, W) = \frac{1}{m}\sum_{i=0}^{m} \ell(\hat{y}_i, y_i; \Theta, W)$, and the validation loss is $\mathcal{L}_{\mathcal{D}_{val}}(\Theta, W) = \frac{1}{q}\sum_{i=m+1}^{m+q} \ell(\hat{y}_i, y_i; \Theta, W)$. We use $||\cdot||_2$ to denote the ℓ_2 norm of $\cdot$, $vec(\cdot)$ to vectorize a matrix by concatenating its rows, and $d_\Theta = |\Theta|$ and , $d_W = |vec(W)|$ to denote the total parameter number of Θ and W, respectively.

Poorly Conditioned Spectral GNNs. We begin by presenting the classic definition of a poorly conditioned problem and then introduce a new concept called the *block condition number*. Using this new concept, we analyze the poorly conditioned nature of spectral GNNs.

Definition 1 (Poorly Conditioned Problem [Golub and Van Loan, 2013])**.** Let $\mathcal{L}_S(\Psi)$ be a convex objective function, parameterized by Ψ, and $\mathbf{H}$ be its associated Hessian matrix. If $\mathbf{H}$ is ill-conditioned, the optimization of $\mathcal{L}_S(\Psi)$ is poorly conditioned. The extent to which $\mathbf{H}$ is ill-conditioned is quantified by the *condition number* of $\mathbf{H}$, defined as

$$\kappa(\mathbf{H}) = \frac{\lambda_{max}(\mathbf{H})}{\lambda_{min}(\mathbf{H})}. \tag{4.1}$$

Here, $\kappa(\mathbf{H})$ is the ratio of the absolute values of the maximum eigenvalue $\lambda_{max}(\mathbf{H})$ to the minimum eigenvalue $\lambda_{min}(\mathbf{H})$.

Remark 1. Eigenvectors corresponding to the maximum eigenvalue $\lambda_{max}(\mathbf{H})$ and the minimum eigenvalue $\lambda_{min}(\mathbf{H})$ represent the most and least sensitive directions of the loss function $\mathcal{L}_S$, respectively. A significant gap between $\lambda_{max}(\mathbf{H})$ and $\lambda_{min}(\mathbf{H})$ induces numerical instability during optimization and slow down the convergence rate [Bonnans et al., 2006].

In spectral GNNs, the Hessian matrix $\mathbf{H}$ can be partitioned into two diagonal blocks $\mathbf{H}_{\Theta,\Theta}$ and $\mathbf{H}_{W,W}$, and two non-diagonal blocks $\mathbf{H}_{\Theta,W}$ and $\mathbf{H}_{W,\Theta}$. The ij-th element of

$\mathbf{H}_{\Theta,\Theta}$ is $\frac{\partial^2 \mathcal{L}_S(\Theta,W)}{\partial \Theta_i \partial \Theta_j}$, where $i, j \in [1, d_\Theta]$. The ij-th element of $\mathbf{H}_{W,W}$ is $\frac{\partial^2 \mathcal{L}_S(\Theta,W)}{\partial W_i \partial W_j}$, where $i, j \in [1, d_W]$. The ij-th element of $\mathbf{H}_{\Theta,W}$ (equivalently $\mathbf{H}^T_{W,\Theta}$) is $\frac{\partial^2 \mathcal{L}_S(\Theta,W)}{\partial \Theta_i \partial W_j}$, with $i \in [1, d_\Theta]$ and $j \in [1, d_W]$.

To understand how the parameters Θ and W influence the optimization process, we propose the notion of block condition number for spectral GNNs.

Definition 2 (Block Condition Number)**.** For a spectral GNN parameterized by Θ and W, the *block condition number* of its Hessian matrix $\mathbf{H}$ with respect to the empirical loss is defined as:

$$\kappa'(\mathbf{H}) = \frac{\max(\lambda_{\max}(\mathbf{H}_{\Theta,\Theta}), \lambda_{\max}(\mathbf{H}_{W,W}))}{\min(\lambda_{\max}(\mathbf{H}_{\Theta,\Theta}), \lambda_{\max}(\mathbf{H}_{W,W}))}. \tag{4.2}$$

Remark 2. The block condition number $\kappa'(\mathbf{H})$ serves as a metric to quantify the disparity between the maximum eigenvalues of the block Hessian matrices $\mathbf{H}_{\Theta,\Theta}$ and $\mathbf{H}_{W,W}$. A spectral GNN is considered *poorly conditioned* if $\kappa'(\mathbf{H})$ is large. According to the interlacing theorem [Horn and Johnson, 2012], when a symmetric matrix is divided into blocks according to parameters, the eigenvalues of the submatrices interlace with the eigenvalues of the full matrix. In the case where the Hessian matrix $\mathbf{H}$ is positive semi-definite, we have $\max(\lambda_{max}(\mathbf{H}_{\Theta,\Theta}), \lambda_{max}(\mathbf{H}_{W,W})) \leq \lambda_{max}(\mathbf{H})$ and $\lambda_{min}(\mathbf{H}) \leq \min(\lambda_{max}(\mathbf{H}_{\Theta,\Theta}), \lambda_{max}(\mathbf{H}_{W,W}))$. Therefore, $\kappa'(\mathbf{H}) \leq \kappa(\mathbf{H})$. A problem identified as poorly conditioned using κ' will also be poorly conditioned when assessed using κ. However, the converse does not hold true. Thus, the block condition number offers a more stringent criterion for determining whether a problem is poorly conditioned, as it directly models the impact of the differences between the parameters Θ and W on the optimization process.

In spectral GNNs, the block condition number for heterophilic graphs tends to be larger than that for homophilic graphs. When $\lambda_{max}(\mathbf{H}_{\Theta,\Theta}) > \lambda_{max}(\mathbf{H}_{W,W})$, the block condition number $\kappa'(\mathbf{H})$ is reduced to the largest eigenvalue ratio shown in Figure 1.2, i.e., $\kappa'(\mathbf{H}) = \alpha$ when applying ChbNet on those four datasets. Thus, the optimization on heterophilic graphs is generally more difficult than the optimization on homophilic graphs.

The relation between the optimal learning rate η^* and the Hessian matrix $\mathbf{H}$ is that $\eta^* = \frac{1}{\lambda_{max}(\mathbf{H})}$ [Granziol et al., 2020]. For spectral GNNs, we observe that $\lambda_{max}(\mathbf{H}_{\Theta,\Theta}) \leq \lambda_{max}(\mathbf{H})$ and $\lambda_{max}(\mathbf{H}_{W,W}) \leq \lambda_{max}(\mathbf{H})$. Consequently, the learning rates must satisfy $\eta \leq \eta_\Theta$ and $\eta \leq \eta_W$, where η, η_Θ, and η_W denote the learning rates for all parameters, Θ, and W, respectively. Given that $\lambda_{max}(\mathbf{H}_{\Theta,\Theta}) \neq \lambda_{max}(\mathbf{H}_{W,W})$ in most cases, employing different learning rates for Θ and W is a common practice in the training of spectral GNNs, as evidenced in recent studies [He et al., 2021, 2022a].

4.3 Asymmetric Learning

In this section, we first introduce the *gradient-parameter norm ratio* (GPNR), which is computed based on the magnitudes of gradients and parameter values. We then discuss the relationship between GPNR and the maximum eigenvalue of the Hessian matrix. After that, we propose *asymmetric preconditioning* on the gradients of spectral GNNs and examine its impact on the Hessian matrix. We theoretically demonstrate that our asymmetric preconditioning can mitigate the poorly conditioned optimization of spectral GNNs under mild assumptions.

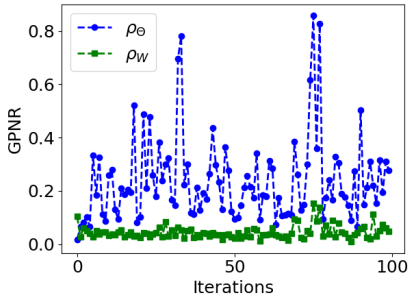


Figure 4.1: The Gradient-Parameter Norm Ratio (GPNR) of ρ_Θ is larger than ρ_W during iterations. It means the updating speed of Θ is faster than that of W.

4.3.1 Gradient-Parameter Norm Ratio

The update rule for gradient descent in spectral GNNs is given by

$$\begin{aligned} \Theta^t &= \Theta^{t-1} - \eta \nabla_\Theta \mathcal{L}_S(\Theta^t, W^t); \\ W^t &= W^{t-1} - \eta \nabla_W \mathcal{L}_S(\Theta^t, W^t), \end{aligned} \tag{4.3}$$

where η denotes the learning rate.

We introduce the notion of the gradient-parameter norm ratio to estimate the update speed of parameters.

Definition 3 (Gradient-Parameter Norm Ratio)**.** For a differentiable function $\mathcal{L}_S$ parameterized by Ψ, the gradient-parameter norm ratio (GPNR) of Ψ is defined as

$$\rho_\Psi = \frac{||\nabla \mathcal{L}_S(\Psi)||_2}{||\Psi||_2}, \tag{4.4}$$

where $\nabla \mathcal{L}_S(\Psi)$ is the gradient of $\mathcal{L}_S$ with respect to Ψ.

Remark 3. The GPNRs of parameters Θ^t and W^t are $\rho_\Theta^t = \frac{||\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t)||_2}{||\Theta||_2}$ and $\rho_W^t = \frac{||\nabla_W \mathcal{L}_S(\Theta^t, W^t)||_2}{||W||_2}$, respectively, at each iteration t. These ratios are effective metrics quantifying the update speeds of Θ and W. Generally, parameters should be updated at similar speeds [You et al., 2017], meaning that parameters with larger magnitudes undergo larger updates. However, our analysis of ρ_Θ^t and ρ_W^t, as shown in Figure 4.1, reveals that ρ_Θ^t is typically larger than ρ_W^t, indicating that Θ is updated more rapidly than W.

We further demonstrate the close relationship between the GPNR and the Hessian matrix.

Assumption 1 (Point Proximity)**.** Let Ψ be a point near a critical point Ψ^*. If $\|\Psi - \Psi^*\|_2$ is close to zero, we assume $\|\Psi - \Psi^*\|_2 \leq \|\Psi\|_2$.

Remark 4. In spectral GNNs, $\Psi = [\Theta; W]$. As the dimension of W is usually in thousands, there is a high probability that $\|\Psi\|_2$ is greater than zero. Thus, when $\|\Psi - \Psi^*\|_2$ is a very small number approximating zero, it is natural to assume that $\|\Psi - \Psi^*\|_2 \leq \|\Psi\|_2$.

Proposition 1 (GPNR and Maximum Eigenvalue of Hessian)**.** *For a neural network parameterized by Ψ and the Hessian matrix $\mathbf{H}_\Psi$ of empirical loss at point Ψ, when Ψ is near a critical point Ψ^* under Assumption 1, we have $\rho_\Psi \leq \lambda_{max}(\mathbf{H}_\Psi)$.*

Proof. Let Ψ be a parameter of the function $\mathcal{L}_S$ near a critical point Ψ^* such that $\epsilon = \Psi - \Psi^*$, $\|\epsilon\|_2 \to 0$ and $\nabla_\Psi \mathcal{L}_S(\Psi^*) = 0$. By the Taylor expansion, we have

$$\mathcal{L}_S(\Psi) = \mathcal{L}_S(\Psi^*) + \frac{1}{2}\epsilon^T \mathbf{H}_{\Psi^*}\epsilon + O(\|\epsilon\|_2^3), \tag{4.5}$$

where $\mathbf{H}_{\Psi^*}$ denotes the Hessian matrix at Ψ^* and $O(\|\epsilon\|_2^3)$ the remainder term captures the error introduced by truncating the Taylor series after the second-order term.

By differentiating both sides of Eq. (4.5) with respect to Ψ, we obtain

$$\nabla_\Psi \mathcal{L}_S(\Psi) = \mathbf{H}_{\Psi^*}\epsilon + O(3\|\epsilon\|_2\epsilon). \tag{4.6}$$

By differentiating again with respect to Ψ, we obtain

$$\mathbf{H}_\Psi = \mathbf{H}_{\Psi^*} + O\left(3 * \left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2} + \|\epsilon\|_2 I\right)\right). \tag{4.7}$$

By the definition of eigenvalue, for any vector Ψ, we have

$$\|\mathbf{H}_\Psi \epsilon\|_2 \leq \lambda_{max}(\mathbf{H}_\Psi)\|\epsilon\|_2. \tag{4.8}$$

This leads to the following:

$$
\begin{aligned}
\frac{\|\nabla_\Psi \mathcal{L}_S(\Psi)\|_2}{\|\Psi\|_2} &= \frac{\|\mathbf{H}_{\Psi^*}\epsilon + O(3\|\epsilon\|_2\epsilon)\|_2}{\|\Psi\|_2} \\
&= \frac{\| \left(\mathbf{H}_\Psi - O\left(3\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2} + \|\epsilon\|_2 I\right)\right)\right)\epsilon + O(3\|\epsilon\|_2\epsilon)\|_2}{\|\Psi\|_2} \\
&\leq \frac{\|\mathbf{H}_\Psi\epsilon\|_2}{\|\Psi\|_2} + \frac{\|O\left(3\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2} + \|\epsilon\|_2 I\right)\right)\epsilon\|_2}{\|\Psi\|_2} \\
&\quad + \frac{\|O(3\|\epsilon\|_2\epsilon)\|_2}{\|\Psi\|_2} \\
&\leq \frac{\lambda_{\max}(\mathbf{H}_\Psi)\|\epsilon\|_2}{\|\Psi\|_2} + \frac{O\left(\lambda_{\max}\left(3\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2} + \|\epsilon\|_2 I\right)\right)\|\epsilon\|_2\right)}{\|\Psi\|_2} \\
&\quad + \frac{O(3\|\epsilon\|_2^2)}{\|\Psi\|_2}, \quad \text{Eq. (4.8)} \\
&\leq \frac{\lambda_{\max}(\mathbf{H}_\Psi)\|\Psi\|_2}{\|\Psi\|_2} + \frac{O\left(\lambda_{\max}\left(3\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2} + \|\epsilon\|_2 I\right)\right)\|\epsilon\|_2\right)}{\|\Psi\|_2} \\
&\quad + \frac{O(3\|\epsilon\|_2^2)}{\|\Psi\|_2}, \quad \text{Assumption 1} \\
&= \lambda_{\max}(\mathbf{H}_\Psi) + \frac{O\left(\lambda_{\max}\left(3\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2} + \|\epsilon\|_2 I\right)\right)\|\epsilon\|_2\right)}{\|\Psi\|_2} + \frac{O(3\|\epsilon\|_2^2)}{\|\Psi\|_2}
\end{aligned}
\tag{4.9}
$$

The rank of matrix $\epsilon\epsilon^T$ is 1 and it has one nonzero eigenvalue $\|\epsilon\|_2^2$. All eigenvalues of the diagonal matrix $\|\epsilon\|_2 I$ are $\|\epsilon\|_2$, i.e.,

$$
\lambda_{max}(\epsilon\epsilon^T) = \|\epsilon\|_2^2; \quad \lambda_{max}(\|\epsilon\|_2 I) = \|\epsilon\|_2. \tag{4.10}
$$

The Weyl Inequality is shown as below.

Lemma 1 (Weyl Inequality [Knutson and Tao, 2001])**.** *Let A and B be two Hermitian matrices and $\lambda_{max}(\cdot)$ denote the maximum eigenvalue of a matrix. The following holds:*

$$
\lambda_{max}(A + B) \leq \lambda_{max}(A) + \lambda_{max}(B). \tag{4.11}
$$

This leads to the following:

$$
\begin{aligned}
&\lambda_{\max}\left(3\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2}+\|\epsilon\|_2 I\right)\right)\\
&\leq 3\left(\lambda_{\max}\left(\frac{\epsilon\epsilon^T}{\|\epsilon\|_2}\right)+\lambda_{\max}\left(\|\epsilon\|_2 I\right)\right), \quad \textit{Lemma 1}\\
&= 3\left(\frac{1}{\|\epsilon\|_2}\lambda_{\max}\left(\epsilon\epsilon^T\right)+\lambda_{\max}\left(\|\epsilon\|_2 I\right)\right)\\
&= 3\left(\frac{1}{\|\epsilon\|_2} * \|\epsilon\|_2^2+\|\epsilon\|_2\right), \quad \textit{Eq. (4.10)}\\
&= 6\|\epsilon\|_2
\end{aligned}
\tag{4.12}
$$

By substituting Eq. (4.12) into Eq. (4.9), we have

$$
\begin{aligned}
\frac{\|\nabla_\Psi \mathcal{L}_S(\Psi)\|_2}{\|\Psi\|_2} &\leq \lambda_{\max}(\mathbf{H}_\Psi)+\frac{O\left(6\|\epsilon\|_2^2\right)}{\|\Psi\|_2}+\frac{O\left(3\|\epsilon\|_2^2\right)}{\|\Psi\|_2}\\
&= \lambda_{\max}(\mathbf{H}_\Psi)+\frac{O\left(9\|\epsilon\|_2^2\right)}{\|\Psi\|_2}
\end{aligned}
$$

When $\|\epsilon\|_2 \to 0$, we have

$$
\rho_\Psi = \frac{\|\nabla_\Psi \mathcal{L}_S(\Psi)\|_2}{\|\Psi\|_2} \leq \lambda_{max}(\mathbf{H}_\Psi).
$$

□

Remark 5. The Hessian matrix $\mathbf{H}_\Psi$ provides insights into how gradients change in the vicinity of the point Ψ. The empirical loss $\mathcal{L}_S$ changes rapidly along the director of the eigenvector corresponding to $\lambda_{max}(\mathbf{H}_\Psi)$. Consequently, a small perturbation around Ψ can result in a significant increase in the gradient $\nabla_\Psi \mathcal{L}_S(\Psi)$, leading to a large $\rho_\Psi = \frac{||\nabla_\Psi \mathcal{L}_S(\Psi)||_2}{||\Psi||_2}$. When the parameter is near a critical point, $\lambda_{max}(\mathbf{H}_\Psi)$ is greater than or equal to the GPNR. Hence, if the GPNR is large, then $\lambda_{max}(\mathbf{H}_\Psi)$ is also large.

4.3.2 Asymmetric Preconditioner

We now define asymmetric preconditioning on the gradients of spectral GNNs. Let

$$
s_\Theta^t = \frac{||\Theta^t||_2}{||\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t)||_2}; \quad s_W^t = \frac{||W^t||_2}{||\nabla_W \mathcal{L}_S(\Theta^t, W^t)||_2}. \tag{4.13}
$$

Then, the *asymmetric preconditioner* for s_Θ^t and s_W^t is a diagonal preconditioner,

defined as

$$R^t = \begin{pmatrix} s_\Theta^t \mathbf{I}_{d_\Theta} & \mathbf{0} \\ \mathbf{0} & s_W^t \mathbf{I}_{d_W} \end{pmatrix} \tag{4.14}$$

where $\mathbf{I}_{d_\Theta}$ and $\mathbf{I}_{d_W}$ are the identity matrices of dimensions d_Θ and d_W, respectively. This yields new gradient updates

$$\begin{aligned} &[\overline{\nabla}_\Theta \mathcal{L}_S(\Theta^t, W^t); \overline{\nabla}_W \mathcal{L}_S(\Theta^t, W^t)] \\ &\quad = R^t [\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t); \nabla_W \mathcal{L}_S(\Theta^t, W^t)]. \end{aligned} \tag{4.15}$$

The GPNRs are also updated: $\rho_\Theta^t = \frac{||\overline{\nabla}_\Theta \mathcal{L}_S(\Theta^t, W^t)||_2}{||\Theta^t||_2} = 1$ and $\rho_W^t = \frac{||\overline{\nabla}_W \mathcal{L}_S(\Theta^t, W^t)||_2}{||W^t||_2} = 1$. Consequently, the parameters Θ and W are updated at the same rate, preventing overly aggressive or insufficient updates.

Asymmetric preconditioning affects the Hessian matrix, thereby altering the optimization landscape and the block condition number. The new Hessian matrix is given by

$$\mathbf{H}' = R^t \mathbf{H}. \tag{4.16}$$

We will prove later that the block condition number decreases after preconditioning, i.e., $\kappa'(\mathbf{H}') \leq \kappa'(\mathbf{H})$.

4.3.3 Moving Average of Parameter Norms

To ensure smooth changes in the parameter norms during iterations, especially when an optimizer updates parameters aggressively, we employ the exponential moving average technique [Shalev-Shwartz and Ben-David, 2014]. The parameter norms π_Θ^t and π_W^t at the t-th iteration are calculated using the exponential moving average as follows:

$$\begin{aligned} \pi_\Theta^t &= \beta_{\pi_\Theta} \pi_\Theta^{t-1} + (1 - \beta_{\pi_\Theta}) ||\Theta^t||_2; \\ \pi_W^t &= \beta_{\pi_W} \pi_W^{t-1} + (1 - \beta_{\pi_W}) ||W^t||_2, \end{aligned} \tag{4.17}$$

where β_{π_Θ} and β_{π_W} are the smoothing factors in $[0, 1]$ for the parameter norms π_Θ and π_W, respectively.

Asymmetric training can be integrated with any optimizer, and the optimization procedure is elaborated in Algorithm 1. Initially, the weights Θ^0 and W^0 are randomly initialized, and the best validation loss $\mathcal{L}_{\mathcal{D}_{val}}^b$ is set to infinity. During each iteration, the raw gradient $[\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t); \nabla_W \mathcal{L}_S(\Theta^t, W^t)]$ of the empirical loss is computed. Based on the specific update rules of the chosen optimizer OP, the update vectors δ_Θ^t and δ_W^t for parameters Θ and W are calculated. Subsequently, the moving average of parameter norms π_Θ^t, π_W^t and elements s_Θ^t, s_W^t in the asymmetric preconditioner R^t are

calculated as follows:

$$s_\Theta^t = \frac{\|\pi_\Theta^t\|_2}{\|\delta_\Theta^t\|_2}; \quad s_W^t = \frac{\|\pi_W^t\|_2}{\|\delta_W^t\|_2}. \tag{4.18}$$

The minimum validation loss $\mathcal{L}_{\mathcal{D}_{val}}^b$ is then updated, along with the corresponding best parameters Θ^b, W^b. The new gradient $[\overline{\nabla}_\Theta \mathcal{L}_S(\Theta^t, W^t); \overline{\nabla}_W \mathcal{L}_S(\Theta^t, W^t)]$ is computed by preconditioning the raw gradient $[\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t); \nabla_W \mathcal{L}_S(\Theta^t, W^t)]$ using the asymmetric preconditioner R^t. This new gradient serves as the input for any optimizer OP, resulting in new update vectors $\overline{\delta}_\Theta^t, \overline{\delta}_W^t$. Consequently, parameters Θ^{t+1} and W^{t+1} for the next iteration are updated accordingly.

Algorithm 1: Asymmetric Training

Input : Training set $S = (X, \{y_i\}_{i=1}^m)$, validation set $\mathcal{D}_{val} = (X, \{y_i\}_{i=m+1}^{m+q})$, loss function ℓ, learning rate η, maximum iteration number $t_{\max}$, and optimizer $OP(\cdot)$

Output: Trained model

Initialize weights Θ^0 and W^0, iteration number $t = 0$, best validation loss $\mathcal{L}_{\mathcal{D}_{val}}^b = +\infty$;
while $t \leq t_{\max}$ **do**
  Compute $[\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t); \nabla_W \mathcal{L}_S(\Theta^t, W^t)]$;
  $\delta_\Theta^t, \delta_W^t \leftarrow OP([\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t); \nabla_W \mathcal{L}_S(\Theta^t, W^t)])$;
  Set parameter norms π_Θ^t and π_W^t by Eq. (4.17);
  Compute s_Θ^t, s_W^t by Eq. (4.18);
  Compute asymmetric preconditioner R^t by Eq. (4.14);
  Compute validation loss $\mathcal{L}_{\mathcal{D}_{val}}(\Theta^t, W^t)$;
  if $\mathcal{L}_{\mathcal{D}_{val}}(\Theta^t, W^t) \leq \mathcal{L}_{\mathcal{D}_{val}}^b$ **then**
    $\Theta^b = \Theta^t$, $W^b = W^t$;
    $\mathcal{L}_{\mathcal{D}_{val}}^b = \mathcal{L}_{\mathcal{D}_{val}}(\Theta^t, W^t)$;
  end
  Get $[\overline{\nabla}_\Theta \mathcal{L}_S(\Theta^t, W^t); \overline{\nabla}_W \mathcal{L}_S(\Theta^t, W^t)]$ by Eq. (4.15);
  $\overline{\delta}_\Theta^t, \overline{\delta}_W^t \leftarrow OP([\overline{\nabla}_\Theta \mathcal{L}_S(\Theta^t, W^t); \overline{\nabla}_W \mathcal{L}_S(\Theta^t, W^t)])$;
  $\Theta^{t+1} = OP(\eta, \Theta^t, \overline{\delta}_\Theta^t)$;
  $W^{t+1} = OP(\eta, W^t, \overline{\delta}_W^t)$;
  $t = t + 1$;
end
return Θ^b, W^b

4.3.4 Smaller Block Condition Number.

Preconditioning the gradient $\nabla_\Psi \mathcal{L}_S(\Psi^t) = [\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t), \nabla_W \mathcal{L}_S(\Theta^t, W^t)]$ with asymmetric preconditioner R^t is equivalent to preconditioning the Hessian matrix $\mathbf{H}^t$ with

R^t in the t-th iteration:

$$R^t\mathbf{H}^t \Leftrightarrow R^t[\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t), \nabla_W \mathcal{L}_S(\Theta^t, W^t)]$$

where $\mathbf{H}^t_{ij} = \frac{\nabla_{\Psi_i}\mathcal{L}_S(\Psi^t)}{\nabla_{\Psi_j}\mathcal{L}_S(\Psi^t)}$, $(R^t\mathbf{H}^t)_{ij} = \frac{r^t_i \nabla_{\Psi_i}\mathcal{L}_S(\Psi^t)}{\nabla_{\Psi_j}\mathcal{L}_S(\Psi^t)}$, and r^t_i is the i-th diagonal element in R^t. To simplify the analysis, we set $\beta_{\pi_\Theta} = \beta_{\pi_W} = 0$ and consider the gradient descent optimizer [Shalev-Shwartz and Ben-David, 2014] for training. We show that preconditioning with the asymmetric preconditioner results in a smaller block condition number of the Hessian when the following assumptions are met.

Dataset	Texas	Wisconsin	Cornell	Actor	Chameleon
ρ_{Ψ_1}	0.011089	0.016569	0.020607	0.009565	0.010492
ρ_{Ψ_2}	0.011085	0.016552	0.020638	0.009590	0.010541
$\lambda_{max}(H_{\Psi_1})$	0.4357	0.4470	0.5855	0.2255	4.3220
$\lambda_{max}(H_{\Psi_2})$	0.4347	0.4460	0.5859	0.2281	4.3205
$\rho_{\Psi_1} \geq \rho_{\Psi_2}$	True	True	False	False	False
$\lambda_{max}(H_{\Psi_1}) \geq \lambda_{max}(H_{\Psi_2})$	True	True	False	False	True

Dataset	Squirrel	Citeseer	Pubmed	Cora
ρ_{Ψ_1}	0.013374	0.001995	0.005756	0.005261
ρ_{Ψ_2}	0.013391	0.001994	0.005721	0.005267
$\lambda_{max}(H_{\Psi_1})$	0.2683	0.1068	0.0933	0.1043
$\lambda_{max}(H_{\Psi_2})$	0.2687	0.1066	0.0931	0.1044
$\rho_{\Psi_1} \geq \rho_{\Psi_2}$	False	True	True	False
$\lambda_{max}(H_{\Psi_1}) \geq \lambda_{max}(H_{\Psi_2})$	False	True	True	False

Table 4.1: Empirical data supporting Assumption 2.

Assumption 2 (Proportional GPNR and Maximum Eigenvalue)**.** Let Ψ and Ψ' be two points near a critical point Ψ^* such that $\|\Psi - \Psi^*\|_2 = \epsilon_1$ and $\|\Psi' - \Psi^*\|_2 = \epsilon_2$, where $\epsilon_1 \to 0$ and $\epsilon_2 \to 0$. Denote $\mathbf{H}_\Psi$ and $\mathbf{H}_{\Psi'}$ as the Hessian matrices of the empirical loss at points Ψ and Ψ', respectively. If $\lambda_{\max}(\mathbf{H}_\Psi) \geq \lambda_{\max}(\mathbf{H}_{\Psi'})$, then $\rho_\Psi \geq \rho_{\Psi'}$.

Remark 6. The relationship between the GPNR and the largest eigenvalue of Hessian is that $\rho_\Psi \leq \lambda_{\max}(\mathbf{H}_\Psi)$ when Ψ is near a critical point Ψ^* as stated in Proposition 1. Consequently, it is reasonable to assume that if the largest eigenvalues of the Hessian matrices at points Ψ and Ψ' satisfy $\lambda_{\max}(\mathbf{H}_\Psi) \geq \lambda_{\max}(\mathbf{H}_{\Psi'})$, then the GPNRs at these points also satisfy $\rho_\Psi \geq \rho_{\Psi'}$. This assumption is supported by empirical data presented in Table 4.1. For two points Ψ_1 and Ψ_2 near a critical point, if $\lambda_{\max}(\Psi_1) > \lambda_{\max}(\Psi_2)$, then $\rho_{\Psi_1} \geq \rho_{\Psi_2}$. Here, $\lambda_{\max}(\Psi)$ denotes the largest eigenvalue of the Hessian at point Ψ, and ρ_Ψ represents the Gradient-Parameter Norm Ratio (GPNR). To test this, we select the point with the minimum empirical loss during training as the critical point. We then add random noise with a scale of 0.1 to generate two nearby points. For these

points, we compute their GPNR and the largest eigenvalue of the Hessian. On most datasets, this assumption is observed to hold.

Iteration t	100	150	200	250	300	350	400	450	500
$\frac{s_\Theta^t}{s_W^t}$	1.5015	2.2624	3.1904	2.8585	6.6780	4.1562	6.7149	11.5285	25.3993
$\frac{\lambda_{max}(\mathbf{H}_{W,W}^t)}{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t)}$	3.7220	0.8198	2.2311	2.1035	2.8941	2.6736	3.8299	11.0213	12.8264

Table 4.2: Empirical data supporting Assumption 3.

Assumption 3 (Mild Scaling)**.** For the asymmetric preconditioner R^t and the Hessian matrix $\mathbf{H}^t$, s_Θ^t and s_W^t in R^t satisfy the inequality $\frac{s_\Theta^t}{s_W^t} \geq \frac{\lambda_{max}(\mathbf{H}_{W,W}^t)}{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t)}$.

Remark 7. This assumption is also mild. The preconditioner is assumed to ensure that Proposition 2 holds. Specifically, it implies that the curvature of the loss function does not undergo significant changes after preconditioning. Empirical data supporting Assumption 3 is shown in Table 4.2. We train ChebyNet on Texas using gradient descent and show the two ratios every 50 iterations. We observe that there is a high probability that $\frac{s_\Theta^t}{s_W^t} \geq \frac{\lambda_{\max}(\mathbf{H}_{W,W}^t)}{\lambda_{\max}(\mathbf{H}_{\Theta,\Theta}^t)}$.

Proposition 2 (Spectral Equivalent Preconditioner.)**.** *Let R^t be the asymmetric preconditioner and $\mathbf{H}^t$ the Hessian matrix at the t-th iteration. If $\lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t) \geq \lambda_{max}(\mathbf{H}_{W,W}^t)$ and $\frac{s_\Theta^t}{s_W^t} \geq \frac{\lambda_{max}(\mathbf{H}_{W,W}^t)}{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t)}$, we have $\lambda_{max}(\mathbf{H}_{\Theta,\Theta}'^t) \geq \lambda_{max}(\mathbf{H}_{W,W}'^t)$ after preconditioning $\mathbf{H}'^t = R^t\mathbf{H}^t$.*

Proof. After preconditioning $\mathbf{H}^t$ with the preconditioner R^t, we have

$$\lambda_{max}(\mathbf{H}_{\Theta,\Theta}'^t) = s_\Theta \lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t);$$
$$\lambda_{max}(\mathbf{H}_{W,W}'^t) = s_W \lambda_{max}(\mathbf{H}_{W,W}^t).$$

This yields

$$\frac{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}'^t)}{\lambda_{max}(\mathbf{H}_{W,W}'^t)} = \frac{s_\Theta}{s_W}\frac{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t)}{\lambda_{max}(\mathbf{H}_{W,W}^t)}.$$

Then, when $\frac{s_\Theta}{s_W} \geq \frac{\lambda_{max}(\mathbf{H}_{W,W}^t)}{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}^t)}$, we obtain

$$\frac{\lambda_{max}(\mathbf{H}_{\Theta,\Theta}'^t)}{\lambda_{max}(\mathbf{H}_{W,W}'^t)} \geq 1.$$

Therefore, $\lambda_{max}(\mathbf{H}_{\Theta,\Theta}'^t) \geq \lambda_{max}(\mathbf{H}_{W,W}'^t)$ is derived. □

Remark 8. This proposition implies that the curvature of the loss function remains largely unchanged after preconditioning when the preconditioner R^t satisfies certain conditions. The sensitivity of the parameters to the loss function is maintained after preconditioning. If the loss function is more sensitive to Θ than W, then Θ remains more sensitive than W after preconditioning, and vice versa.

Dataset	Texas	Wisconsin	Cornell	Actor	Chameleon	Squirrel	Citeseer	Pubmed	Cora
$\lambda_{max}(H_{\Theta,\Theta})$	0.2681	1.0724	1.5734	0.1954	2.4245	0.0596	0.0042	0.1711	0.1891
$\lambda_{max}(H_{W,W})$	0.0219	0.0249	0.0138	0.0182	0.0093	0.0252	0.0065	0.0165	0.0299
$\lambda_{max}(H)$	0.4504	1.2951	1.6910	0.2134	2.4449	0.0890	0.0455	0.1723	0.2408

Table 4.3: Empirical data supporting underlying assumption that $\lambda_{max}(H) \geq \lambda_{max}(H_{\Theta,\Theta}) \geq \lambda_{max}(H_{W,W})$.

Theorem 1 (Smaller Block Condition Number.)**.** *Given an asymmetric preconditioner R^t, the Hessian matrix $\mathbf{H}^t$ at the t-th iteration, and the gradient descent as the optimizer, under Assumption 1, Assumption 2 and Assumption 3, we have $\kappa'(\mathbf{H}'^t) \leq \kappa'(\mathbf{H}^t)$ after preconditioning $\mathbf{H}'^t = R^t\mathbf{H}^t$.*

Proof. (1) We first prove that, under Assumption 1, the following holds for the GPNRs of Θ and W and the largest eigenvalues of block Hessian matrices $\mathbf{H}_{\Theta,\Theta}$ and $\mathbf{H}_{W,W}$: $\rho_\Theta \leq \lambda_{max}(\mathbf{H}_{\Theta,\Theta})$ and $\rho_{W,W} \leq \lambda_{max}(\mathbf{H}_{W,W})$.

For any parameter $\Psi = \{\Theta, W\}$ around a critical point $\Psi^* = \{\Theta^*, W^*\}$ such that $\epsilon_\Theta = \Theta - \Theta^*$, $\|\epsilon_\Theta\|_2 \to 0$ and $\epsilon_W = W - W^*$ and $\|\epsilon_W\|_2 \to 0$, the second-order Taylor expansion with the remainder term for Θ is

$$\begin{aligned}\mathcal{L}_S(\Theta, W) = \mathcal{L}_S(\Theta^*, W^*) + \nabla_\Theta \mathcal{L}_S(\Theta^*, W^*)^T \epsilon_\Theta \\ + \frac{1}{2}\epsilon_\Theta^T \mathbf{H}_{\Theta^*,\Theta^*}\epsilon_\Theta + O(\|\epsilon_\Theta\|_2^3).\end{aligned} \tag{4.19}$$

By differentiating both side of Eq. (4.19) with respect to Θ, we have

$$\nabla_\Theta \mathcal{L}_S(\Theta, W) = \mathbf{H}_{\Theta^*,\Theta^*}\epsilon_\Theta + O(3\|\epsilon_\Theta\|_2\epsilon_\Theta). \tag{4.20}$$

By differentiating both side of Eq. (4.19) twice with respect to Θ, we obtain

$$\mathbf{H}_{\Theta,\Theta} = \mathbf{H}_{\Theta^*,\Theta^*} + O\left(3 * \left(\frac{\epsilon_\Theta \epsilon_\Theta^T}{\|\epsilon_\Theta\|_2} + \|\epsilon_\Theta\|_2 * I\right)\right). \tag{4.21}$$

Repeat steps in Eq. (4.9), when $\|\epsilon_\Theta\|_2 \to 0$, we have

$$\rho_\Theta = \frac{\|\nabla_\Theta \mathcal{L}_S(\Theta, W)\|_2}{\|\Theta\|_2} \leq \lambda_{max}(\mathbf{H}_{\Theta,\Theta}). \tag{4.22}$$

By repeating Taylor expansion at $\Psi^* = \{\Theta^*, W^*\}$ for $\epsilon_W = W - W^*$ and above steps, when $\|\epsilon_W\|_2 \to 0$, we have

$$\rho_W = \frac{\|\nabla_W \mathcal{L}_S(\Theta, W)\|_2}{\|W\|_2} \leq \lambda_{max}(\mathbf{H}_{W,W}). \tag{4.23}$$

(2) Next, we show that under Assumption 2 and Assumption 3, the block condition number decreases after preconditioning.

For the Hessian matrix $\mathbf{H}^t$ and its preconditioned form $\mathbf{H}'^t = R^t\mathbf{H}^t$, where R^t is the asymmetric preconditioner with the diagonal elements s^t_Θ and s^t_W set according to Eq. (4.13). By Assumption 3, if $\lambda_{\max}(\mathbf{H}^t_{\Theta,\Theta}) \geq \lambda_{\max}(\mathbf{H}^t_{W,W})$, we have $\lambda_{\max}(\mathbf{H}'^t_{\Theta,\Theta}) \geq \lambda_{\max}(\mathbf{H}'^t_{W,W})$ after preconditioning. Thus, the block condition number of $\mathbf{H}'^t$ is obtained as

$$\begin{aligned}\kappa'(\mathbf{H}'^t) &= \frac{s^t_\Theta \lambda_{max}(\mathbf{H}^t_{\Theta,\Theta})}{s^t_W \lambda_{max}(\mathbf{H}^t_{W,W})} \\ &= \frac{s^t_\Theta}{s^t_W}\kappa'(\mathbf{H}^t).\end{aligned} \tag{4.24}$$

According to Assumption 2, when $\lambda_{\max}(\mathbf{H}^t_{\Theta,\Theta}) \geq \lambda_{\max}(\mathbf{H}^t_{W,W})$, we have $\rho^t_\Theta \geq \rho^t_W$, i.e., $\frac{\|\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t)\|_2}{\|\Theta^t\|_2} \geq \frac{\|\nabla_W \mathcal{L}_S(\Theta^t, W^t)\|_2}{\|W^t\|_2}$. When s^t_Θ and s_W are set according to Eq. (4.13), we have

$$\begin{aligned}\frac{s^t_\Theta}{s^t_W} &= \frac{\frac{\|\Theta^t\|_2}{\|\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t)\|_2}}{\frac{\|W^t\|_2}{\|\nabla_W \mathcal{L}_S(\Theta^t, W^t)\|_2}} \\ &= \frac{\frac{\|\nabla_W \mathcal{L}_S(\Theta^t, W^t)\|_2}{\|W^t\|_2}}{\frac{\|\nabla_\Theta \mathcal{L}_S(\Theta^t, W^t)\|_2}{\|\Theta^t\|_2}} \\ &\leq 1.\end{aligned} \tag{4.25}$$

By substituting Eq. (4.25) into Eq. (4.24), we obtain

$$\kappa'(\mathbf{H}'^t) \leq \kappa(\mathbf{H}^t).$$

□

Remark 9. This theorem indicates that the block condition number decreases in each iteration during training when applying the asymmetric preconditioner. When the block condition number of the Hessian matrix approaches one, changes in Θ and W lead to approximately the same changes in the empirical loss. As the block condition number evaluates how poorly conditioned an optimization problem is, a small block

Statistics	Texas	Wisconsin	Cornell	Actor	Chameleon	Squirrel
# Nodes	183	251	183	7,600	890	2,223
# Edges	295	466	280	26,752	27,168	131,436
# Features	1,703	1,703	1,703	932	2,325	2,089
# Classes	5	5	5	5	5	5
# Edge Homophily	0.11	0.21	0.3	0.22	0.24	0.22

Table 4.4: Statistics of six small heterophilic datasets [Pei et al., 2020, Platonov et al., 2023a, Rozemberczki et al., 2021].

Statistics	Roman_Empire	Amazon_Ratings	Tolokers	Minesweeper	Questions
# Nodes	22,662	24,492	11,758	10,000	48,921
# Edges	32,927	93,050	519,000	39,402	153,540
# Features	300	300	10	7	301
# Classes	18	5	2	2	2
# Edge Homophily	0.05	0.38	0.59	0.68	0.84

Table 4.5: Statistics of five large heterophilic datasets Platonov et al. [2023a].

Statistics	Citeseer	Pubmed	Cora	Computers	Photo	Coauthor-CS	Coauthor-Physics
# Nodes	3,327	19,717	2,708	13,752	7,650	18,333	134,493
# Edges	4,676	44,327	5,278	491,722	238,162	163,788	495,924
# Features	3,703	500	1,433	767	745	6,805	8,415
# Classes	6	5	7	10	8	15	5
# Edge Homophily	0.74	0.8	0.81	0.78	0.83	0.81	0.93

Table 4.6: Statistics of homophilic datasets, including three small datasets (Citeseer, Pubmed, Cora) and four large datasets (Computers, Photo, Coauthor-CS, Coauthor-Physics) [Kipf and Welling, 2017, Shchur et al., 2018, Zeng et al., 2020].

condition number indicts an easy optimization problem. The underlying assumption $\lambda_{\max}(\mathbf{H}^t) \geq \lambda_{\max}(\mathbf{H}^t_{\Theta,\Theta}) \geq \lambda_{\max}(\mathbf{H}^t_{W,W})$ in Theorem 1 holds on real-world graph datasets. Specifically, we present the values of $\lambda_{\max}(\mathbf{H})$, $\lambda_{\max}(\mathbf{H}_{\Theta,\Theta})$ and $\lambda_{\max}(\mathbf{H}_{W,W})$ at the 50-th iteration when ChebNet is trained on different datasets in Table 4.3. We observe that the underlying assumption holds.

4.4 Experiments

4.4.1 Experimental Setup

We present the datasets utilized in our experiments to evaluate the performance of spectral GNNs for node classification task. These datasets include graphs of diverse structures, node features, and class distributions, providing a comprehensive benchmark for assessing the effectiveness of our proposed solutions.

We categorize these datasets into four groups based on their characteristics: small

heterophiic graphs, large heterophilic graph, homophilic graphs. Detailed descriptions of these datasets are as follows.

- Texas, Wisconsin, Cornell [Pei et al., 2020] are created from webpages of universities. Each node is a webpage and if there is a hyperlink from one webpage to another, there is a directed edge connecting two nodes. We use the processed version in [Pei et al., 2020]. Five class labels for these nodes are student, staff, faculty, course and project. Node features are bag of words representing the page.

- Actor [Tang et al., 2009] is a dataset created from the film-director-actor-writer network. Only the actor sub-network is included. Each node is an actor and there is an edge between two nodes if they occour in the same Wikipedia page. Node features are key words of Wikipedia pages. All nodes are classified into five types according to key words of an actor's Wikipedia.

- Chameleon, Squirrel [Platonov et al., 2023b] are two datasets on specific topics of Wikipedia pages Rozemberczki et al. [2021]. Each node is a webpage and edges are mutual links between two webpages. Node features are representive nouns of the page and node label is assigned according to the monthly traffic of the web page. The original version of two datasets contains redundant nodes and edges, we use the directed clean version of Chameleon and Squirrel provided by [Platonov et al., 2023a] which removes repeated nodes in graphs.

- Roman_Empire [Platonov et al., 2023b] are from [Platonov et al., 2023a]. Roman_Empire is created from the Roman Empire article from Wikipedia. Nodes in the graph correspond to words in text and number of nodes equals the length of article. Edges are created when two words meet the condition that two words follow each other in text or two words are connected in the dependency tree of a sentence.

- Amazon_Ratings [Platonov et al., 2023b] is based on the Amazon product co-purchasing network from SNAP datasets [Leskovec and Krevl, 2014]. Nodes are products and edges represent that two products are frequently bought together.

- Tolokers [Platonov et al., 2023b] is from the Toloka crowdsourcing platform. Nodes represent tolokers participating in at least 13 seleted projects. If there is an edge between two nodes, it means that two tolokers have worked on same task.

- Minesweeper [Platonov et al., 2023b] is from the Minesweeper game. A graph is a regular 100×100 grid. Node is the cell and it is connected with its eight neighboring cells.

- Questions [Platonov et al., 2023b] is created from data from the question-answering website Yandex Q. Nodes represent users. There is an edge between two nodes if one user has answered questions of the other user in a one-year interval.

- Citeseer, Pubmed, Cora [Yang et al., 2016] datasets are created from academic citation graphs. Each node is a research work and uses a bag of words as feature vector for representation. The node label is assigned according to the research field of the work.

- Computers, Photo [Shchur et al., 2018] are also from the Amazon co-purchase data where nodes represent products and edges represent two products bought at the same time.

- Coauthor-CS, Coauthor-Physics [Shchur et al., 2018] are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. Nodes are authors and edges exist when two authors co-author a paper.

The statistical information of the datasets, including node numbers, edge number, feature dimensions, node class numbers, edge homophilic ratios are summarized in in Table 4.6.

To empirically validate the effectiveness of our approach, we perform experiments on above eighteen benchmark datasets for node classification tasks. For each dataset, a sparse splitting strategy is employed where nodes are randomly partitioned into train/-validation/test sets with proportions of 2.5%/2.5%/95%, respectively. Notably, for Citeseer, Pubmed, and Cora datasets, the setting from Chien et al. [2021], He et al. [2022a] is adopted: 20 nodes per class are allocated for training, 500 nodes for validation, and 1,000 nodes for testing.

We evaluate five prominent spectral GNN models as our baselines: ChebNet [Defferrard et al., 2016], GPRGNN [Chien et al., 2021], BernNet [He et al., 2021], JacobiConv [Wang and Zhang, 2022], and ChebNetII [He et al., 2022a], and compare their performances in the setups with and without asymmetric learning across all datasets. For each GNN model, we use GNN (S) to denote the model trained using Adam optimizer [Kingma and Ba, 2014] and GNN (AS) to denote the model trained using Adam optimizer with asymmetric learning, with $\Delta \uparrow$ indicating the performance improvement.

We conduct a grid search for hyper-parameters used during the training of spectral GNNs, including dropout rate, learning rate, exponential decay parameter, early stopping, and maximum training epochs. The exact search ranges for different hyperparameters are detailed in Table 4.7.

Hyper Parameters	GNNs	Range
dropout in MLP	All	0.5, 0.7, 0.8, 0.9
dropout after MLP	All	0.5, 0.7, 0.8, 0.9
learning rate of Θ	All	0.001, 0.01, 0.05
weight decay of Θ	All	0.0, 0.0001, 0.0005
weight decay of W	All	0.0, 0.0001, 0.0005
learning rate of W	All	0.005, 0.01, 0.05
β_{π_Θ}	All	0.9, 0.99
β_{π_W}	All	0.9, 0.99
a	JacobiConv	$-2.0, -1.0, -0.5, 0.0, 0.5, 1.0, 2.0$
b	JacobiConv	$-2.0, -1.0, -0.5, 0.0, 0.5, 1.0, 2.0$
initialization parameter α	JacobiConv	0.1, 0.5, 0.9, 1.0, 2.0
initialization parameter α	GPRGNN	0.1, 0.5, 0.9, 1.0, 2.0

Table 4.7: Grid search ranges of hyper parameters in asymmetric learning.

Model	Texas	Wisconsin	Actor	Chameleon	Squirrel	Cornell
ChebNet (S)	36.76±6.76	33.17±7.83	24.94±1.33	25.65±3.56	22.58±2.56	40.52±8.15
ChebNet (AS)	50.81±8.03	33.21±5.33	25.13±0.83	38.19±1.26	27.24±1.95	40.92±7.92
$\Delta\uparrow$	+14.05	+0.04	+0.19	+12.54	+4.66	+0.40
ChebNetII (S)	48.44±10.87	41.33±9.25	29.29±0.8	33.48±4.59	30.8±2.65	37.69±11.1
ChebNetII (AS)	50.75±10.0	47.33±10.17	29.49±0.9	35.58±3.7	35.94±2.66	51.85±6.01
$\Delta\uparrow$	+2.31	+6.00	+0.20	+2.10	+5.14	+14.16
JacobiConv (S)	50.17±7.87	46.08±9.08	32.66±0.71	26.57±3.6	23.15±4.47	49.71±7.75
JacobiConv (AS)	52.66±8.9	49.5±9.75	32.78±0.71	29.95±3.37	24.71±3.83	52.83±8.73
$\Delta\uparrow$	+2.49	+3.42	+0.12	+3.38	+1.56	+3.12
GPRGNN (S)	48.55±7.0	40.79±3.62	30.2±0.95	30.44±4.29	24.33±2.68	46.53±7.23
GPRGNN (AS)	49.54±5.5	43.46±6.29	30.75±0.76	31.63±4.41	24.82±4.63	47.63±6.71
$\Delta\uparrow$	+0.99	+2.67	+0.55	+1.19	+0.49	+1.10
BernNet (S)	52.14±8.09	49.33±8.21	30.57±0.78	29.45±3.22	25.94±3.35	48.15±6.59
BernNet (AS)	53.87±9.66	52.46±6.29	30.8±0.69	30.46±5.31	27.5±3.62	49.88±8.73
$\Delta\uparrow$	+1.73	+3.13	+0.23	+1.01	+1.56	+1.73

Table 4.8: Test accuracy with/without asymmetric learning on small heterophilic datasets (Texas, Wisconsin, Actor, Chameleon, Squirrel, Cornell). High accuracy indicates good performance.

4.4.2 Results and Discussion

Tables 4.8 to 4.10 present the results for heterophilic graphs and homophilic graphs, respectively.

Q1: How does asymmetric learning perform with graphs of varying edge homophily ratios? We notice that asymmetric learning provides more performance gains on heterophilic graphs compared to homophilic ones. As shown in Table 4.8, on six small heterophilic datasets, there is an average accuracy improvement of 3.08%, while an accuracy increase of approximately 1.36% is achieved on Romain_Empire and Amazon_Ratings in Table 4.9. In Table 4.10, the average accuracy improvement is

GNNs	Roman_Empire	Amazon_Ratings	Minesweeper	Tolokers	Questions
ChebNet (S)	45.32±0.65	38.87±0.26	86.34±0.19	69.88±0.26	48.55±1.04
ChebNet (AS)	48.11±0.41	39.38±0.19	86.71±0.15	70.05±0.36	51.84±0.31
Δ ↑	+2.79	+0.51	+0.37	+0.17	+3.29
ChebNetII (S)	55.06±0.32	38.33±0.42	74.49±3.76	69.37±0.6	63.99±0.32
ChebNetII (AS)	55.2±0.3	39.07±0.28	78.16±0.1	69.65±0.81	63.68±0.81
Δ ↑	+0.14	+0.74	+3.67	+0.28	-0.31
JacobiConv (S)	52.92±0.7	40.18±0.53	87.4±0.13	70.24±0.18	55.68±0.54
JacobiConv (AS)	54.16±0.38	47.17±1.37	87.66±0.11	70.08±0.28	56.25±0.65
Δ ↑	+1.24	+6.99	+0.26	-0.16	+0.57
GPRGNN (S)	55.48±1.3	39.86±0.3	86.68±0.32	67.05±0.97	53.76±0.41
GPRGNN (AS)	55.99±1.14	39.89±0.16	87.11±0.39	68.22±1.1	56.18±0.3
Δ ↑	+0.51	+0.03	+0.43	+1.17	+2.42
BernNet (S)	55.3±0.41	39.33±0.25	76.64±0.29	69.31±0.69	65.41±0.33
BernNet (AS)	55.5±0.2	39.78±0.3	77.02±0.17	70.39±0.62	64.88±0.31
Δ ↑	+0.20	+0.45	+0.38	+1.08	-0.53

Table 4.9: Performance of GNNs with/without asymmetric learning on large heterophilic datasets (Roman-Empire, Amazon-Ratings, Minesweeper, Tolokers, Questions) when data splitting is 2.5%/2.5%/95%. Test accuracy is used as the metric for Roman-Empire and Amazon-Ratings datasets and ROC AUC is reported on Minesweeper, Tolokers, Questions. High accuracy and ROC AUC indict good performance.

0.46% on homophilic graphs. These results align with our theoretical analysis, showing that asymmetric learning facilitates the optimization of poorly conditioned problems by reducing the block condition number of the Hessian matrix. Empirically, we observe that the block condition numbers of Hessian matrices are larger on heterophilic graphs compared to homophilic graphs. Consequently, asymmetric learning yields a more significant improvement in classification accuracy on heterophilic graphs than on homophilic graphs.

Q2: How does asymmetric learning perform with different spectral GNNs? Spectral GNNs can be categorized based on their polynomial basis into orthogonal spectral GNNs (ChebNet, ChebNetII, and JacobiConv) and non-orthogonal spectral GNNs (GPRGNN and BernNet). We observe that asymmetric learning generally exhibits superior performance with orthogonal spectral GNNs compared to non-orthogonal spectral GNNs. For example, there is a 5.15% accuracy improvement with orthogonal spectral GNNs using asymmetric learning on the six small heterophilic datasets (Table 4.8), whereas non-orthogonal spectral GNNs show a 1.69% accuracy improvement. A plausible explanation is that the interference between different orders of orthogonal polynomial bases is significantly lower than that between non-orthogonal polynomial bases. Suppose that $\Theta = \{\theta_k\}_{k=0}^{K}$, where each θ_k is associated with a polynomial basis T_k of order k. Due to the interference in non-orthogonal polynomial bases, a sophisticated preconditioner is required, rather than simply scaling all θ_k uniformly by s_Θ, to achieve better performance.

Q3: How does asymmetric learning perform with graphs of different sizes? Asym-

Model	Citeseer	Pubmed	Cora	Computers	Photo	Coauthor-CS	Coauthor-Physics
ChebNet (S)	65.49±1.39	74.88±1.47	80.26±0.86	78.76±1.98	90.26±2.19	90.74±0.6	94.69±0.31
ChebNet (AS)	66.63±1.1	76.53±1.52	81.02±0.92	79.08±2.6	92.38±0.41	90.76±0.73	94.92±0.1
Δ ↑	+1.14	+1.65	+0.76	+0.32	+2.12	+0.02	+0.23
ChebNetII (S)	70.07±1.13	78.87±1.28	82.55±0.78	83.49±0.9	92.51±0.43	92.18±0.23	94.96±0.13
ChebNetII (AS)	70.75±0.78	79.5±0.94	82.52±0.95	84.38±1.3	92.69±0.43	93.41±0.23	95.13±0.09
Δ ↑	+0.68	+0.63	-0.03	+0.89	+0.18	+1.23	+0.17
JacobiConv (S)	67.86±1.08	76.89±1.67	77.44±1.23	84.77±0.7	92.18±0.42	93.06±0.3	95.07±0.16
JacobiConv (AS)	68.17±0.9	77.77±1.37	77.68±1.24	84.91±0.46	92.54±0.27	93.22±0.31	95.18±0.16
Δ ↑	+0.31	+0.88	+0.24	+0.14	+0.36	+0.16	+0.11
GPRGNN (S)	70.23±1.29	79.86±1.81	83.5±0.94	83.5±0.99	92.46±0.29	92.61±0.19	95.16±0.08
GPRGNN (AS)	70.23±1.03	80.66±1.64	83.26±0.69	84.18±0.71	92.68±0.43	93.06±0.49	95.24±0.13
Δ ↑	+0.00	+0.80	-0.24	+0.68	+0.22	+0.45	+0.08
BernNet (S)	68.64±1.54	79.58±0.87	82.37±0.96	85.1±0.97	92.65±0.22	92.3±0.27	95.29±0.15
BernNet (AS)	69.46±1.16	79.62±1.24	82.47±0.56	85.57±0.96	92.82±0.36	93.13±0.37	95.19±0.18
Δ ↑	+0.82	+0.04	+0.10	+0.47	+0.17	+0.83	-0.10

Table 4.10: Test accuracy with/without asymmetric learning on homophilic graphs (Citeseer, Pubmed, Cora, Coauther-CS, Coauther-Physics, Amazon-Computer and Amazon-Photo). High accuracy indicates good performance.

metric learning shows greater performance improvement on datasets with smaller graph sizes. In Table 4.8, there is an average accuracy improvement of 3.08% on smaller graphs on the left. For larger graphs in Table 4.9, improvements are smaller, with a 1.36% increase in accuracy. Specifically, spectral GNNs achieve test accuracy improvements exceeding 10% on smaller datasets such as Texas, Wisconsin, Cornell, and Chameleon, which contain fewer than 1,000 nodes. As smaller training data sizes can often lead to instability in the gradient of empirical loss during training, these findings suggest that Asymmetric learning mitigates this issue by adjusting parameter updates without risking overly aggressive updates.

Q4: How do the exponential decay parameters β_{π_Θ} and β_{π_W} affect the performance of asymmetric learning? We explore the impact of the hyperparameters β_{π_Θ} and β_{π_W} on test accuracy when using ChebNet for node classification tasks. The parameters β_{π_Θ} and β_{π_W} increase from 0 to 0.9 in steps of 0.1, with an additional value at 0.99. Figure 4.2 illustrates that fluctuations in performance are observed only in

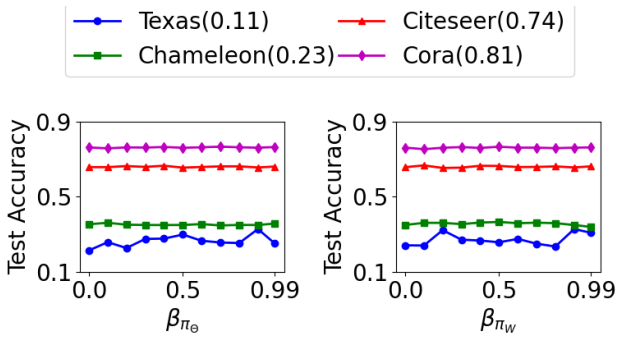


Figure 4.2: The test accuracy of ChebNet when applying asymmetric learning with varying values of β_{π_Θ} and β_{π_W}.

the Texas dataset, while marginal differences are noted in other datasets. This suggests that the performance is not sensitive to β_{π_Θ} and β_{π_W} when training samples are sufficient. Maintaining a training data splitting ratio of 2.5% can ensure an adequate amount of training data for large graphs.

4.5 Summary

In this chapter, we have demonstrated that the parameters Θ and W in spectral GNNs play distinct roles, resulting in a poorly conditioned optimization problem. To address this, we introduced the concept of the block condition number to estimate the optimization difficulty and proposed asymmetric learning, a novel preconditioning technique designed to facilitate the optimization process. Our experiments validate the efficacy of asymmetric learning, particularly in the optimization on heterophilic graphs. These findings underscore the potential of asymmetric learning to enhance the performance and training efficiency of spectral GNNs across diverse graph datasets.